



Parselmouth Documentation

Release 0.5.0.dev0

Yannick Jadoul

Aug 14, 2023

CONTENTS

1	Installation	3
1.1	Basics	3
1.2	Python distributions	3
1.3	PsychoPy	4
1.4	Troubleshooting	5
2	Examples	7
2.1	Plotting	7
2.2	Batch processing of files	12
2.3	Pitch manipulation and Praat commands	15
2.4	PsychoPy experiments	17
2.5	Web service	21
2.6	Projects using Parselmouth	25
3	API Reference	27
3.1	parselmouth	27
3.2	parselmouth.praat	160
4	Citing Parselmouth	165
5	Indices and tables	167
	Python Module Index	169
	Index	171

Parselmouth is a Python library for the [Praat](#) software.

Though other attempts have been made at porting functionality from Praat to Python, Parselmouth is unique in its aim to provide a complete and Pythonic interface to the internal Praat code. While other projects either wrap Praat's scripting language or reimplementing parts of Praat's functionality in Python, Parselmouth directly accesses Praat's C/C++ code (which means the algorithms and their output are exactly the same as in Praat) and provides efficient access to the program's data, but *also* provides an interface that looks no different from any other Python library.

Please note that Parselmouth is currently in premature state and in active development. While the amount of functionality that is currently present is not huge, more will be added over the next few months. As such, *feedback* and possibly *contributions* are highly appreciated.

Drop by our [Gitter chat room](#) or post a message to our [Google discussion group](#) if you have any question, remarks, or requests!

INSTALLATION

1.1 Basics

Parselmouth can be installed like any other Python library, using (a recent version of) the Python package manager `pip`, on Linux, macOS, and Windows:

```
pip install praat-parselmouth
```

To update your installed version to the latest release, add `-U` (or `--upgrade`) to the command:

```
pip install -U praat-parselmouth
```

Warning: While the Python module itself is called `parselmouth`, the Parselmouth package on the Python Package Index has the name `praat-parselmouth`.

Note: To figure out if you can or should update, the version number of your current Parselmouth installation can be found in the `parselmouth.VERSION` variables. The version of Praat on which this version of Parselmouth is based and the release date of that Praat version are available as `PRAAT_VERSION` and `PRAAT_VERSION_DATE`, respectively.

1.2 Python distributions

Anaconda

If you use the Anaconda distribution of Python, you can use the same `pip` command in a terminal of the appropriate Anaconda environment, either activated through the [Anaconda Navigator](#) or [conda tool](#).

Homebrew & MacPorts

We currently do not have Homebrew or MacPorts packages to install Parselmouth. Normally, Parselmouth can just be installed with the accompanying `pip` of these distributions.

PyPy

Binary wheels for recent versions of PyPy are available on the Python Package Index (PyPI), and can be installed with `pip`.

Other

For other distributions of Python, we are expecting that our package is compatible with the Python versions that are out there and that `pip` can handle the installation. If you are using yet another Python distribution, we are definitely interested in hearing about it, so that we can add it to this list!

1.3 PsychoPy

As a Python library, Parselmouth can be used in a PsychoPy experiment. There are two different ways in which PsychoPy can be installed: it can just be manually installed as a standard Python library, in which case Parselmouth can just be installed next to it with `pip`. For Windows and Mac OS X, however, *standalone* versions of PsychoPy exist, and the software does currently not allow for external libraries to be installed with `pip`.

To install Parselmouth in a standalone version of PsychoPy, the following script can be opened and run from within the PsychoPy Coder interface: `psychopy_installation.py`

Note: If running the script results in an error mentioning `TLSV1_ALERT_PROTOCOL_VERSION`, the version of PsychoPy/Python is too old and you will need to follow the manual instructions underneath.

Alternatively, you can follow these steps to manually install Parselmouth into a standalone version of PsychoPy:

0. Find out which version of Python PsychoPy is running.
 - To do so, you can run `import sys; print(sys.version_info)` in the *Shell* tab of the PsychoPy Coder interface. Remember the first two numbers of the version (major and minor; e.g., 3.6).
 - On *Windows*, also run `import platform; print(platform.architecture()[0])` and remember whether the Python executable's architecture is 32bit or 64bit.
1. Go to <https://pypi.org/project/praat-parselmouth/>.
2. Download the file `praat_parselmouth-x.y.z-cpVV-cpVVm-AA.whl` (for *Windows*) or `praat_parselmouth-x.y.z-cpVV-cpVVm-macosx_10_6_intel.whl` (for *Mac OS X*) - where:
 - `x.y.z` will be the version of Parselmouth you want to install
 - `VV` are the first two numbers of the Python version
 - For *Windows*, `AA` is `win32` if you have a 32bit architecture, and `win_amd64` for 64bitBe sure to find the right file in the list, containing both the correct Python version, and `win32/win_amd64` (*Windows*) or `macosx` (*Mac OS X*) in its name!
3. Rename the downloaded file by replacing the `.whl` extension by `.zip`.
4. Extract this zip archive somewhere on your computer, in your directory of choice. Remember the name and location of the extracted folder that contains the file `parselmouth.pyd` (*Windows*) or `parselmouth.so` (*Mac OS X*).
5. Open PsychoPy, open the *Preferences* window, go to the *General* tab.
6. In the *General* tab of the PsychoPy *Preferences*, in the *paths* field, add the folder where you just extracted the Parselmouth library to the list, enclosing the path in quotemarks. (On *Windows*, also replace all `\` characters by `/`.)
 - For example, if the list was empty (`[]`), you could make it look like `['C:/Users/Yannick/parselmouth-psychopy/']` or `['/Users/yannick/parselmouth-psychopy/']`.
 - On *Windows*, to find the right location to enter in the PsychoPy settings, right click `parselmouth.pyd`, choose *Properties*, and look at the *Location* field.
 - On *Mac OS X*, to find the right location to enter in the PsychoPy settings, right click `parselmouth.so`, choose *Get info*, and look at the *where* field.
 - On *Mac OS X*, dragging the folder into a terminal window will also give you the full path with slashes.
7. Click *Ok* to save the PsychoPy settings, close the *Preferences* window, and restart PsychoPy.

8. *Optional:* if you want to check if Parselmouth was installed correctly, open the PsychoPy Coder interface, open the *Shell* tab, and type `import parselmouth`.

- If this results in an error message, please let us know, and we'll try to help you fix what went wrong!
- If this does not give you an error, congratulations, you can now use Parselmouth in your PsychoPy Builder!

Note: These instructions were tested with the standalone versions 3.1.3 and 1.85.2 of PsychoPy. Things might have changed since then, so if running the script or following the manual steps results in an error, please do not hesitate to get in touch.

1.4 Troubleshooting

It is possible that you run into more problems when trying to install or use Parselmouth. Supporting all of the different Python versions out there is not an easy job, as there are plenty of different platforms and setups.

If you run into problems and these common solutions are not solving them, please drop by the [Gitter chat room](#), write a message in the [Google discussion group](#), create a [GitHub issue](#), or write [me](#) a quick email. We would be very happy to solve these problems, so that future users can avoid them!

1.4.1 Multiple Python versions

In case you have multiple installations of Python and don't know which `pip` belongs to which Python version (*looking at you, OS X*):

```
python -m pip install praat-parselmouth
```

Finding out the exact location of the python executable (to call the previous command) for a certain Python installation can be done by typing the following lines in your Python interpreter:

```
>>> import sys
>>> print(sys.executable)
```

If executing this in your Python shell would for example print `/usr/bin/python`, then you would run `/usr/bin/python -m pip install praat-parselmouth` in a terminal to install Parselmouth. (`-U` can again be added to update an already installation to the latest version.)

Combining these two approaches, you can install Parselmouth from within Python itself without knowing where that version of Python is installed:

```
>>> import sys, subprocess
>>> subprocess.call([sys.executable, '-m', 'pip', 'install', 'praat-parselmouth'])
```

Extra arguments to `pip` can be added by inserting them as strings into the list of arguments passed to `subprocess.call` (e.g., to update an existing installation of Parselmouth: `[..., 'install', '-U', 'praat-parselmouth']`).

1.4.2 Pip version

If the standard way to install Parselmouth results in an error or takes a long time, try updating `pip` to the latest version (as `pip` needs to be a reasonably recent version to install the binary, precompiled wheels) by running

```
pip install -U pip
```

If you do not have `pip` installed, you follow these instructions to install `pip`: <https://pip.pypa.io/en/stable/installing/>

1.4.3 ImportError: DLL load failed on Windows

Sometimes on Windows, the installation works, but importing Parselmouth fails with an error message saying `ImportError: DLL load failed: The specified module could not be found..` This error is caused by some missing system files, but can luckily be solved quite easily by installing the “Microsoft Visual C++ Redistributable for Visual Studio 2017”.

The “Microsoft Visual C++ Redistributable for Visual Studio 2019” installer can be downloaded from [Microsoft’s website](#), listed under the “Other Tools and Frameworks” section. These are the direct download links to the relevant files:

- For a 64-bit Python installation: https://aka.ms/vs/16/release/VC_redist.x64.exe
- For a 32-bit Python installation: https://aka.ms/vs/16/release/VC_redist.x86.exe

To check which Python version you are using, you can look at the first line of output when starting a Python shell. The version information should contain `[MSC v.xxxx 64 bit (AMD64)]` in a 64-bit installation, or `[MSC v.xxxx 32 bit (Intel)]` in a 32-bit installation.

EXAMPLES

Parselmouth can be used in various contexts to combine Praat functionality with standard Python features or other Python libraries. The following examples give an idea of the range of possibilities:

2.1 Plotting

Using Parselmouth, it is possible to use the existing Python plotting libraries – such as [Matplotlib](#) and [seaborn](#) – to make custom visualizations of the speech data and analysis results obtained by running Praat’s algorithms.

The following examples visualize an audio recording of someone saying “*The north wind and the sun [...]*”: `the_north_wind_and_the_sun.wav`, extracted from a [Wikipedia Commons](#) audio file.

We start out by importing `parselmouth`, some common Python plotting libraries `matplotlib` and `seaborn`, and the `numpy` numeric library.

```
[1]: import parselmouth

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

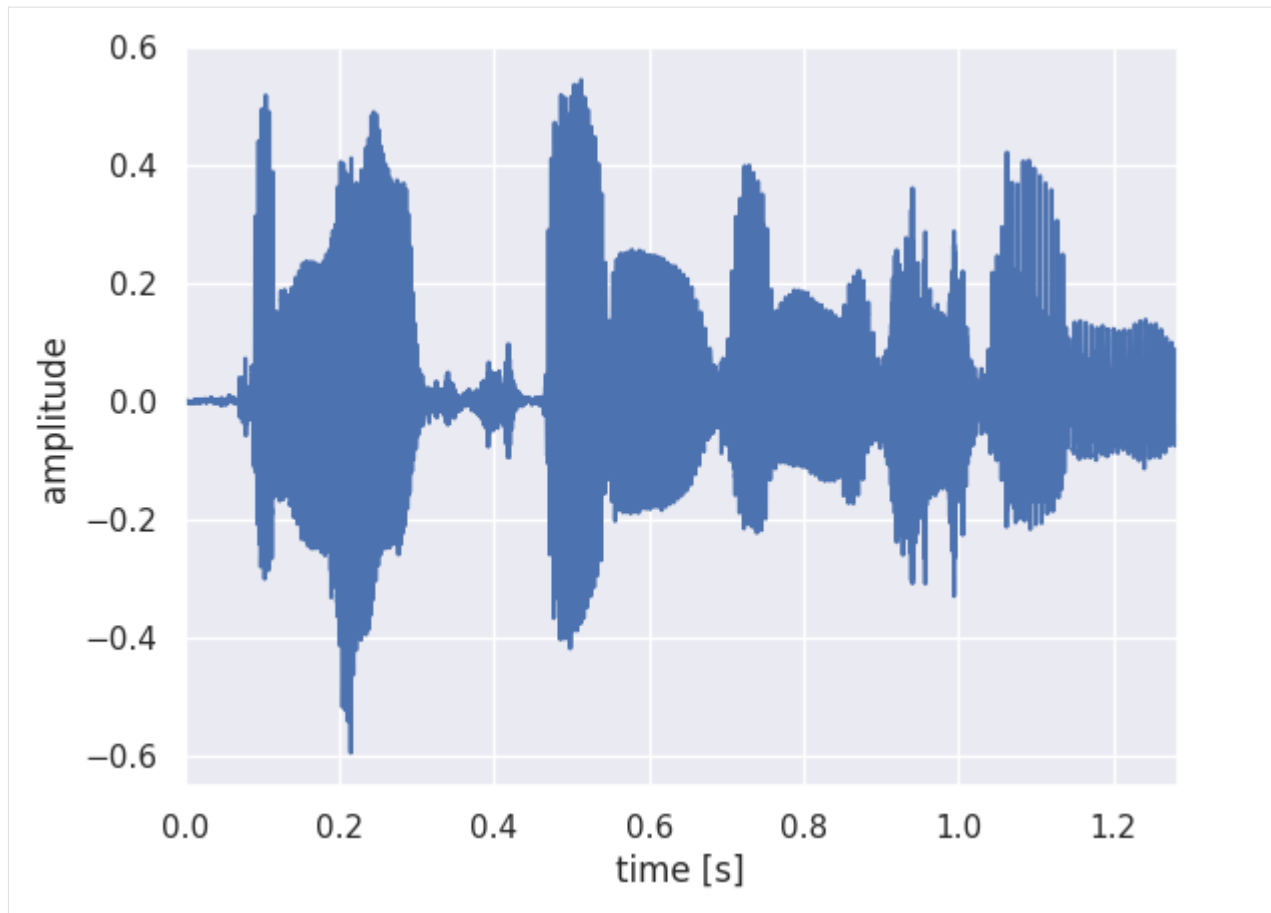
[2]: sns.set() # Use seaborn's default style to make attractive graphs
plt.rcParams['figure.dpi'] = 100 # Show nicely large images in this notebook
```

Once we have the necessary libraries for this example, we open and read in the audio file and plot the raw waveform.

```
[3]: snd = parselmouth.Sound("audio/the_north_wind_and_the_sun.wav")
```

`snd` is now a Parselmouth *Sound* object, and we can access its values and other properties to plot them with the common `matplotlib` Python library:

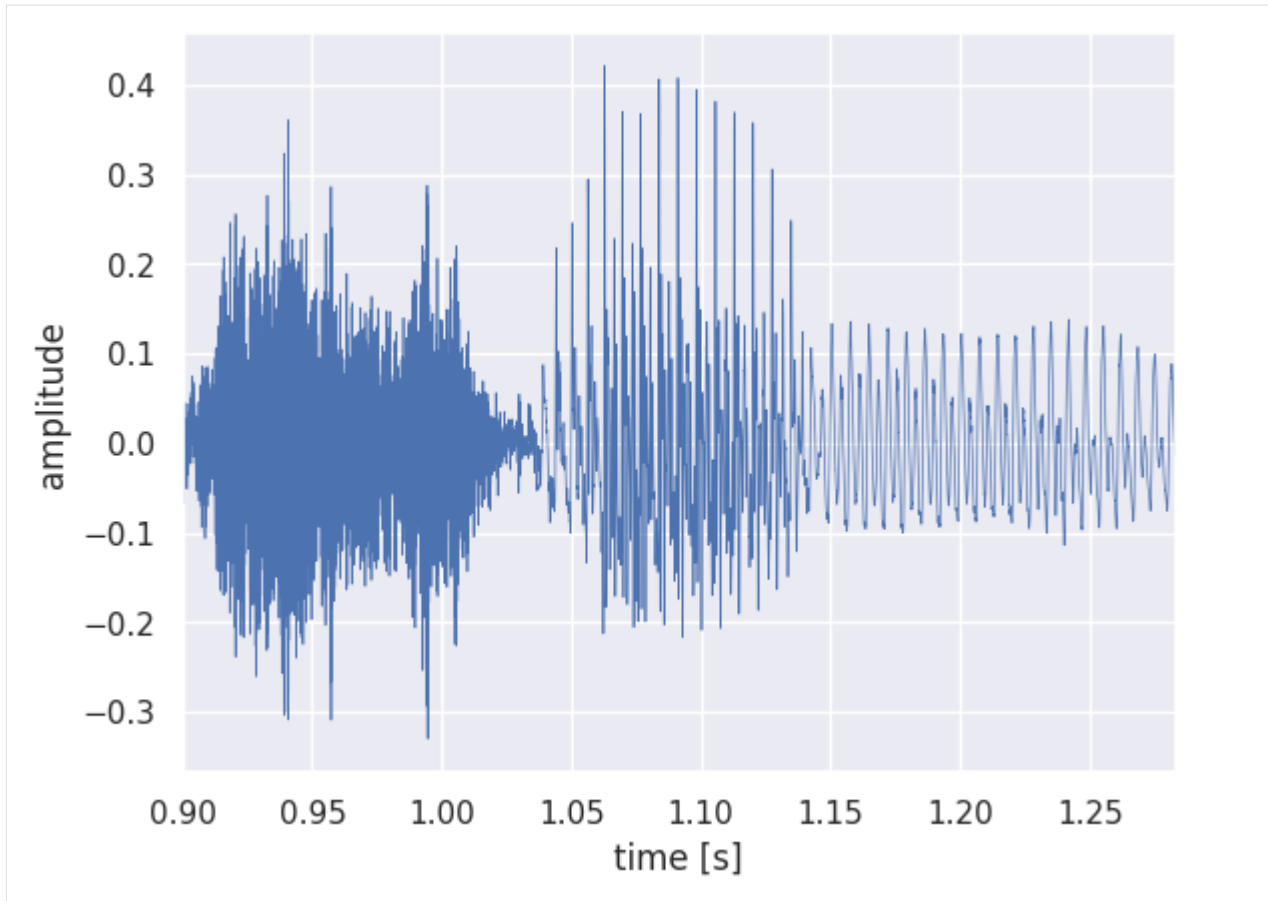
```
[4]: plt.figure()
plt.plot(snd.xs(), snd.values.T)
plt.xlim([snd.xmin, snd.xmax])
plt.xlabel("time [s]")
plt.ylabel("amplitude")
plt.show() # or plt.savefig("sound.png"), or plt.savefig("sound.pdf")
```



It is also possible to extract part of the speech fragment and plot it separately. For example, let's extract the word “sun” and plot its waveform with a finer line.

```
[5]: snd_part = snd.extract_part(from_time=0.9, preserve_times=True)
```

```
[6]: plt.figure()
plt.plot(snd_part.xs(), snd_part.values.T, linewidth=0.5)
plt.xlim([snd_part.xmin, snd_part.xmax])
plt.xlabel("time [s]")
plt.ylabel("amplitude")
plt.show()
```



Next, we can write a couple of ordinary Python functions to plot a Parselmouth Spectrogram and Intensity.

```
[7]: def draw_spectrogram(spectrogram, dynamic_range=70):
    X, Y = spectrogram.x_grid(), spectrogram.y_grid()
    sg_db = 10 * np.log10(spectrogram.values)
    plt.pcolormesh(X, Y, sg_db, vmin=sg_db.max() - dynamic_range, cmap='afmhot')
    plt.ylim([spectrogram.ymin, spectrogram.ymax])
    plt.xlabel("time [s]")
    plt.ylabel("frequency [Hz]")

    def draw_intensity(intensity):
        plt.plot(intensity.xs(), intensity.values.T, linewidth=3, color='w')
        plt.plot(intensity.xs(), intensity.values.T, linewidth=1)
        plt.grid(False)
        plt.ylim(0)
        plt.ylabel("intensity [dB]")
```

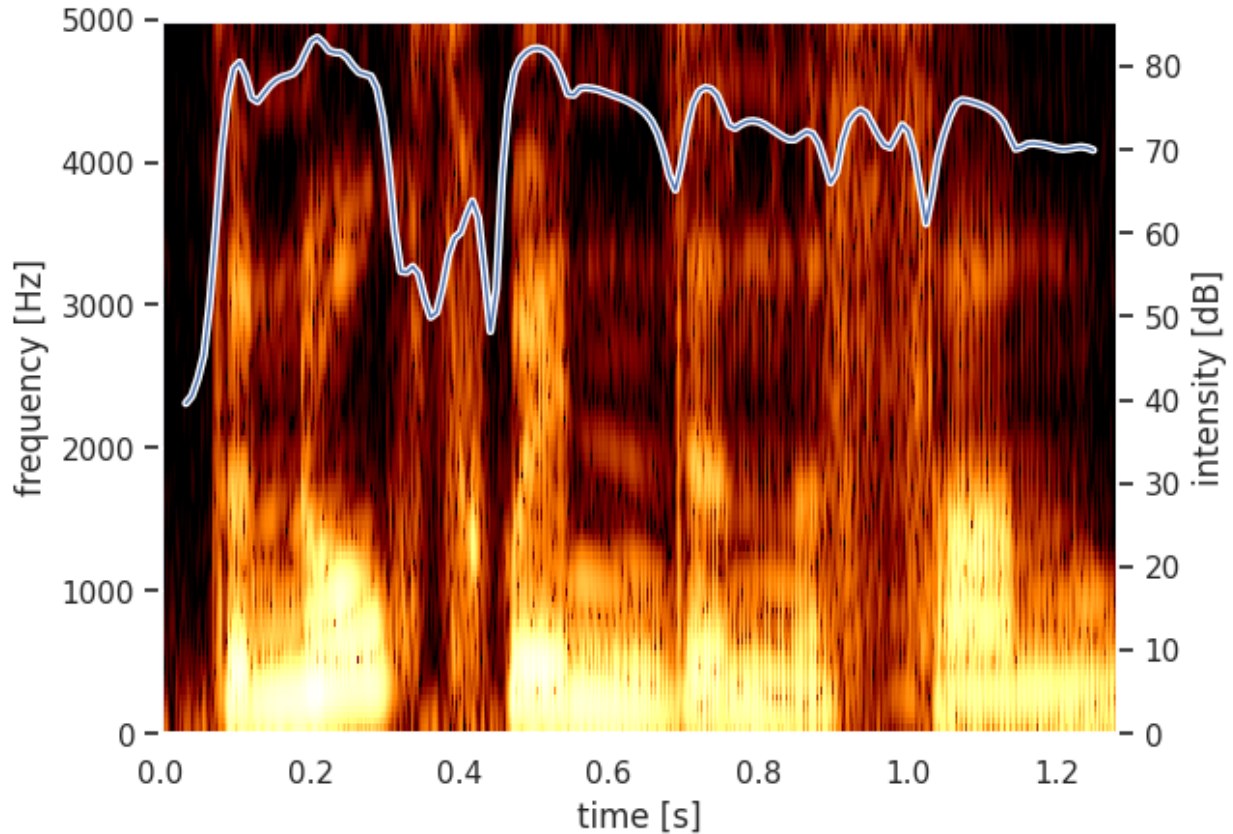
After defining how to plot these, we use Praat (through Parselmouth) to calculate the spectrogram and intensity to actually plot the intensity curve overlaid on the spectrogram.

```
[8]: intensity = snd.to_intensity()
    spectrogram = snd.to_spectrogram()
    plt.figure()
    draw_spectrogram(spectrogram)
```

(continues on next page)

(continued from previous page)

```
plt.twinx()
draw_intensity(intensity)
plt.xlim([snd.xmin, snd.xmax])
plt.show()
```



The Parselmouth functions and methods have the same arguments as the Praat commands, so we can for example also change the window size of the spectrogram analysis to get a narrow-band spectrogram. Next to that, let's now have Praat calculate the pitch of the fragment, so we can plot it instead of the intensity.

```
[9]: def draw_pitch(pitch):
    # Extract selected pitch contour, and
    # replace unvoiced samples by NaN to not plot
    pitch_values = pitch.selected_array['frequency']
    pitch_values[pitch_values==0] = np.nan
    plt.plot(pitch.xs(), pitch_values, 'o', markersize=5, color='w')
    plt.plot(pitch.xs(), pitch_values, 'o', markersize=2)
    plt.grid(False)
    plt.ylim(0, pitch.ceiling)
    plt.ylabel("fundamental frequency [Hz]")
```

```
[10]: pitch = snd.to_pitch()
```

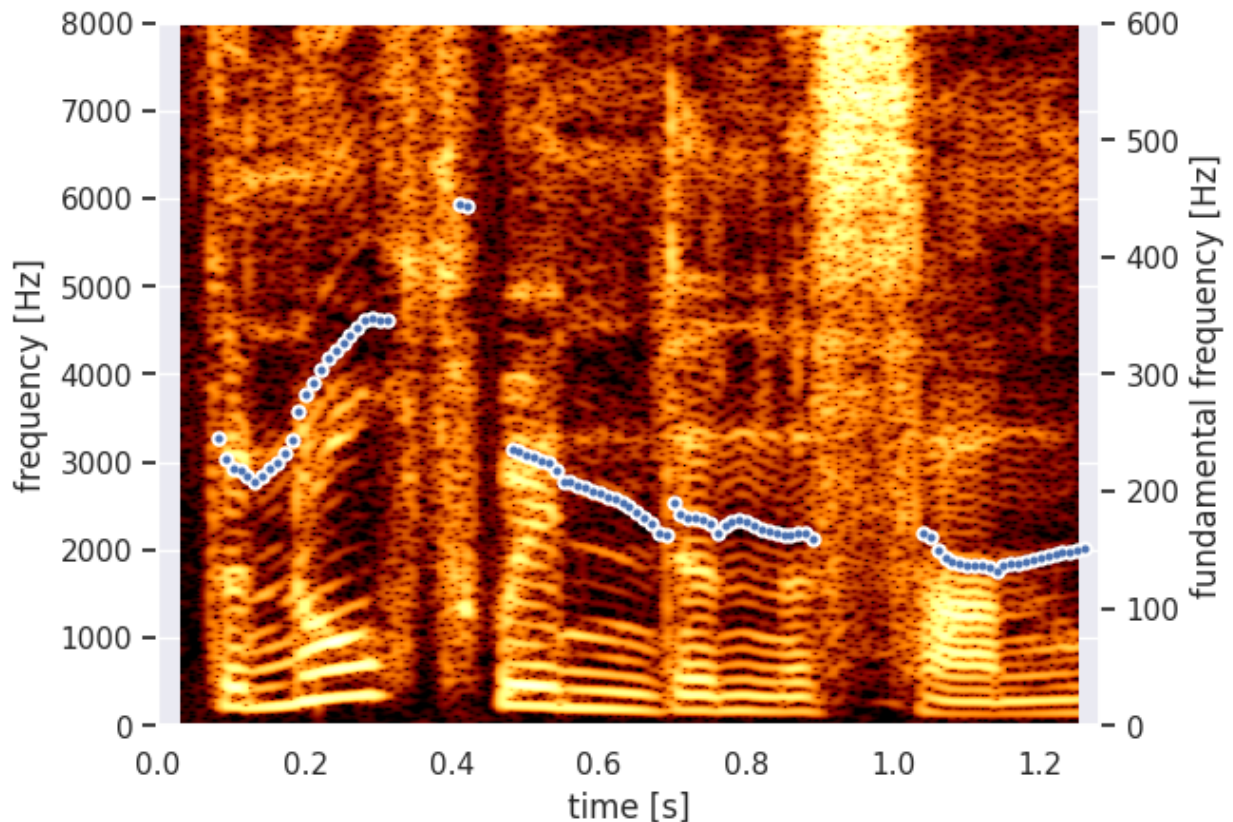
```
[11]: # If desired, pre-emphasize the sound fragment before calculating the spectrogram
pre_emphasized_snd = snd.copy()
```

(continues on next page)

(continued from previous page)

```
pre_emphasized_snd.pre_emphasize()
spectrogram = pre_emphasized_snd.to_spectrogram(window_length=0.03, maximum_
↪ frequency=8000)
```

```
[12]: plt.figure()
draw_spectrogram(spectrogram)
plt.twinx()
draw_pitch(pitch)
plt.xlim([snd.xmin, snd.xmax])
plt.show()
```



Using the [FacetGrid](#) functionality from [seaborn](#), we can even plot a structured grid of multiple custom spectrograms. For example, we will read a CSV file (using the [pandas](#) library) that contains the digit that was spoken, the ID of the speaker and the file name of the audio fragment: `digit_list.csv`, `1_b.wav`, `2_b.wav`, `3_b.wav`, `4_b.wav`, `5_b.wav`, `1_y.wav`, `2_y.wav`, `3_y.wav`, `4_y.wav`, `5_y.wav`

```
[13]: import pandas as pd

def facet_util(data, **kwargs):
    digit, speaker_id = data[['digit', 'speaker_id']].iloc[0]
    sound = parselmouth.Sound("audio/{_}_{_}.wav".format(digit, speaker_id))
    draw_spectrogram(sound.to_spectrogram())
    plt.twinx()
    draw_pitch(sound.to_pitch())
    # If not the rightmost column, then clear the right side axis
```

(continues on next page)

(continued from previous page)

```

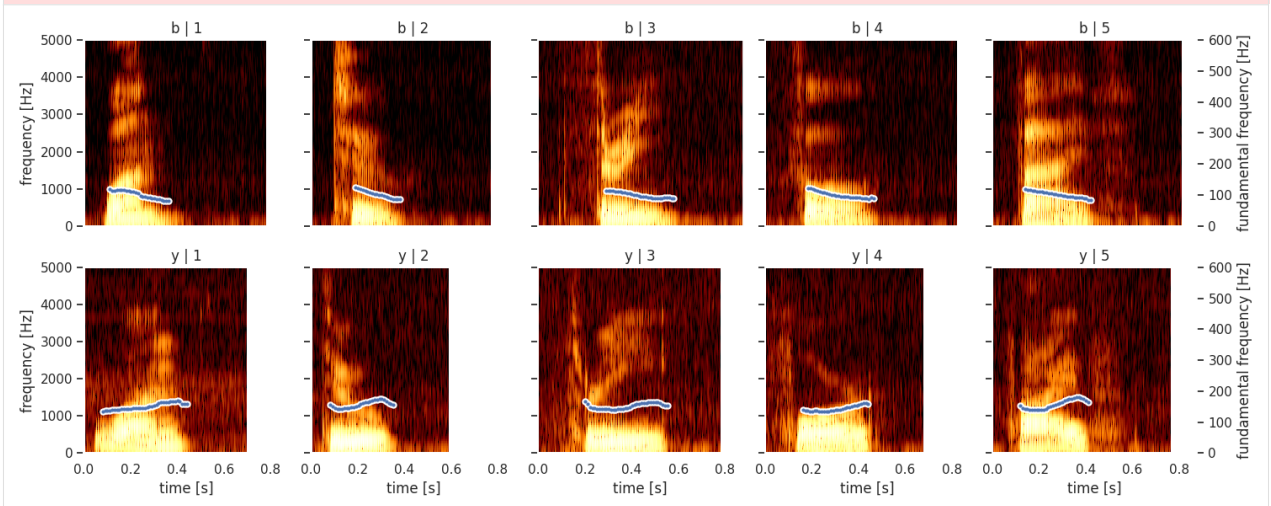
if digit != 5:
    plt.ylabel("")
    plt.yticks([])

results = pd.read_csv("other/digit_list.csv")

grid = sns.FacetGrid(results, row='speaker_id', col='digit')
grid.map_dataframe(facet_util)
grid.set_titles(col_template="{col_name}", row_template="{row_name}")
grid.set_axis_labels("time [s]", "frequency [Hz]")
grid.set(facecolor='white', xlim=(0, None))
plt.show()

/home/docs/checkouts/readthedocs.org/user_builds/parselmouth/envs/docs/lib/python3.11/
↳ site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
↳ tight
self._figure.tight_layout(*args, **kwargs)

```



2.2 Batch processing of files

Using the Python standard libraries (i.e., the `glob` and `os` modules), we can also quickly code up batch operations e.g. over all files with a certain extension in a directory. For example, we can make a list of all `.wav` files in the `audio` directory, use Praat to pre-emphasize these *Sound* objects, and then write the pre-emphasized sound to a `WAV` and `AIFF` format file.

```

[1]: # Find all .wav files in a directory, pre-emphasize and save as new .wav and .aiff file
import parselmouth

import glob
import os.path

for wave_file in glob.glob("audio/*.wav"):
    print("Processing {}".format(wave_file))
    s = parselmouth.Sound(wave_file)
    s.pre_emphasize()

```

(continues on next page)

(continued from previous page)

```
s.save(os.path.splitext(wave_file)[0] + "_pre.wav", 'WAV') # or parselmouth.
↳ SoundFileFormat.WAV instead of 'WAV'
s.save(os.path.splitext(wave_file)[0] + "_pre.aiff", 'AIFF')
```

```
Processing audio/1_y.wav...
Processing audio/2_y.wav...
Processing audio/1_b.wav...
Processing audio/5_b.wav...
Processing audio/5_y.wav...
Processing audio/3_y.wav...
Processing audio/bat.wav...
Processing audio/2_b.wav...
Processing audio/the_north_wind_and_the_sun.wav...
Processing audio/3_b.wav...
Processing audio/bet.wav...
Processing audio/4_b.wav...
Processing audio/4_y.wav...
```

After running this, the original home directory now contains all of the original .wav files pre-emphasised and written again as .wav and .aiff files. The reading, pre-emphasis, and writing are all done by Praat, while looping over all .wav files is done by standard Python code.

[2]: # List the current contents of the audio/ folder

```
!ls audio/

1_b.wav      2_y_pre.aiff  4_b_pre.wav   bat.wav
1_b_pre.aiff 2_y_pre.wav   4_y.wav       bat_pre.aiff
1_b_pre.wav  3_b.wav      4_y_pre.aiff  bat_pre.wav
1_y.wav      3_b_pre.aiff 4_y_pre.wav   bet.wav
1_y_pre.aiff 3_b_pre.wav   5_b.wav       bet_pre.aiff
1_y_pre.wav  3_y.wav      5_b_pre.aiff  bet_pre.wav
2_b.wav      3_y_pre.aiff 5_b_pre.wav   the_north_wind_and_the_sun.wav
2_b_pre.aiff 3_y_pre.wav   5_y.wav       the_north_wind_and_the_sun_pre.aiff
2_b_pre.wav  4_b.wav      5_y_pre.aiff  the_north_wind_and_the_sun_pre.wav
2_y.wav      4_b_pre.aiff 5_y_pre.wav
```

[3]: # Remove the generated audio files again, to clean up the output from this example

```
!rm audio/*_pre.wav
!rm audio/*_pre.aiff
```

Similarly, we can use the [pandas](#) library to read a CSV file with data collected in an experiment, and loop over that data to e.g. extract the mean harmonics-to-noise ratio. The results CSV has the following structure:

condition	...	pp_id
0	...	1877
1	...	801
1	...	2456
0	...	3126

The following code would read such a table, loop over it, use Praat through Parselmouth to calculate the analysis of each row, and then write an augmented CSV file to disk. To illustrate we use an example set of sound fragments: [results.csv](#), [1_b.wav](#), [2_b.wav](#), [3_b.wav](#), [4_b.wav](#), [5_b.wav](#), [1_y.wav](#), [2_y.wav](#), [3_y.wav](#), [4_y.wav](#), [5_y.wav](#)

In our example, the original CSV file, `results.csv` contains the following table:

```
[4]: import pandas as pd
```

```
print(pd.read_csv("other/results.csv"))
```

	condition	pp_id
0	3	y
1	5	y
2	4	b
3	2	y
4	5	b
5	2	b
6	3	b
7	1	y
8	1	b
9	4	y

```
[5]: def analyse_sound(row):
```

```
    condition, pp_id = row['condition'], row['pp_id']
    filepath = "audio/{_}_{_}.wav".format(condition, pp_id)
    sound = parselmouth.Sound(filepath)
    harmonicity = sound.to_harmonicity()
    return harmonicity.values[harmonicity.values != -200].mean()
```

```
# Read in the experimental results file
```

```
dataframe = pd.read_csv("other/results.csv")
```

```
# Apply parselmouth wrapper function row-wise
```

```
dataframe['harmonics_to_noise'] = dataframe.apply(analyse_sound, axis='columns')
```

```
# Write out the updated dataframe
```

```
dataframe.to_csv("processed_results.csv", index=False)
```

We can now have a look at the results by reading in the `processed_results.csv` file again:

```
[6]: print(pd.read_csv("processed_results.csv"))
```

	condition	pp_id	harmonics_to_noise
0	3	y	22.615414
1	5	y	16.403205
2	4	b	17.839167
3	2	y	21.054674
4	5	b	16.092489
5	2	b	12.378289
6	3	b	15.718858
7	1	y	16.704779
8	1	b	12.874451
9	4	y	18.431586

```
[7]: # Clean up, remove the CSV file generated by this example
```

```
!rm processed_results.csv
```

2.3 Pitch manipulation and Praat commands

Another common use of Praat functionality is to manipulate certain features of an existing audio fragment. For example, in the context of a perception experiment one might want to change the pitch contour of an existing audio stimulus while keeping the rest of the acoustic features the same. Parselmouth can then be used to access the Praat algorithms that accomplish this, from Python.

Since this Praat Manipulation functionality has currently not been ported to Parselmouth’s Python interface, we will need to use Parselmouth interface to access *raw* Praat commands.

In this example, we will increase the pitch contour of an audio recording of the word “four”, `4_b.wav`, by one octave. To do so, let’s start by importing Parselmouth and opening the audio file:

```
[1]: import parselmouth

sound = parselmouth.Sound("audio/4_b.wav")
```

We can also listen to this audio fragment:

```
[2]: from IPython.display import Audio
Audio(data=sound.values, rate=sound.sampling_frequency)

[2]: <IPython.lib.display.Audio object>
```

However, now we want to use the Praat Manipulation functionality, but unfortunately, Parselmouth does not yet contain a Manipulation class and the necessary functionality is not directly accessible through the *Sound* object `sound`. To directly access the Praat commands conveniently from Python, we can make use of the *`parselmouth.praat.call`* function.

```
[3]: from parselmouth.praat import call

manipulation = call(sound, "To Manipulation", 0.01, 75, 600)

[4]: type(manipulation)
[4]: parselmouth.Data
```

Note how we first pass in the object(s) that would be selected in Praat’s object list. The next argument to this function is the name of the command as it would be used in a script or can be seen in the Praat user interface. Finally, the arguments to this command’s parameters are passed to the function (in this case, Praat’s default values for “*Time step (s)*”, “*Minimum pitch (Hz)*”, and “*Maximum pitch (Hz)*”). This call to `parselmouth.praat.call` will then return the result of the command as a Python type or Parselmouth object. In this case, a Praat Manipulation object would be created, so our function returns a `parselmouth.Data` object, as a `parselmouth.Manipulation` class does not exist in Parselmouth. However, we can still query the class name the underlying Praat object has:

```
[5]: manipulation.class_name
[5]: 'Manipulation'
```

Next, we can continue using Praat functionality to further use this manipulation object similar to how one would achieve this in Praat. Here, note how we can mix normal Python (e.g. integers and lists), together with the normal use of Parselmouth as Python library (e.g., `sound.xmin`) as well as with the `parselmouth.praat.call` function.

```
[6]: pitch_tier = call(manipulation, "Extract pitch tier")

call(pitch_tier, "Multiply frequencies", sound.xmin, sound.xmax, 2)
```

(continues on next page)

(continued from previous page)

```
call([pitch_tier, manipulation], "Replace pitch tier")
sound_octave_up = call(manipulation, "Get resynthesis (overlap-add)")
```

```
[7]: type(sound_octave_up)
```

```
[7]: parselmouth.Sound
```

The last invocation of `call` resulted in a Praat Sound object being created and returned. Because Parselmouth knows that this type corresponds to a `parselmouth.Sound` Python object, the Python type of this object is not a `parselmouth.Data`. Rather, this object is now equivalent to the one we created at the start of this example. As such, we can use this new object normally, calling methods and accessing its contents. Let's listen and see if we succeeded in increasing the pitch by one octave:

```
[8]: Audio(data=sound_octave_up.values, rate=sound_octave_up.sampling_frequency)
```

```
[8]: <IPython.lib.display.Audio object>
```

And similarly, we could also for example save the sound to a new file.

```
[9]: sound_octave_up.save("4_b_octave_up.wav", "WAV")
```

```
[10]: Audio(filename="4_b_octave_up.wav")
```

```
[10]: <IPython.lib.display.Audio object>
```

```
[11]: # Clean up the created audio file again
      !rm 4_b_octave_up.wav
```

We can of course also turn this combination of commands into a custom function, to be reused in later code:

```
[12]: def change_pitch(sound, factor):
      manipulation = call(sound, "To Manipulation", 0.01, 75, 600)

      pitch_tier = call(manipulation, "Extract pitch tier")

      call(pitch_tier, "Multiply frequencies", sound.xmin, sound.xmax, factor)

      call([pitch_tier, manipulation], "Replace pitch tier")
      return call(manipulation, "Get resynthesis (overlap-add)")
```

Using Jupyter widgets, one can then change the audio file or the pitch change factor, and interactively hear how this sounds.

To try this for yourself, open an online, interactive version of this notebook on Binder! (see link at the top of this notebook)

```
[13]: import ipywidgets
      import glob

      def interactive_change_pitch(audio_file, factor):
          sound = parselmouth.Sound(audio_file)
          sound_changed_pitch = change_pitch(sound, factor)
          return Audio(data=sound_changed_pitch.values, rate=sound_changed_pitch.sampling_
```

(continues on next page)

(continued from previous page)

```
↪ frequency)

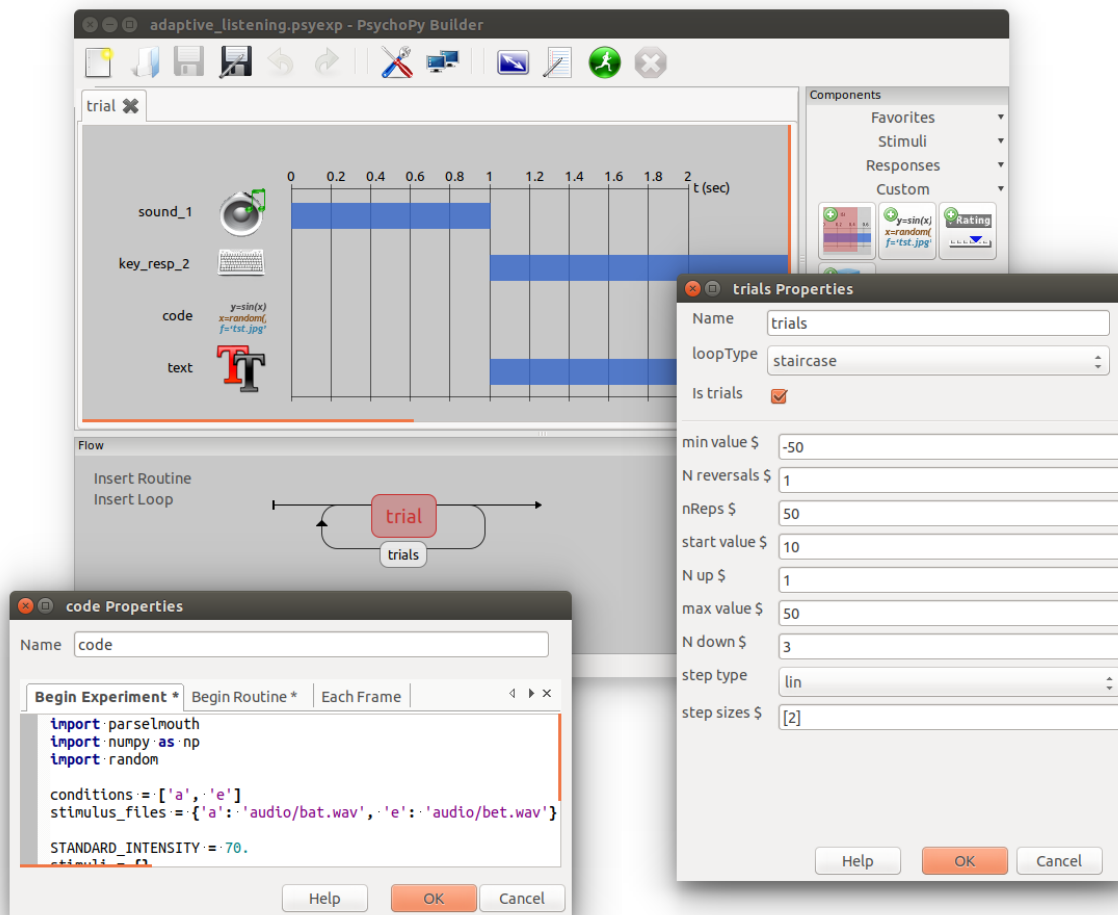
#w = ipywidgets.interact(interactive_change_pitch,
#                          audio_file=ipywidgets.Dropdown(options=sorted(glob.glob("audio/
↪ *.wav")), value="audio/4_b.wav"),
#                          factor=ipywidgets.FloatSlider(min=0.25, max=4, step=0.05,
↪ value=1.5))
```

2.4 PsychoPy experiments

Parselmouth also allows Praat functionality to be included in an interactive [PsychoPy](#) experiment (refer to the subsection on *installing Parselmouth for PsychoPy* for detailed installation instructions for the PsychoPy graphical interface, the *PsychoPy Builder*). The following example shows how easily Python code that uses Parselmouth can be injected in such an experiment; following an adaptive staircase experimental design, at each trial of the experiment a new stimulus is generated based on the responses of the participant. See e.g. Kaernbach, C. (2001). Adaptive threshold estimation with unforced-choice tasks. *Attention, Perception, & Psychophysics*, 63, 1377–1388., or the PsychoPy tutorial at <https://www.psychopy.org/coder/tutorial2.html>.

In this example, we use an adaptive staircase experiment to determine the minimal amount of noise that makes the participant unable to distinguish between two audio fragments, “bat” and “bet” ([bat.wav](#), [bet.wav](#)). At every iteration of the experiment, we want to generate a version of these audio files with a specific signal-to-noise ratio, of course using Parselmouth to do so. Depending on whether the participant correctly identifies whether the noisy stimulus was “bat” or “bet”, the noise level is then either increased or decreased.

As Parselmouth is just another Python library, using it from the PsychoPy *Coder* interface or from a standard Python script that imports the `psychopy` module is quite straightforward. However, PsychoPy also features a so-called *Builder* interface, which is a graphical interface to set up experiments with minimal or no coding. In this *Builder*, a user can create multiple experimental ‘*routines*’ out of different ‘*components*’ and combine them through ‘*loops*’, that can all be configured graphically:



For our simple example, we create a single routine `trial`, with a Sound, a Keyboard, and a Text component. We also insert a loop around this routine of the type `staircase`, such that PsychoPy will take care of the actual implementation of the loop in adaptive staircase design. The full PsychoPy experiment which can be opened in the *Builder* can be downloaded here: [adaptive_listening.psyexp](#)

Finally, to customize the behavior of the `trial` routine and to be able to use Parselmouth inside the PsychoPy experiment, we still add a Code component to the routine. This component will allow us to write Python code that interacts with the rest of the components and with the adaptive staircase loop. The Code components has different tabs, that allow us to insert custom code at different points during the execution of our trial.

First, there is the **Begin Experiment** tab. The code in this tab is executed only once, at the start of the experiment. We use this to set up the Python environment, importing modules and initializing variables, and defining constants:

```
[1]: # ** Begin Experiment **

import parselmouth
import numpy as np
import random

conditions = ['a', 'e']
stimulus_files = {'a': "audio/bat.wav", 'e': "audio/bet.wav"}
```

(continues on next page)

(continued from previous page)

```

STANDARD_INTENSITY = 70.
stimuli = {}
for condition in conditions:
    stimulus = parselmouth.Sound(stimulus_files[condition])
    stimulus.scale_intensity(STANDARD_INTENSITY)
    stimuli[condition] = stimulus

```

The code in the **Begin Routine** tab is executed before the routine, so in our example, for every iteration of the surrounding staircase loop. This allows us to actually use Parselmouth to generate the stimulus that should be played to the participant during this iteration of the routine. To do this, we need to access the current value of the adaptive staircase algorithm: PsychoPy stores this in the Python variable `level`. For example, at some point during the experiment, this could be 10 (representing a signal-to-noise ratio of 10 dB):

```
[2]: level = 10
```

To execute the code we want to put in the **Begin Routine** tab, we need to add a few variables that would be made available by the PsychoPy Builder, normally:

```

[3]: # 'filename' variable is also set by PsychoPy and contains base file name of saved log/
    ↪ output files
filename = "data/participant_staircase_23032017"

# PsychoPy also create a Trials object, containing e.g. information about the current_
    ↪ iteration of the loop
# So let's quickly fake this, in this example, such that the code can be executed without_
    ↪ errors
# In PsychoPy this would be a `psychopy.data.TrialHandler` (https://www.psychopy.org/api/data.html#psychopy.data.TrialHandler)
class MockTrials:
    def addResponse(self, response):
        print("Registering that this trial was {}successful".format(" " if response else
    ↪ "un"))
    trials = MockTrials()
    trials.thisTrialN = 5 # We only need the 'thisTrialN' attribute of the 'trials' variable

# The Sound component can also be accessed by it's name, so let's quickly mock that as_
    ↪ well
# In PsychoPy this would be a `psychopy.sound.Sound` (https://www.psychopy.org/api/sound.html#psychopy.sound.Sound)
    ↪ html#psychopy.sound.Sound)
class MockSound:
    def setSound(self, file_name):
        print("Setting audio file of Sound component to '{}'.format(file_name))
    sound_1 = MockSound()

# And the same for our Keyboard component, `key_resp_2`:
class MockKeyboard:
    pass
key_resp_2 = MockKeyboard()

# Finally, let's also seed the random module to have a consistent output across different_
    ↪ runs
random.seed(42)

```

```
[4]: # Let's also create the directory where we will store our example output
!mkdir data
```

Now, we can execute the code that would be in the **Begin Routine** tab:

```
[5]: # ** Begin Routine **

random_condition = random.choice(conditions)
random_stimulus = stimuli[random_condition]

noise_samples = np.random.normal(size=random_stimulus.n_samples)
noisy_stimulus = parselmouth.Sound(noise_samples,
                                   sampling_frequency=random_stimulus.sampling_frequency)
noisy_stimulus.scale_intensity(STANDARD_INTENSITY - level)
noisy_stimulus.values += random_stimulus.values
noisy_stimulus.scale_intensity(STANDARD_INTENSITY)

# use 'filename' to save our custom stimuli
stimulus_file_name = filename + "_stimulus_" + str(trials.thisTrialN) + ".wav"
noisy_stimulus.resample(44100).save(stimulus_file_name, 'WAV')
sound_1.setSound(stimulus_file_name)

Setting audio file of Sound component to 'data/participant_staircase_23032017_stimulus_5.
↪wav'
```

Let's listen to the file we have just generated and that we would play to the participant:

```
[6]: from IPython.display import Audio
Audio(filename="data/participant_staircase_23032017_stimulus_5.wav")

[6]: <IPython.lib.display.Audio object>
```

In this example, we do not really need to have code executed during the trial (i.e., in the **Each Frame** tab). However, at the end of the trial, we need to inform the PsychoPy staircase loop whether the participant was correct or not, because this will affect the further execution the adaptive staircase, and thus value of the `level` variable set by PsychoPy. For this we add a final line in the **End Routine** tab. Let's say the participant guessed “bat” and pressed the a key:

```
[7]: key_resp_2.keys = 'a'
```

The **End Routine** tab then contains the following code to check the participant's answer against the randomly chosen condition, and to inform the `trials` object of whether the participant was correct:

```
[8]: # ** End Routine **

trials.addResponse(key_resp_2.keys == random_condition)

Registering that this trial was successful
```

```
[9]: # Clean up the output directory again
!rm -r data
```

2.5 Web service

Since Parselmouth is a normal Python library, it can also easily be used within the context of a web server. There are several Python frameworks that allow to quickly set up a web server or web service. In this examples, we will use [Flask](#) to show how easily one can set up a web service that uses Parselmouth to access Praat functionality such as the pitch track estimation algorithms. This functionality can then be accessed by clients without requiring either Praat, Parselmouth, or even Python to be installed, for example within the context of an online experiment.

All that is needed to set up the most basic web server in Flask is a single file. We adapt [the standard Flask example](#) to accept a sound file, access Parselmouth's [Sound.to_pitch](#), and then send back the list of pitch track frequencies. Note that apart from [saving the file that was sent in the HTTP request](#) and encoding the resulting list of frequencies in JSON, the Python code of the `pitch_track` function is the same as one would write in a normal Python script using Parselmouth.

```
[1]: %%writefile server.py

from flask import Flask, request, jsonify
import tempfile

app = Flask(__name__)

@app.route('/pitch_track', methods=['POST'])
def pitch_track():
    import parselmouth

    # Save the file that was sent, and read it into a parselmouth.Sound
    with tempfile.NamedTemporaryFile() as tmp:
        tmp.write(request.files['audio'].read())
        sound = parselmouth.Sound(tmp.name)

    # Calculate the pitch track with Parselmouth
    pitch_track = sound.to_pitch().selected_array['frequency']

    # Convert the NumPy array into a list, then encode as JSON to send back
    return jsonify(list(pitch_track))
```

Writing server.py

Normally, we can then run the server typing `FLASK_APP=server.py flask run` on the command line, as explained in the [Flask documentation](#). Please do note that to run this server publicly, in a secure way and as part of a bigger setup, other options are available to deploy! Refer to the [Flask deployment documentation](#).

However, to run the server from this Jupyter notebook and still be able to run the other cells that access the functionality on the client side, the following code will start the server in a separate thread and print the output of the running server.

```
[2]: import os
import subprocess
import sys
import time

# Start a subprocess that runs the Flask server
p = subprocess.Popen([sys.executable, "-m", "flask", "run"], env=dict(**os.environ,
↪ FLASK_APP="server.py"), stdout=subprocess.PIPE, stderr=subprocess.PIPE)
```

(continues on next page)

(continued from previous page)

```
# Start two subthreads that forward the output from the Flask server to the output of
↳ the Jupyter notebook
def forward(i, o):
    while p.poll() is None:
        l = i.readline().decode('utf-8')
        if l:
            o.write("[SERVER] " + l)

import threading
threading.Thread(target=forward, args=(p.stdout, sys.stdout)).start()
threading.Thread(target=forward, args=(p.stderr, sys.stderr)).start()

# Let's give the server a bit of time to make sure it has started
time.sleep(2)
```

```
[SERVER] * Serving Flask app 'server.py'
[SERVER] * Debug mode: off
```

```
[SERVER] WARNING: This is a development server. Do not use it in a production deployment.
↳ Use a production WSGI server instead.
[SERVER] * Running on http://127.0.0.1:5000
[SERVER] Press CTRL+C to quit
```

Now that the server is up and running, we can make a standard HTTP request to this web service. For example, we can send a Wave file with an audio recording of someone saying “*The north wind and the sun [...]*”: `the_north_wind_and_the_sun.wav`, extracted from a [Wikipedia Commons](#) audio file.

```
[3]: from IPython.display import Audio
Audio(filename="audio/the_north_wind_and_the_sun.wav")

[3]: <IPython.lib.display.Audio object>
```

To do so, we use the `requests` library in this example, but we could use any library to send a standard HTTP request.

```
[4]: import requests
import json

# Load the file to send
files = {'audio': open("audio/the_north_wind_and_the_sun.wav", 'rb')}
# Send the HTTP request and get the reply
reply = requests.post("http://127.0.0.1:5000/pitch_track", files=files)
# Extract the text from the reply and decode the JSON into a list
pitch_track = json.loads(reply.text)
print(pitch_track)

[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 245.46350823786898, 228.46732333120045, 220.229881904913,
↳ 217.9494117767135, 212.32120094882643, 208.42371077564596, 213.3210292245136, 219.
↳ 22164169979897, 225.08564349338334, 232.58018420251648, 243.6102854675347, 267.
↳ 9586673940531, 283.57192373203253, 293.09087794771966, 303.9716558501677, 314.
↳ 16812500255537, 320.11744147538917, 326.34395013825196, 333.3632387299925, 340.
↳ 0277922275489, 345.8240749033839, 348.57743419008335, 346.9665344057159, 346.
↳ 53179321965666, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 445.1355539937184, 442.
↳ 99367847432956, 0.0, 0.0, 0.0, 0.0, 0.0, 236.3912949256524, 233.77304383699934, 231.
↳ 61759183978316, 229.252937317608, 226.5388725505901, 223.6713912521482, 217.
```

(continues on next page)

(continued from previous page)

```

→ 56247158178041, 208.75233223541412, 208.36854272051312, 205.1132684638252, 202.
→ 99628328370704, 200.74245529822406, 198.379243723561, 195.71387722456126, 192.
→ 92640662381228, 189.55087006373063, 186.29856999154498, 182.60612897184708, 178.
→ 0172095327713, 171.7286500573546, 164.43397092360505, 163.15047735066148, 190.
→ 94898597265222, 180.11404296436555, 177.42215658133307, 176.85852955755865, 175.
→ 90234348007218, 172.72381274834703, 165.07291074214982, 170.84308758689093, 173.
→ 84326581969435, 175.39817924857263, 174.73813404735137, 171.30666910901442, 167.
→ 57344824865035, 165.26925804867895, 164.0488248694515, 163.3665771538607, 162.
→ 9182321154844, 164.4049979046003, 164.16734205916592, 160.17875848111373, 0.0, 0.0, 0.
→ 0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 163.57343758482958, 160.
→ 63654708070163, 150.27906547408838, 143.6142724404569, 139.70737167424176, 138.
→ 15535972924215, 137.401926952887, 137.45520345586323, 136.78723483908712, 135.
→ 18334597312617, 132.3066180187801, 136.04747210818914, 138.65745092917942, 139.
→ 1335736781387, 140.238485464634, 141.83711308294014, 143.10991285599226, 144.
→ 40501561368708, 146.07295382762607, 147.47513524525806, 148.1692013818143, 149.
→ 54122031709116, 151.0336292203337]

```

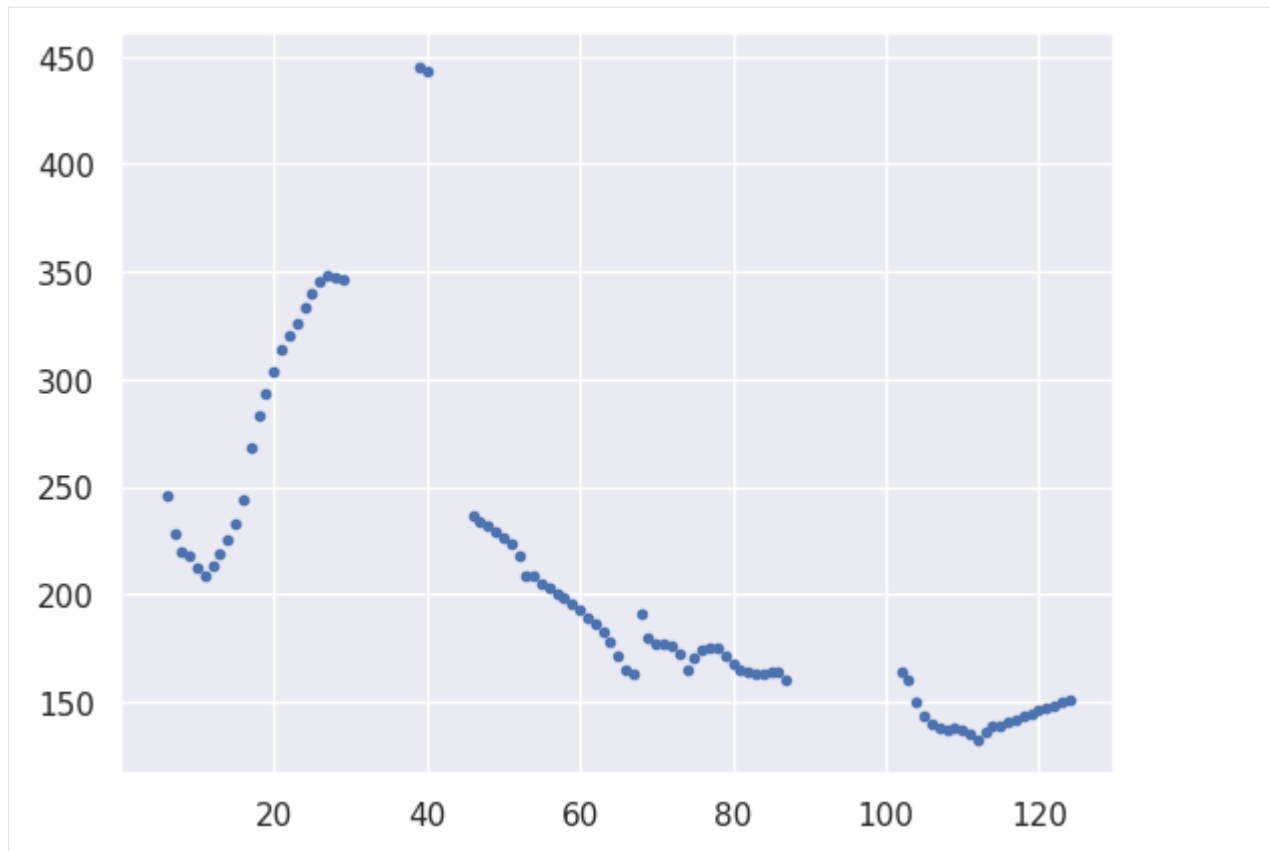
```
[SERVER] 127.0.0.1 - - [14/Aug/2023 22:10:38] "POST /pitch_track HTTP/1.1" 200 -
```

Since we used the standard `json` library from Python to decode the reply from server, `pitch_track` is now a normal list of floats and we can for example plot the estimated pitch track:

```
[5]: import matplotlib.pyplot as plt
import seaborn as sns
```

```
[6]: sns.set() # Use seaborn's default style to make attractive graphs
plt.rcParams['figure.dpi'] = 100 # Show nicely large images in this notebook
```

```
[7]: plt.figure()
plt.plot([float('nan') if x == 0.0 else x for x in pitch_track], '.')
plt.show()
```



Refer to the [examples on plotting](#) for more details on using Parselmouth for plotting.

Importantly, Parselmouth is thus only needed by the server; the client only needs to be able to send a request and read the reply. Consequently, we could even use a different programming language on the client's side. For example, one could make build a HTML page with JavaScript to make the request and do something with the reply:

```
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
  <script type="text/javascript" src="jquery.min.js"></script>
  <script type="text/javascript" src="plotly.min.js"></script>
  <script type="text/javascript">
    var update_plot = function() {
      var audio = document.getElementById("audio").files[0];
      var formData = new FormData();
      formData.append("audio", audio);

      $.getJSON({url: "http://127.0.0.1:5000/pitch_track", method: "POST",
        data: formData, processData: false, contentType: false,
        success: function(data){
          Plotly.newPlot("plot", [{ x: [...Array(data.length).keys()],
            y: data.map(function(x) { return x == 0.
      ↪ 0 ? undefined : x; }},
            type: "lines" }]);});
    };
  </script>
</head>
```

(continues on next page)

(continued from previous page)

```

<body>
<form onsubmit="update_plot(); return false;">
  <input type="file" name="audio" id="audio" />
  <input type="submit" value="Get pitch track" />
  <div id="plot" style="width:1000px;height:600px;"></div>
</form>
</body>

```

Again, one thing to take into account is the security of running such a web server. However, apart from deploying the flask server in a secure and performant way, we also need one extra thing to circumvent a standard security feature of the browser. Without handling Cross Origin Resource Sharing (CORS) on the server, the JavaScript code on the client side will not be able to access the web service's reply. A Flask extension exists however, [Flask-CORS](#), and we refer to its documentation for further details.

```

[8]: # Let's shut down the server
p.kill()

```

```

[9]: # Cleaning up the file that was written to disk
!rm server.py

```

2.6 Projects using Parselmouth

The following projects provide larger, real-life examples and demonstrate the use of Parselmouth:

- The [my-voice-analysis](#) and [myprosody](#) projects by Shahab Sabahi ([@Shahabks](#)) provide Python libraries for voice analysis and acoustical statistics, interfacing Python to his previously developed Praat scripts.
- David R. Feinberg ([@drfeinberg](#)) has written multiple Python scripts and programs with Parselmouth to analyse properties of speech recordings:
 - [Praat Scripts](#) is a collection of Praat scripts used in research, translated into Python.
 - [Voice Lab Software](#) is a GUI application to measure and manipulate voices.

Note: If you have a project using Parselmouth that could be useful for others, and want to add it to this list, do let us know on [Gitter](#)!

API REFERENCE

Parselmouth consists of two main modules, *parselmouth* and *parselmouth.praat*, though both modules will be imported on importing *parselmouth*.

<i>parselmouth</i>	Main module with a Python interface to Praat.
<i>parselmouth.praat</i>	Submodule with functions to call Praat commands and run Praat scripts.

3.1 *parselmouth*

Main module with a Python interface to Praat.

`parselmouth.VERSION = '0.5.0.dev0'`

This version of Parselmouth.

`parselmouth.PRAAT_VERSION = '6.2.23'`

The Praat version on which this version of Parselmouth is based.

`parselmouth.PRAAT_VERSION_DATE = '8 October 2022'`

The release date of the Praat version on which this version of Parselmouth is based.

Functions

<i>read</i>	Read a file into a <i>parselmouth.Data</i> object.
-------------	--

3.1.1 *parselmouth.read*

`parselmouth.read(file_path: str) → parselmouth.Data`

Read a file into a *parselmouth.Data* object.

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

Classes

<i>AmplitudeScaling</i>	
<i>CC</i>	
<i>Data</i>	
<i>Formant</i>	
<i>FormantUnit</i>	
<i>Function</i>	
<i>Harmonicity</i>	
<i>Intensity</i>	
<i>Interpolation</i>	alias of <i>ValueInterpolation</i>
<i>MFCC</i>	
<i>Matrix</i>	
<i>Pitch</i>	
<i>PitchUnit</i>	
<i>Sampled</i>	
<i>SampledXY</i>	
<i>SignalOutsideTimeDomain</i>	
<i>Sound</i>	A fragment of audio, represented by one or multiple channels of floating point values between -1 and 1, sampled at a fixed sampling frequency.
<i>SoundFileFormat</i>	
<i>SpectralAnalysisWindowShape</i>	
<i>Spectrogram</i>	
<i>Spectrum</i>	
<i>TextGrid</i>	
<i>Thing</i>	
<i>TimeFrameSampled</i>	
<i>TimeFunction</i>	
<i>ValueInterpolation</i>	
<i>Vector</i>	
3.1. parselmouth	29
<i>WindowShape</i>	

3.1.2 parselmouth.AmplitudeScaling

class parselmouth.AmplitudeScaling

Bases: pybind11_object

Methods

<code>__init__</code>

Attributes

<code>INTEGRAL</code>

<code>NORMALIZE</code>

<code>PEAK_0_99</code>

<code>SUM</code>

<code>name</code>

<code>value</code>

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.AmplitudeScaling) → int`

`__init__(self: parselmouth.AmplitudeScaling, value: int) → None`

`__init__(self: parselmouth.AmplitudeScaling, arg0: str) → None`

`__int__(self: parselmouth.AmplitudeScaling) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

`__repr__(self: object) → str`

`__str__()`

name(self: handle) -> str

`INTEGRAL = <AmplitudeScaling.INTEGRAL: 1>`

`NORMALIZE = <AmplitudeScaling.NORMALIZE: 3>`

`PEAK_0_99 = <AmplitudeScaling.PEAK_0_99: 4>`

SUM = <AmplitudeScaling.SUM: 2>

property name

property value

3.1.3 parselmouth.CC

class parselmouth.CC

Bases: *TimeFrameSampled*, *Sampled*

Methods

<code>__init__</code>	
<code>copy</code>	
<code>frame_number_to_time</code>	
<code>get_c0_value_in_frame</code>	
<code>get_end_time</code>	
<code>get_frame</code>	
<code>get_frame_number_from_time</code>	
<code>get_number_of_coefficients</code>	
<code>get_number_of_frames</code>	
<code>get_start_time</code>	
<code>get_time_from_frame_number</code>	
<code>get_time_step</code>	
<code>get_total_duration</code>	
<code>get_value_in_frame</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	

continues on next page

Table 1 – continued from previous page

<code>save_as_text_file</code>
<code>scale_times_by</code>
<code>scale_times_to</code>
<code>scale_x_by</code>
<code>scale_x_to</code>
<code>shift_times_by</code>
<code>shift_times_to</code>
<code>shift_x_by</code>
<code>shift_x_to</code>
<code>t_bins</code>
<code>t_grid</code>
<code>time_to_frame_number</code>
<code>to_array</code>
<code>to_matrix</code>
<code>ts</code>
<code>x_bins</code>
<code>x_grid</code>
<code>xs</code>

Attributes

<i>centre_time</i>
<i>class_name</i>
<i>dt</i>
<i>duration</i>
<i>dx</i>
<i>end_time</i>
<i>fmax</i>
<i>fmin</i>
<i>full_name</i>
<i>max_n_coefficients</i>
<i>n_frames</i>
<i>name</i>
<i>nt</i>
<i>nx</i>
<i>start_time</i>
<i>t1</i>
<i>time_range</i>
<i>time_step</i>
<i>tmax</i>
<i>tmin</i>
<i>total_duration</i>
<i>trange</i>
<i>x1</i>
<i>xmax</i>
<i>xmin</i>
<i>xrange</i>

```

class FileFormat
    Bases: pybind11_object
    __eq__(self: object, other: object) → bool
    __hash__(self: object) → int
    __index__(self: parselmouth.Data.FileFormat) → int
    __init__(self: parselmouth.Data.FileFormat, value: int) → None
    __init__(self: parselmouth.Data.FileFormat, arg0: str) → None
    __int__(self: parselmouth.Data.FileFormat) → int
    __ne__(self: object, other: object) → bool
    __new__(**kwargs)
    __repr__(self: object) → str
    __str__()
    name(self: handle) -> str
    BINARY = <FileFormat.BINARY: 2>
    SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
    TEXT = <FileFormat.TEXT: 0>
    property name
    property value

class Frame
    Bases: pybind11_object
    __getitem__(self: parselmouth.CC.Frame, i: int) → float
    __init__(*args, **kwargs)
    __len__(self: parselmouth.CC.Frame) → int
    __new__(**kwargs)
    __setitem__(self: parselmouth.CC.Frame, i: int, value: float) → None
    to_array(self: parselmouth.CC.Frame) → numpy.ndarray[numpy.float64]
    property c
    property c0
    __copy__(self: parselmouth.Data) → parselmouth.Data
    __deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
    __eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
    __getitem__(self: parselmouth.CC, i: int) → parselmouth.CC.Frame
    __getitem__(self: parselmouth.CC, ij: Tuple[int, int]) → float

```

```

__init__(*args, **kwargs)

__iter__(self: parselmouth.CC) → Iterator

__len__(self: parselmouth.Sampled) → int

__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool

__new__(**kwargs)

__setitem__(self: parselmouth.CC, ij: Tuple[int, int], value: float) → None

__str__(self: parselmouth.Thing) → str

copy(self: parselmouth.Data) → parselmouth.Data

frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_c0_value_in_frame(self: parselmouth.CC, frame_number: Positive[int]) → float

get_end_time(self: parselmouth.Function) → float

get_frame(self: parselmouth.CC, frame_number: Positive[int]) → parselmouth.CC.Frame

get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float

get_number_of_coefficients(self: parselmouth.CC, frame_number: Positive[int]) → int

get_number_of_frames(self: parselmouth.Sampled) → int

get_start_time(self: parselmouth.Function) → float

get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_time_step(self: parselmouth.Sampled) → float

get_total_duration(self: parselmouth.Function) → float

get_value_in_frame(self: parselmouth.CC, frame_number: Positive[int], index: Positive[int]) → float

info(self: parselmouth.Thing) → str
    
```

static read(file_path: str) → parselmouth.Data

Read a file into a *parselmouth.Data* object.

Parameters

file_path (str) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

```

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    → None
    
```

```

save_as_binary_file(self: parselmouth.Data, file_path: str) → None
save_as_short_text_file(self: parselmouth.Data, file_path: str) → None
save_as_text_file(self: parselmouth.Data, file_path: str) → None
scale_times_by(self: parselmouth.Function, scale: Positive[float]) → None
scale_times_to(self: parselmouth.Function, new_start_time: float, new_end_time: float) → None
scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None
scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None
shift_times_by(self: parselmouth.Function, seconds: float) → None
shift_times_to(self: parselmouth.Function, time: float, new_time: float) → None
shift_times_to(self: parselmouth.Function, time: str, new_time: float) → None
shift_x_by(self: parselmouth.Function, shift: float) → None
shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None
t_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
t_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
time_to_frame_number(self: parselmouth.Sampled, time: float) → float
to_array(self: parselmouth.CC) → numpy.ndarray[numpy.float64]
to_matrix(self: parselmouth.CC) → parselmouth.Matrix
ts(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
__hash__ = None
property centre_time
property class_name
property dt
property duration
property dx
property end_time
property fmax
property fmin
property full_name

```

```

property max_n_coefficients
property n_frames
property name
property nt
property nx
property start_time
property t1
property time_range
property time_step
property tmax
property tmin
property total_duration
property trange
property x1
property xmax
property xmin
property xrange

```

3.1.4 parselmouth.Data

class `parselmouth.Data`

Bases: *Thing*

Methods

<code>__init__</code>	
<code>copy</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	

Attributes

<i>class_name</i>
<i>full_name</i>
<i>name</i>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object)` → `bool`

`__hash__(self: object)` → `int`

`__index__(self: parselmouth.Data.FileFormat)` → `int`

`__init__(self: parselmouth.Data.FileFormat, value: int)` → `None`

`__init__(self: parselmouth.Data.FileFormat, arg0: str)` → `None`

`__int__(self: parselmouth.Data.FileFormat)` → `int`

`__ne__(self: object, other: object)` → `bool`

`__new__(**kwargs)`

`__repr__(self: object)` → `str`

`__str__()`

`name(self: handle)` → `str`

`BINARY = <FileFormat.BINARY: 2>`

`SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>`

`TEXT = <FileFormat.TEXT: 0>`

property `name`

property `value`

`__copy__(self: parselmouth.Data)` → `parselmouth.Data`

`__deepcopy__(self: parselmouth.Data, memo: dict)` → `parselmouth.Data`

`__eq__(self: parselmouth.Data, other: parselmouth.Data)` → `bool`

`__init__(*args, **kwargs)`

`__ne__(self: parselmouth.Data, other: parselmouth.Data)` → `bool`

`__new__(**kwargs)`

`__str__(self: parselmouth.Thing)` → `str`

`copy(self: parselmouth.Data)` → `parselmouth.Data`

info(*self*: parselmouth.Thing) → str

static read(*file_path*: str) → parselmouth.Data

Read a file into a *parselmouth.Data* object.

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

save(*self*: parselmouth.Data, *file_path*: str, *format*: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>) → None

save_as_binary_file(*self*: parselmouth.Data, *file_path*: str) → None

save_as_short_text_file(*self*: parselmouth.Data, *file_path*: str) → None

save_as_text_file(*self*: parselmouth.Data, *file_path*: str) → None

__hash__ = None

property class_name

property full_name

property name

3.1.5 parselmouth.Formant

class parselmouth.**Formant**

Bases: *TimeFrameSampled*, *Sampled*

Methods

__init__

copy

frame_number_to_time

get_bandwidth_at_time

get_end_time

get_frame_number_from_time

continues on next page

Table 2 – continued from previous page

<code>get_number_of_frames</code>	
<code>get_start_time</code>	
<code>get_time_from_frame_number</code>	
<code>get_time_step</code>	
<code>get_total_duration</code>	
<code>get_value_at_time</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_times_by</code>	
<code>scale_times_to</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_times_by</code>	
<code>shift_times_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>t_bins</code>	
<code>t_grid</code>	
<code>time_to_frame_number</code>	
<code>ts</code>	
<code>x_bins</code>	
<code>x_grid</code>	

continues on next page

Table 2 – continued from previous page

<i>XS</i>

Attributes

<code>centre_time</code>
<code>class_name</code>
<code>dt</code>
<code>duration</code>
<code>dx</code>
<code>end_time</code>
<code>full_name</code>
<code>n_frames</code>
<code>name</code>
<code>nt</code>
<code>nx</code>
<code>start_time</code>
<code>t1</code>
<code>time_range</code>
<code>time_step</code>
<code>tmax</code>
<code>tmin</code>
<code>total_duration</code>
<code>trange</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__`(*self*: *object*, *other*: *object*) → bool

```

__hash__(self: object) → int

__index__(self: parselmouth.Data.FileFormat) → int

__init__(self: parselmouth.Data.FileFormat, value: int) → None
__init__(self: parselmouth.Data.FileFormat, arg0: str) → None

__int__(self: parselmouth.Data.FileFormat) → int

__ne__(self: object, other: object) → bool

__new__(**kwargs)

__repr__(self: object) → str

__str__()
    name(self: handle) -> str
BINARY = <FileFormat.BINARY: 2>
SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
TEXT = <FileFormat.TEXT: 0>

property name
property value

__copy__(self: parselmouth.Data) → parselmouth.Data
__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
__init__(*args, **kwargs)
__len__(self: parselmouth.Sampled) → int
__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
__new__(**kwargs)
__str__(self: parselmouth.Thing) → str
copy(self: parselmouth.Data) → parselmouth.Data
frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float
get_bandwidth_at_time(self: parselmouth.Formant, formant_number: Positive[int], time: float, unit:
    parselmouth.FormantUnit = <FormantUnit.HERTZ: 0>) → float
get_end_time(self: parselmouth.Function) → float
get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float
get_number_of_frames(self: parselmouth.Sampled) → int
get_start_time(self: parselmouth.Function) → float
get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float
    
```

get_time_step(self: *parselmouth.Sampled*) → float

get_total_duration(self: *parselmouth.Function*) → float

get_value_at_time(self: *parselmouth.Formant*, formant_number: *Positive[int]*, time: *float*, unit: *parselmouth.FormantUnit* = <*FormantUnit.HERTZ*: 0>) → float

info(self: *parselmouth.Thing*) → str

static read(file_path: *str*) → *parselmouth.Data*
 Read a file into a *parselmouth.Data* object.

Parameters
file_path (*str*) – The path of the file on disk to read.

Returns
 The Praat Data object that was read.

Return type
parselmouth.Data

See also:
Praat: “Read from file...”

save(self: *parselmouth.Data*, file_path: *str*, format: *parselmouth.Data.FileFormat* = <*FileFormat.TEXT*: 0>) → None

save_as_binary_file(self: *parselmouth.Data*, file_path: *str*) → None

save_as_short_text_file(self: *parselmouth.Data*, file_path: *str*) → None

save_as_text_file(self: *parselmouth.Data*, file_path: *str*) → None

scale_times_by(self: *parselmouth.Function*, scale: *Positive[float]*) → None

scale_times_to(self: *parselmouth.Function*, new_start_time: *float*, new_end_time: *float*) → None

scale_x_by(self: *parselmouth.Function*, scale: *Positive[float]*) → None

scale_x_to(self: *parselmouth.Function*, new_xmin: *float*, new_xmax: *float*) → None

shift_times_by(self: *parselmouth.Function*, seconds: *float*) → None

shift_times_to(self: *parselmouth.Function*, time: *float*, new_time: *float*) → None

shift_times_to(self: *parselmouth.Function*, time: *str*, new_time: *float*) → None

shift_x_by(self: *parselmouth.Function*, shift: *float*) → None

shift_x_to(self: *parselmouth.Function*, x: *float*, new_x: *float*) → None

t_bins(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

t_grid(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

time_to_frame_number(self: *parselmouth.Sampled*, time: *float*) → float

ts(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

x_bins(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

```

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
__hash__ = None
property centre_time
property class_name
property dt
property duration
property dx
property end_time
property full_name
property n_frames
property name
property nt
property nx
property start_time
property t1
property time_range
property time_step
property tmax
property tmin
property total_duration
property trange
property x1
property xmax
property xmin
property xrange

```

3.1.6 parselmouth.FormantUnit

class parselmouth.**FormantUnit**

Bases: pybind11_object

Methods

<code>__init__</code>

Attributes

<code>BARK</code>

<code>HERTZ</code>

<code>name</code>

<code>value</code>

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.FormantUnit) → int`

`__init__(self: parselmouth.FormantUnit, value: int) → None`

`__init__(self: parselmouth.FormantUnit, arg0: str) → None`

`__int__(self: parselmouth.FormantUnit) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

`__repr__(self: object) → str`

`__str__()`

name(self: handle) -> str

BARK = <FormantUnit.BARK: 1>

HERTZ = <FormantUnit.HERTZ: 0>

property name

property value

3.1.7 parselmouth.Function

class `parselmouth.Function`

Bases: `Data`

Methods

<code>__init__</code>	
<code>copy</code>	
<code>info</code>	
<code>read</code>	Read a file into a <code>parselmouth.Data</code> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	

Attributes

<code>class_name</code>
<code>full_name</code>
<code>name</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>

class `FileFormat`

Bases: `pybind11_object`

```

__eq__(self: object, other: object) → bool
__hash__(self: object) → int
__index__(self: parselmouth.Data.FileFormat) → int
__init__(self: parselmouth.Data.FileFormat, value: int) → None
__init__(self: parselmouth.Data.FileFormat, arg0: str) → None
__int__(self: parselmouth.Data.FileFormat) → int
__ne__(self: object, other: object) → bool
__new__(**kwargs)
__repr__(self: object) → str
__str__()
    name(self: handle) -> str
BINARY = <FileFormat.BINARY: 2>
SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
TEXT = <FileFormat.TEXT: 0>
property name
property value
__copy__(self: parselmouth.Data) → parselmouth.Data
__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
__init__(*args, **kwargs)
__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
__new__(**kwargs)
__str__(self: parselmouth.Thing) → str
copy(self: parselmouth.Data) → parselmouth.Data
info(self: parselmouth.Thing) → str
static read(file_path: str) → parselmouth.Data
    Read a file into a parselmouth.Data object.
        Parameters
            file_path (str) – The path of the file on disk to read.
        Returns
            The Praat Data object that was read.
        Return type
            parselmouth.Data

```

See also:

Praat: “*Read from file...*”

```

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    → None

save_as_binary_file(self: parselmouth.Data, file_path: str) → None

save_as_short_text_file(self: parselmouth.Data, file_path: str) → None

save_as_text_file(self: parselmouth.Data, file_path: str) → None

scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None

shift_x_by(self: parselmouth.Function, shift: float) → None

shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None

__hash__ = None

property class_name

property full_name

property name

property xmax

property xmin

property xrange

```

3.1.8 parselmouth.Harmonicity

```

class parselmouth.Harmonicity
    Bases: TimeFrameSampled, Vector

```

Methods

<code>__init__</code>
<code>add</code>
<code>as_array</code>
<code>at_xy</code>
<code>copy</code>
<code>divide</code>
<code>formula</code>
<code>frame_number_to_time</code>

continues on next page

Table 3 – continued from previous page

<code>get_column_distance</code>	
<code>get_end_time</code>	
<code>get_frame_number_from_time</code>	
<code>get_highest_x</code>	
<code>get_highest_y</code>	
<code>get_lowest_x</code>	
<code>get_lowest_y</code>	
<code>get_maximum</code>	
<code>get_minimum</code>	
<code>get_number_of_columns</code>	
<code>get_number_of_frames</code>	
<code>get_number_of_rows</code>	
<code>get_row_distance</code>	
<code>get_start_time</code>	
<code>get_sum</code>	
<code>get_time_from_frame_number</code>	
<code>get_time_step</code>	
<code>get_total_duration</code>	
<code>get_value</code>	
<code>get_value_at_xy</code>	
<code>get_value_in_cell</code>	
<code>get_x_of_column</code>	
<code>get_y_of_row</code>	
<code>info</code>	
<code>multiply</code>	
<code>read</code>	Read a file into a <code>parselmouth.Data</code> object.

continues on next page

Table 3 – continued from previous page

<code>save</code>
<code>save_as_binary_file</code>
<code>save_as_headerless_spreadsheet_file</code>
<code>save_as_matrix_text_file</code>
<code>save_as_short_text_file</code>
<code>save_as_text_file</code>
<code>scale</code>
<code>scale_peak</code>
<code>scale_times_by</code>
<code>scale_times_to</code>
<code>scale_x_by</code>
<code>scale_x_to</code>
<code>set_value</code>
<code>shift_times_by</code>
<code>shift_times_to</code>
<code>shift_x_by</code>
<code>shift_x_to</code>
<code>subtract</code>
<code>subtract_mean</code>
<code>t_bins</code>
<code>t_grid</code>
<code>time_to_frame_number</code>
<code>ts</code>
<code>x_bins</code>
<code>x_grid</code>
<code>xs</code>

continues on next page

Table 3 – continued from previous page

<i>y_bins</i>
<i>y_grid</i>
<i>ys</i>

Attributes

<i>centre_time</i>
<i>class_name</i>
<i>dt</i>
<i>duration</i>
<i>dx</i>
<i>dy</i>
<i>end_time</i>
<i>full_name</i>
<i>n_columns</i>
<i>n_frames</i>
<i>n_rows</i>
<i>name</i>
<i>nt</i>
<i>nx</i>
<i>ny</i>
<i>start_time</i>
<i>t1</i>
<i>time_range</i>
<i>time_step</i>
<i>tmax</i>

continues on next page

Table 4 – continued from previous page

<code>tmin</code>
<code>total_duration</code>
<code>trange</code>
<code>values</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>
<code>y1</code>
<code>ymax</code>
<code>ymin</code>
<code>yrange</code>

```

class FileFormat
    Bases: pybind11_object
    __eq__(self: object, other: object) → bool
    __hash__(self: object) → int
    __index__(self: parselmouth.Data.FileFormat) → int
    __init__(self: parselmouth.Data.FileFormat, value: int) → None
    __init__(self: parselmouth.Data.FileFormat, arg0: str) → None
    __int__(self: parselmouth.Data.FileFormat) → int
    __ne__(self: object, other: object) → bool
    __new__(**kwargs)
    __repr__(self: object) → str
    __str__()
        name(self: handle) -> str
    BINARY = <FileFormat.BINARY: 2>
    SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
    TEXT = <FileFormat.TEXT: 0>
    property name

```

```

    property value

__add__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__copy__(self: parselmouth.Data) → parselmouth.Data
__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
__iadd__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__imul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__init__(*args, **kwargs)
__isub__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__itruediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__len__(self: parselmouth.Sampled) → int
__mul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
__new__(**kwargs)
__radd__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__rmul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__str__(self: parselmouth.Thing) → str
__sub__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__truediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
add(self: parselmouth.Vector, number: float) → None
as_array(self: parselmouth.Matrix) → numpy.ndarray[numpy.float64]
at_xy(self: parselmouth.Matrix, x: float, y: float) → float
copy(self: parselmouth.Data) → parselmouth.Data
divide(self: parselmouth.Vector, factor: float) → None
formula(self: parselmouth.Matrix, formula: str, from_x: float | None = None, to_x: float | None = None,
        from_y: float | None = None, to_y: float | None = None) → None
formula(self: parselmouth.Matrix, formula: str, x_range: Tuple[float | None, float | None] = (None, None),
        y_range: Tuple[float | None, float | None] = (None, None)) → None
frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float
get_column_distance(self: parselmouth.Matrix) → float
get_end_time(self: parselmouth.Function) → float
get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float

```

```

get_highest_x(self: parselmouth.Matrix) → float
get_highest_y(self: parselmouth.Matrix) → float
get_lowest_x(self: parselmouth.Matrix) → float
get_lowest_y(self: parselmouth.Matrix) → float
get_maximum(self: parselmouth.Matrix) → float
get_minimum(self: parselmouth.Matrix) → float
get_number_of_columns(self: parselmouth.Matrix) → int
get_number_of_frames(self: parselmouth.Sampled) → int
get_number_of_rows(self: parselmouth.Matrix) → int
get_row_distance(self: parselmouth.Matrix) → float
get_start_time(self: parselmouth.Function) → float
get_sum(self: parselmouth.Matrix) → float
get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float
get_time_step(self: parselmouth.Sampled) → float
get_total_duration(self: parselmouth.Function) → float
get_value(self: parselmouth.Harmonicity, time: float, interpolation: parselmouth.ValueInterpolation =
    <ValueInterpolation.CUBIC: 2>) → float
get_value_at_xy(self: parselmouth.Matrix, x: float, y: float) → float
get_value_in_cell(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int])
    → float
get_x_of_column(self: parselmouth.Matrix, column_number: Positive[int]) → float
get_y_of_row(self: parselmouth.Matrix, row_number: Positive[int]) → float
info(self: parselmouth.Thing) → str
multiply(self: parselmouth.Vector, factor: float) → None
static read(file_path: str) → parselmouth.Data
    Read a file into a parselmouth.Data object.

```

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

```

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    → None

save_as_binary_file(self: parselmouth.Data, file_path: str) → None

save_as_headerless_spreadsheet_file(self: parselmouth.Matrix, file_path: str) → None

save_as_matrix_text_file(self: parselmouth.Matrix, file_path: str) → None

save_as_short_text_file(self: parselmouth.Data, file_path: str) → None

save_as_text_file(self: parselmouth.Data, file_path: str) → None

scale(self: parselmouth.Vector, scale: Positive[float]) → None

scale_peak(self: parselmouth.Vector, new_peak: Positive[float] = 0.99) → None

scale_times_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_times_to(self: parselmouth.Function, new_start_time: float, new_end_time: float) → None

scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None

set_value(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int], new_value:
    float) → None

shift_times_by(self: parselmouth.Function, seconds: float) → None

shift_times_to(self: parselmouth.Function, time: float, new_time: float) → None

shift_times_to(self: parselmouth.Function, time: str, new_time: float) → None

shift_x_by(self: parselmouth.Function, shift: float) → None

shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None

subtract(self: parselmouth.Vector, number: float) → None

subtract_mean(self: parselmouth.Vector) → None

t_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

t_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

time_to_frame_number(self: parselmouth.Sampled, time: float) → float

ts(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

y_bins(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

y_grid(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

ys(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

```

`__hash__ = None`
`property centre_time`
`property class_name`
`property dt`
`property duration`
`property dx`
`property dy`
`property end_time`
`property full_name`
`property n_columns`
`property n_frames`
`property n_rows`
`property name`
`property nt`
`property nx`
`property ny`
`property start_time`
`property t1`
`property time_range`
`property time_step`
`property tmax`
`property tmin`
`property total_duration`
`property trange`
`property values`
`property x1`
`property xmax`
`property xmin`
`property xrange`
`property y1`
`property ymax`

property `ymin`

property `yrange`

3.1.9 `parselmouth.Intensity`

class `parselmouth.Intensity`

Bases: *TimeFrameSampled*, *Vector*

Methods

<code>__init__</code>
<code>add</code>
<code>as_array</code>
<code>at_xy</code>
<code>copy</code>
<code>divide</code>
<code>formula</code>
<code>frame_number_to_time</code>
<code>get_average</code>
<code>get_column_distance</code>
<code>get_end_time</code>
<code>get_frame_number_from_time</code>
<code>get_highest_x</code>
<code>get_highest_y</code>
<code>get_lowest_x</code>
<code>get_lowest_y</code>
<code>get_maximum</code>
<code>get_minimum</code>
<code>get_number_of_columns</code>

continues on next page

Table 5 – continued from previous page

<code>get_number_of_frames</code>	
<code>get_number_of_rows</code>	
<code>get_row_distance</code>	
<code>get_start_time</code>	
<code>get_sum</code>	
<code>get_time_from_frame_number</code>	
<code>get_time_step</code>	
<code>get_total_duration</code>	
<code>get_value</code>	
<code>get_value_at_xy</code>	
<code>get_value_in_cell</code>	
<code>get_x_of_column</code>	
<code>get_y_of_row</code>	
<code>info</code>	
<code>multiply</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_headerless_spreadsheet_file</code>	
<code>save_as_matrix_text_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale</code>	
<code>scale_peak</code>	
<code>scale_times_by</code>	
<code>scale_times_to</code>	

continues on next page

Table 5 – continued from previous page

<code>scale_x_by</code>
<code>scale_x_to</code>
<code>set_value</code>
<code>shift_times_by</code>
<code>shift_times_to</code>
<code>shift_x_by</code>
<code>shift_x_to</code>
<code>subtract</code>
<code>subtract_mean</code>
<code>t_bins</code>
<code>t_grid</code>
<code>time_to_frame_number</code>
<code>ts</code>
<code>x_bins</code>
<code>x_grid</code>
<code>xs</code>
<code>y_bins</code>
<code>y_grid</code>
<code>ys</code>

Attributes

<code>centre_time</code>
<code>class_name</code>
<code>dt</code>
<code>duration</code>

continues on next page

Table 6 – continued from previous page

<i>dx</i>
<i>dy</i>
<i>end_time</i>
<i>full_name</i>
<i>n_columns</i>
<i>n_frames</i>
<i>n_rows</i>
<i>name</i>
<i>nt</i>
<i>nx</i>
<i>ny</i>
<i>start_time</i>
<i>t1</i>
<i>time_range</i>
<i>time_step</i>
<i>tmax</i>
<i>tmin</i>
<i>total_duration</i>
<i>trange</i>
<i>values</i>
<i>x1</i>
<i>xmax</i>
<i>xmin</i>
<i>xrange</i>
<i>y1</i>
<i>ymax</i>

continues on next page

Table 6 – continued from previous page

<i>ymin</i>
<i>yrange</i>

class AveragingMethod

Bases: pybind11_object

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.Intensity.AveragingMethod) → int`

`__init__(self: parselmouth.Intensity.AveragingMethod, value: int) → None`

`__init__(self: parselmouth.Intensity.AveragingMethod, arg0: str) → None`

`__int__(self: parselmouth.Intensity.AveragingMethod) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

`__repr__(self: object) → str`

`__str__()`

name(self: handle) -> str

DB = <AveragingMethod.DB: 3>

ENERGY = <AveragingMethod.ENERGY: 1>

MEDIAN = <AveragingMethod.MEDIAN: 0>

SONES = <AveragingMethod.SONES: 2>

property name

property value

class FileFormat

Bases: pybind11_object

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.Data.FileFormat) → int`

`__init__(self: parselmouth.Data.FileFormat, value: int) → None`

`__init__(self: parselmouth.Data.FileFormat, arg0: str) → None`

`__int__(self: parselmouth.Data.FileFormat) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

```

    __repr__(self: object) → str
    __str__()
        name(self: handle) -> str
    BINARY = <FileFormat.BINARY: 2>
    SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
    TEXT = <FileFormat.TEXT: 0>
    property name
    property value
    __add__(self: parselmouth.Vector, number: float) → parselmouth.Vector
    __copy__(self: parselmouth.Data) → parselmouth.Data
    __deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
    __eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
    __iadd__(self: parselmouth.Vector, number: float) → parselmouth.Vector
    __imul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
    __init__(*args, **kwargs)
    __isub__(self: parselmouth.Vector, number: float) → parselmouth.Vector
    __itruediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
    __len__(self: parselmouth.Sampled) → int
    __mul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
    __ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
    __new__(**kwargs)
    __radd__(self: parselmouth.Vector, number: float) → parselmouth.Vector
    __rmul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
    __str__(self: parselmouth.Thing) → str
    __sub__(self: parselmouth.Vector, number: float) → parselmouth.Vector
    __truediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
    add(self: parselmouth.Vector, number: float) → None
    as_array(self: parselmouth.Matrix) → numpy.ndarray[numpy.float64]
    at_xy(self: parselmouth.Matrix, x: float, y: float) → float
    copy(self: parselmouth.Data) → parselmouth.Data

```

```

divide(self: parselmouth.Vector, factor: float) → None

formula(self: parselmouth.Matrix, formula: str, from_x: float | None = None, to_x: float | None = None,
         from_y: float | None = None, to_y: float | None = None) → None

formula(self: parselmouth.Matrix, formula: str, x_range: Tuple[float | None, float | None] = (None, None),
         y_range: Tuple[float | None, float | None] = (None, None)) → None

frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_average(self: parselmouth.Intensity, from_time: Optional[float] = None, to_time: Optional[float] =
             None, averaging_method: parselmouth.Intensity.AveragingMethod =
             <AveragingMethod.ENERGY: 1>) → float

get_column_distance(self: parselmouth.Matrix) → float

get_end_time(self: parselmouth.Function) → float

get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float

get_highest_x(self: parselmouth.Matrix) → float

get_highest_y(self: parselmouth.Matrix) → float

get_lowest_x(self: parselmouth.Matrix) → float

get_lowest_y(self: parselmouth.Matrix) → float

get_maximum(self: parselmouth.Matrix) → float

get_minimum(self: parselmouth.Matrix) → float

get_number_of_columns(self: parselmouth.Matrix) → int

get_number_of_frames(self: parselmouth.Sampled) → int

get_number_of_rows(self: parselmouth.Matrix) → int

get_row_distance(self: parselmouth.Matrix) → float

get_start_time(self: parselmouth.Function) → float

get_sum(self: parselmouth.Matrix) → float

get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_time_step(self: parselmouth.Sampled) → float

get_total_duration(self: parselmouth.Function) → float

get_value(self: parselmouth.Intensity, time: float, interpolation: parselmouth.ValueInterpolation =
             <ValueInterpolation.CUBIC: 2>) → float

get_value_at_xy(self: parselmouth.Matrix, x: float, y: float) → float

get_value_in_cell(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int])
                 → float

get_x_of_column(self: parselmouth.Matrix, column_number: Positive[int]) → float

```

get_y_of_row(self: *parselmouth.Matrix*, row_number: *Positive[int]*) → float

info(self: *parselmouth.Thing*) → str

multiply(self: *parselmouth.Vector*, factor: *float*) → None

static read(file_path: *str*) → *parselmouth.Data*

Read a file into a *parselmouth.Data* object.

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

save(self: *parselmouth.Data*, file_path: *str*, format: *parselmouth.Data.FileFormat* = <*FileFormat.TEXT*: 0>) → None

save_as_binary_file(self: *parselmouth.Data*, file_path: *str*) → None

save_as_headerless_spreadsheet_file(self: *parselmouth.Matrix*, file_path: *str*) → None

save_as_matrix_text_file(self: *parselmouth.Matrix*, file_path: *str*) → None

save_as_short_text_file(self: *parselmouth.Data*, file_path: *str*) → None

save_as_text_file(self: *parselmouth.Data*, file_path: *str*) → None

scale(self: *parselmouth.Vector*, scale: *Positive[float]*) → None

scale_peak(self: *parselmouth.Vector*, new_peak: *Positive[float]* = 0.99) → None

scale_times_by(self: *parselmouth.Function*, scale: *Positive[float]*) → None

scale_times_to(self: *parselmouth.Function*, new_start_time: *float*, new_end_time: *float*) → None

scale_x_by(self: *parselmouth.Function*, scale: *Positive[float]*) → None

scale_x_to(self: *parselmouth.Function*, new_xmin: *float*, new_xmax: *float*) → None

set_value(self: *parselmouth.Matrix*, row_number: *Positive[int]*, column_number: *Positive[int]*, new_value: *float*) → None

shift_times_by(self: *parselmouth.Function*, seconds: *float*) → None

shift_times_to(self: *parselmouth.Function*, time: *float*, new_time: *float*) → None

shift_times_to(self: *parselmouth.Function*, time: *str*, new_time: *float*) → None

shift_x_by(self: *parselmouth.Function*, shift: *float*) → None

shift_x_to(self: *parselmouth.Function*, x: *float*, new_x: *float*) → None

subtract(self: *parselmouth.Vector*, number: *float*) → None

```

subtract_mean(self: parselmouth.Vector) → None

t_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

t_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

time_to_frame_number(self: parselmouth.Sampled, time: float) → float

ts(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

y_bins(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

y_grid(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

ys(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

__hash__ = None

property centre_time

property class_name

property dt

property duration

property dx

property dy

property end_time

property full_name

property n_columns

property n_frames

property n_rows

property name

property nt

property nx

property ny

property start_time

property t1

property time_range

property time_step

```

```

property tmax
property tmin
property total_duration
property trange
property values
property x1
property xmax
property xmin
property xrange
property y1
property ymax
property ymin
property yrange

```

3.1.10 parselmouth.Interpolation

`parselmouth.Interpolation`
 alias of *ValueInterpolation*

3.1.11 parselmouth.MFCC

`class parselmouth.MFCC`
 Bases: *CC*

Methods

<code>__init__</code>
<code>convolve</code>
<code>copy</code>
<code>cross_correlate</code>
<code>extract_features</code>
<code>frame_number_to_time</code>
<code>get_c0_value_in_frame</code>

continues on next page

Table 7 – continued from previous page

<code>get_end_time</code>	
<code>get_frame</code>	
<code>get_frame_number_from_time</code>	
<code>get_number_of_coefficients</code>	
<code>get_number_of_frames</code>	
<code>get_start_time</code>	
<code>get_time_from_frame_number</code>	
<code>get_time_step</code>	
<code>get_total_duration</code>	
<code>get_value_in_frame</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_times_by</code>	
<code>scale_times_to</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_times_by</code>	
<code>shift_times_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>t_bins</code>	
<code>t_grid</code>	

continues on next page

Table 7 – continued from previous page

<i>time_to_frame_number</i>
<i>to_array</i>
<i>to_matrix</i>
<i>to_matrix_features</i>
<i>to_sound</i>
<i>ts</i>
<i>x_bins</i>
<i>x_grid</i>
<i>xs</i>

Attributes

<code>centre_time</code>
<code>class_name</code>
<code>dt</code>
<code>duration</code>
<code>dx</code>
<code>end_time</code>
<code>fmax</code>
<code>fmin</code>
<code>full_name</code>
<code>max_n_coefficients</code>
<code>n_frames</code>
<code>name</code>
<code>nt</code>
<code>nx</code>
<code>start_time</code>
<code>t1</code>
<code>time_range</code>
<code>time_step</code>
<code>tmax</code>
<code>tmin</code>
<code>total_duration</code>
<code>trange</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>

```

class FileFormat
    Bases: pybind11_object
    __eq__(self: object, other: object) → bool
    __hash__(self: object) → int
    __index__(self: parselmouth.Data.FileFormat) → int
    __init__(self: parselmouth.Data.FileFormat, value: int) → None
    __init__(self: parselmouth.Data.FileFormat, arg0: str) → None
    __int__(self: parselmouth.Data.FileFormat) → int
    __ne__(self: object, other: object) → bool
    __new__(**kwargs)
    __repr__(self: object) → str
    __str__()
    name(self: handle) -> str
    BINARY = <FileFormat.BINARY: 2>
    SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
    TEXT = <FileFormat.TEXT: 0>
    property name
    property value

class Frame
    Bases: pybind11_object
    __getitem__(self: parselmouth.CC.Frame, i: int) → float
    __init__(*args, **kwargs)
    __len__(self: parselmouth.CC.Frame) → int
    __new__(**kwargs)
    __setitem__(self: parselmouth.CC.Frame, i: int, value: float) → None
    to_array(self: parselmouth.CC.Frame) → numpy.ndarray[numpy.float64]
    property c
    property c0
    __copy__(self: parselmouth.Data) → parselmouth.Data
    __deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
    __eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
    __getitem__(self: parselmouth.CC, i: int) → parselmouth.CC.Frame
    __getitem__(self: parselmouth.CC, ij: Tuple[int, int]) → float
    
```

```

__init__(*args, **kwargs)

__iter__(self: parselmouth.CC) → Iterator

__len__(self: parselmouth.Sampled) → int

__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool

__new__(**kwargs)

__setitem__(self: parselmouth.CC, ij: Tuple[int, int], value: float) → None

__str__(self: parselmouth.Thing) → str

convolve(self: parselmouth.MFCC, other: parselmouth.MFCC, scaling: parselmouth.AmplitudeScaling =
    <AmplitudeScaling.PEAK_0_99: 4>, signal_outside_time_domain:
    parselmouth.SignalOutsideTimeDomain = <SignalOutsideTimeDomain.ZERO: 1>) →
    parselmouth.Sound

copy(self: parselmouth.Data) → parselmouth.Data

cross_correlate(self: parselmouth.MFCC, other: parselmouth.MFCC, scaling:
    parselmouth.AmplitudeScaling = <AmplitudeScaling.PEAK_0_99: 4>,
    signal_outside_time_domain: parselmouth.SignalOutsideTimeDomain =
    <SignalOutsideTimeDomain.ZERO: 1>) → parselmouth.Sound

extract_features(self: parselmouth.MFCC, window_length: Positive[float] = 0.025, include_energy: bool
    = False) → parselmouth.Matrix

frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_c0_value_in_frame(self: parselmouth.CC, frame_number: Positive[int]) → float

get_end_time(self: parselmouth.Function) → float

get_frame(self: parselmouth.CC, frame_number: Positive[int]) → parselmouth.CC.Frame

get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float

get_number_of_coefficients(self: parselmouth.CC, frame_number: Positive[int]) → int

get_number_of_frames(self: parselmouth.Sampled) → int

get_start_time(self: parselmouth.Function) → float

get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_time_step(self: parselmouth.Sampled) → float

get_total_duration(self: parselmouth.Function) → float

get_value_in_frame(self: parselmouth.CC, frame_number: Positive[int], index: Positive[int]) → float

info(self: parselmouth.Thing) → str

static read(file_path: str) → parselmouth.Data
    Read a file into a parselmouth.Data object.

```

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

save(*self*: *parselmouth.Data*, *file_path*: *str*, *format*: *parselmouth.Data.FileFormat* = <*FileFormat.TEXT*: 0>) → *None*

save_as_binary_file(*self*: *parselmouth.Data*, *file_path*: *str*) → *None*

save_as_short_text_file(*self*: *parselmouth.Data*, *file_path*: *str*) → *None*

save_as_text_file(*self*: *parselmouth.Data*, *file_path*: *str*) → *None*

scale_times_by(*self*: *parselmouth.Function*, *scale*: *Positive[float]*) → *None*

scale_times_to(*self*: *parselmouth.Function*, *new_start_time*: *float*, *new_end_time*: *float*) → *None*

scale_x_by(*self*: *parselmouth.Function*, *scale*: *Positive[float]*) → *None*

scale_x_to(*self*: *parselmouth.Function*, *new_xmin*: *float*, *new_xmax*: *float*) → *None*

shift_times_by(*self*: *parselmouth.Function*, *seconds*: *float*) → *None*

shift_times_to(*self*: *parselmouth.Function*, *time*: *float*, *new_time*: *float*) → *None*

shift_times_to(*self*: *parselmouth.Function*, *time*: *str*, *new_time*: *float*) → *None*

shift_x_by(*self*: *parselmouth.Function*, *shift*: *float*) → *None*

shift_x_to(*self*: *parselmouth.Function*, *x*: *float*, *new_x*: *float*) → *None*

t_bins(*self*: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

t_grid(*self*: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

time_to_frame_number(*self*: *parselmouth.Sampled*, *time*: *float*) → *float*

to_array(*self*: *parselmouth.CC*) → *numpy.ndarray[numpy.float64]*

to_matrix(*self*: *parselmouth.CC*) → *parselmouth.Matrix*

to_matrix_features(*self*: *parselmouth.MFCC*, *window_length*: *Positive[float]* = 0.025, *include_energy*: *bool* = *False*) → *parselmouth.Matrix*

to_sound(*self*: *parselmouth.MFCC*) → *parselmouth.Sound*

ts(*self*: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

x_bins(*self*: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

x_grid(*self*: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

xs(*self*: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

__hash__ = *None*

property centre_time
property class_name
property dt
property duration
property dx
property end_time
property fmax
property fmin
property full_name
property max_n_coefficients
property n_frames
property name
property nt
property nx
property start_time
property t1
property time_range
property time_step
property tmax
property tmin
property total_duration
property trange
property x1
property xmax
property xmin
property xrange

3.1.12 parselmouth.Matrix

class `parselmouth.Matrix`

Bases: *SampledXY*

Methods

<code>__init__</code>	
<code>as_array</code>	
<code>at_xy</code>	
<code>copy</code>	
<code>formula</code>	
<code>get_column_distance</code>	
<code>get_highest_x</code>	
<code>get_highest_y</code>	
<code>get_lowest_x</code>	
<code>get_lowest_y</code>	
<code>get_maximum</code>	
<code>get_minimum</code>	
<code>get_number_of_columns</code>	
<code>get_number_of_rows</code>	
<code>get_row_distance</code>	
<code>get_sum</code>	
<code>get_value_at_xy</code>	
<code>get_value_in_cell</code>	
<code>get_x_of_column</code>	
<code>get_y_of_row</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i><code>parselmouth.Data</code></i> object.

continues on next page

Table 8 – continued from previous page

<code>save</code>
<code>save_as_binary_file</code>
<code>save_as_headerless_spreadsheet_file</code>
<code>save_as_matrix_text_file</code>
<code>save_as_short_text_file</code>
<code>save_as_text_file</code>
<code>scale_x_by</code>
<code>scale_x_to</code>
<code>set_value</code>
<code>shift_x_by</code>
<code>shift_x_to</code>
<code>x_bins</code>
<code>x_grid</code>
<code>xs</code>
<code>y_bins</code>
<code>y_grid</code>
<code>ys</code>

Attributes

<code>class_name</code>
<code>dx</code>
<code>dy</code>
<code>full_name</code>
<code>n_columns</code>
<code>n_rows</code>
<code>name</code>
<code>nx</code>
<code>ny</code>
<code>values</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>
<code>y1</code>
<code>ymax</code>
<code>ymin</code>
<code>yrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object)` → `bool`

`__hash__(self: object)` → `int`

`__index__(self: parselmouth.Data.FileFormat)` → `int`

`__init__(self: parselmouth.Data.FileFormat, value: int)` → `None`

`__init__(self: parselmouth.Data.FileFormat, arg0: str)` → `None`

`__int__(self: parselmouth.Data.FileFormat)` → `int`

```

__ne__(self: object, other: object) → bool

__new__(**kwargs)

__repr__(self: object) → str

__str__()
    name(self: handle) -> str

BINARY = <FileFormat.BINARY: 2>

SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>

TEXT = <FileFormat.TEXT: 0>

property name
property value

__copy__(self: parselmouth.Data) → parselmouth.Data

__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data

__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool

__init__(*args, **kwargs)

__len__(self: parselmouth.Sampled) → int

__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool

__new__(**kwargs)

__str__(self: parselmouth.Thing) → str

as_array(self: parselmouth.Matrix) → numpy.ndarray[numpy.float64]

at_xy(self: parselmouth.Matrix, x: float, y: float) → float

copy(self: parselmouth.Data) → parselmouth.Data

formula(self: parselmouth.Matrix, formula: str, from_x: float | None = None, to_x: float | None = None,
        from_y: float | None = None, to_y: float | None = None) → None

formula(self: parselmouth.Matrix, formula: str, x_range: Tuple[float | None, float | None] = (None, None),
        y_range: Tuple[float | None, float | None] = (None, None)) → None

get_column_distance(self: parselmouth.Matrix) → float

get_highest_x(self: parselmouth.Matrix) → float

get_highest_y(self: parselmouth.Matrix) → float

get_lowest_x(self: parselmouth.Matrix) → float

get_lowest_y(self: parselmouth.Matrix) → float

get_maximum(self: parselmouth.Matrix) → float

get_minimum(self: parselmouth.Matrix) → float

```

```

get_number_of_columns(self: parselmouth.Matrix) → int
get_number_of_rows(self: parselmouth.Matrix) → int
get_row_distance(self: parselmouth.Matrix) → float
get_sum(self: parselmouth.Matrix) → float
get_value_at_xy(self: parselmouth.Matrix, x: float, y: float) → float
get_value_in_cell(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int])
    → float
get_x_of_column(self: parselmouth.Matrix, column_number: Positive[int]) → float
get_y_of_row(self: parselmouth.Matrix, row_number: Positive[int]) → float
info(self: parselmouth.Thing) → str
static read(file_path: str) → parselmouth.Data
    Read a file into a parselmouth.Data object.

    Parameters
        file_path (str) – The path of the file on disk to read.

    Returns
        The Praat Data object that was read.

    Return type
        parselmouth.Data

See also:
    Praat: “Read from file...”

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    → None

save_as_binary_file(self: parselmouth.Data, file_path: str) → None

save_as_headerless_spreadsheet_file(self: parselmouth.Matrix, file_path: str) → None

save_as_matrix_text_file(self: parselmouth.Matrix, file_path: str) → None

save_as_short_text_file(self: parselmouth.Data, file_path: str) → None

save_as_text_file(self: parselmouth.Data, file_path: str) → None

scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None

set_value(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int], new_value:
    float) → None

shift_x_by(self: parselmouth.Function, shift: float) → None

shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None

x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

```

```

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
y_bins(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
y_grid(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
ys(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
__hash__ = None
property class_name
property dx
property dy
property full_name
property n_columns
property n_rows
property name
property nx
property ny
property values
property x1
property xmax
property xmin
property xrange
property y1
property ymax
property ymin
property yrange

```

3.1.13 parselmouth.Pitch

```

class parselmouth.Pitch
    Bases: TimeFrameSampled, Sampled

```

Methods

<code>__init__</code>
<code>copy</code>
<code>count_differences</code>
<code>count_voiced_frames</code>
<code>fifth_down</code>
<code>fifth_up</code>
<code>formula</code>
<code>frame_number_to_time</code>
<code>get_end_time</code>
<code>get_frame</code>
<code>get_frame_number_from_time</code>
<code>get_mean_absolute_slope</code>
<code>get_number_of_frames</code>
<code>get_slope_without_octave_jumps</code>
<code>get_start_time</code>
<code>get_time_from_frame_number</code>
<code>get_time_step</code>
<code>get_total_duration</code>
<code>get_value_at_time</code>
<code>get_value_in_frame</code>
<code>info</code>
<code>interpolate</code>
<code>kill_octave_jumps</code>
<code>octave_down</code>
<code>octave_up</code>

continues on next page

Table 9 – continued from previous page

<code>path_finder</code>	
<code>read</code>	Read a file into a <code>parselmouth.Data</code> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_times_by</code>	
<code>scale_times_to</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_times_by</code>	
<code>shift_times_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>smooth</code>	
<code>step</code>	
<code>subtract_linear_fit</code>	
<code>t_bins</code>	
<code>t_grid</code>	
<code>time_to_frame_number</code>	
<code>to_array</code>	
<code>to_matrix</code>	
<code>to_sound_hum</code>	
<code>to_sound_pulses</code>	
<code>to_sound_sine</code>	
<code>ts</code>	

continues on next page

Table 9 – continued from previous page

<i>unvoice</i>
<i>x_bins</i>
<i>x_grid</i>
<i>xs</i>

Attributes

<i>ceiling</i>
<i>centre_time</i>
<i>class_name</i>
<i>dt</i>
<i>duration</i>
<i>dx</i>
<i>end_time</i>
<i>full_name</i>
<i>max_n_candidates</i>
<i>n_frames</i>
<i>name</i>
<i>nt</i>
<i>nx</i>
<i>selected</i>
<i>selected_array</i>
<i>start_time</i>
<i>t1</i>
<i>time_range</i>
<i>time_step</i>
<i>tmax</i>
<i>tmin</i>
<i>total_duration</i>
<i>trange</i>
<i>x1</i>
<i>xmax</i>
<i>xmin</i>
<i>xrange</i>

```

class Candidate
    Bases: pybind11_object
    __init__(*args, **kwargs)
    __new__(**kwargs)
    property frequency
    property strength

class FileFormat
    Bases: pybind11_object
    __eq__(self: object, other: object) → bool
    __hash__(self: object) → int
    __index__(self: parselmouth.Data.FileFormat) → int
    __init__(self: parselmouth.Data.FileFormat, value: int) → None
    __init__(self: parselmouth.Data.FileFormat, arg0: str) → None
    __int__(self: parselmouth.Data.FileFormat) → int
    __ne__(self: object, other: object) → bool
    __new__(**kwargs)
    __repr__(self: object) → str
    __str__()
    name(self: handle) -> str
    BINARY = <FileFormat.BINARY: 2>
    SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
    TEXT = <FileFormat.TEXT: 0>
    property name
    property value

class Frame
    Bases: pybind11_object
    __getitem__(self: parselmouth.Pitch.Frame, i: int) → parselmouth.Pitch.Candidate
    __init__(*args, **kwargs)
    __len__(self: parselmouth.Pitch.Frame) → int
    __new__(**kwargs)
    as_array(self: parselmouth.Pitch.Frame) → numpy.ndarray
    select(self: parselmouth.Pitch.Frame, candidate: parselmouth.Pitch.Candidate) → None
    select(self: parselmouth.Pitch.Frame, i: int) → None
    
```

```

    unvoice(self: parselmouth.Pitch.Frame) → None

    property candidates

    property intensity

    property selected

__copy__(self: parselmouth.Data) → parselmouth.Data
__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
__getitem__(self: parselmouth.Pitch, i: int) → parselmouth.Pitch.Frame
__getitem__(self: parselmouth.Pitch, ij: Tuple[int, int]) → parselmouth.Pitch.Candidate
__init__(*args, **kwargs)
__iter__(self: parselmouth.Pitch) → Iterator
__len__(self: parselmouth.Sampled) → int
__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
__new__(**kwargs)
__str__(self: parselmouth.Thing) → str
copy(self: parselmouth.Data) → parselmouth.Data
count_differences(self: parselmouth.Pitch, other: parselmouth.Pitch) → str
count_voiced_frames(self: parselmouth.Pitch) → int
fifth_down(self: parselmouth.Pitch, from_time: float | None = None, to_time: float | None = None) → None
fifth_up(self: parselmouth.Pitch, from_time: float | None = None, to_time: float | None = None) → None
formula(self: parselmouth.Pitch, formula: str) → None
frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float
get_end_time(self: parselmouth.Function) → float
get_frame(self: parselmouth.Pitch, frame_number: Positive[int]) → parselmouth.Pitch.Frame
get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float
get_mean_absolute_slope(self: parselmouth.Pitch, unit: parselmouth.PitchUnit = <PitchUnit.HERTZ: 0>) → float
get_number_of_frames(self: parselmouth.Sampled) → int
get_slope_without_octave_jumps(self: parselmouth.Pitch) → float
get_start_time(self: parselmouth.Function) → float
get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float

```

get_time_step(*self*: *parselmouth.Sampled*) → float

get_total_duration(*self*: *parselmouth.Function*) → float

get_value_at_time(*self*: *parselmouth.Pitch*, *time*: float, *unit*: *parselmouth.PitchUnit* = <*PitchUnit.HERTZ*: 0>, *interpolation*: *parselmouth.ValueInterpolation* = <*ValueInterpolation.LINEAR*: 1>) → float

get_value_in_frame(*self*: *parselmouth.Pitch*, *frame_number*: int, *unit*: *parselmouth.PitchUnit* = <*PitchUnit.HERTZ*: 0>) → float

info(*self*: *parselmouth.Thing*) → str

interpolate(*self*: *parselmouth.Pitch*) → *parselmouth.Pitch*

kill_octave_jumps(*self*: *parselmouth.Pitch*) → *parselmouth.Pitch*

octave_down(*self*: *parselmouth.Pitch*, *from_time*: float | None = None, *to_time*: float | None = None) → None

octave_up(*self*: *parselmouth.Pitch*, *from_time*: float | None = None, *to_time*: float | None = None) → None

path_finder(*self*: *parselmouth.Pitch*, *silence_threshold*: float = 0.03, *voicing_threshold*: float = 0.45, *octave_cost*: float = 0.01, *octave_jump_cost*: float = 0.35, *voiced_unvoiced_cost*: float = 0.14, *ceiling*: Positive[float] = 600.0, *pull_formants*: bool = False) → None

static read(*file_path*: str) → *parselmouth.Data*

Read a file into a *parselmouth.Data* object.

Parameters

file_path (str) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

save(*self*: *parselmouth.Data*, *file_path*: str, *format*: *parselmouth.Data.FileFormat* = <*FileFormat.TEXT*: 0>) → None

save_as_binary_file(*self*: *parselmouth.Data*, *file_path*: str) → None

save_as_short_text_file(*self*: *parselmouth.Data*, *file_path*: str) → None

save_as_text_file(*self*: *parselmouth.Data*, *file_path*: str) → None

scale_times_by(*self*: *parselmouth.Function*, *scale*: Positive[float]) → None

scale_times_to(*self*: *parselmouth.Function*, *new_start_time*: float, *new_end_time*: float) → None

scale_x_by(*self*: *parselmouth.Function*, *scale*: Positive[float]) → None

scale_x_to(*self*: *parselmouth.Function*, *new_xmin*: float, *new_xmax*: float) → None

```

shift_times_by(self: parselmouth.Function, seconds: float) → None

shift_times_to(self: parselmouth.Function, time: float, new_time: float) → None
shift_times_to(self: parselmouth.Function, time: str, new_time: float) → None

shift_x_by(self: parselmouth.Function, shift: float) → None

shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None

smooth(self: parselmouth.Pitch, bandwidth: Positive[float] = 10.0) → parselmouth.Pitch

step(self: parselmouth.Pitch, step: float, precision: Positive[float] = 0.1, from_time: float | None = None,
      to_time: float | None = None) → None

subtract_linear_fit(self: parselmouth.Pitch, unit: parselmouth.PitchUnit = <PitchUnit.HERTZ: 0>) →
    parselmouth.Pitch

t_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

t_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

time_to_frame_number(self: parselmouth.Sampled, time: float) → float

to_array(self: parselmouth.Pitch) → numpy.ndarray[parselmouth.Pitch.Candidate]

to_matrix(self: parselmouth.Pitch) → parselmouth.Matrix

to_sound_hum(self: parselmouth.Pitch, from_time: float | None = None, to_time: float | None = None) →
    parselmouth.Sound

to_sound_pulses(self: parselmouth.Pitch, from_time: float | None = None, to_time: float | None = None)
    → parselmouth.Sound

to_sound_sine(self: parselmouth.Pitch, from_time: float | None = None, to_time: float | None = None,
    sampling_frequency: Positive[float] = 44100.0, round_to_nearest_zero_crossing: float =
    True) → parselmouth.Sound

ts(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

unvoice(self: parselmouth.Pitch, from_time: float | None = None, to_time: float | None = None) → None

x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

__hash__ = None

property ceiling

property centre_time

property class_name

property dt

property duration

```

```

property dx
property end_time
property full_name
property max_n_candidates
property n_frames
property name
property nt
property nx
property selected
property selected_array
property start_time
property t1
property time_range
property time_step
property tmax
property tmin
property total_duration
property trange
property x1
property xmax
property xmin
property xrange

```

3.1.14 parselmouth.PitchUnit

```
class parselmouth.PitchUnit
```

Bases: pybind11_object

Methods

```
__init__
```

Attributes

<code>ERB</code>
<code>HERTZ</code>
<code>HERTZ_LOGARITHMIC</code>
<code>LOG_HERTZ</code>
<code>MEL</code>
<code>SEMITONES_1</code>
<code>SEMITONES_100</code>
<code>SEMITONES_200</code>
<code>SEMITONES_440</code>
<code>name</code>
<code>value</code>

```

__eq__(self: object, other: object) → bool
__hash__(self: object) → int
__index__(self: parselmouth.PitchUnit) → int
__init__(self: parselmouth.PitchUnit, value: int) → None
__init__(self: parselmouth.PitchUnit, arg0: str) → None
__int__(self: parselmouth.PitchUnit) → int
__ne__(self: object, other: object) → bool
__new__(**kwargs)
__repr__(self: object) → str
__str__()
    name(self: handle) -> str
ERB = <PitchUnit.ERB: 8>
HERTZ = <PitchUnit.HERTZ: 0>
HERTZ_LOGARITHMIC = <PitchUnit.HERTZ_LOGARITHMIC: 1>
LOG_HERTZ = <PitchUnit.LOG_HERTZ: 3>
MEL = <PitchUnit.MEL: 2>

```

```
SEMITONES_1 = <PitchUnit.SEMITONES_1: 4>
SEMITONES_100 = <PitchUnit.SEMITONES_100: 5>
SEMITONES_200 = <PitchUnit.SEMITONES_200: 6>
SEMITONES_440 = <PitchUnit.SEMITONES_440: 7>
```

property name

property value

3.1.15 parselmouth.Sampled

class `parselmouth.Sampled`

Bases: *Function*

Methods

<code>__init__</code>	
<code>copy</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>x_bins</code>	
<code>x_grid</code>	
<code>xs</code>	

Attributes

<i>class_name</i>
<i>dx</i>
<i>full_name</i>
<i>name</i>
<i>nx</i>
<i>x1</i>
<i>xmax</i>
<i>xmin</i>
<i>xrange</i>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object)` → bool

`__hash__(self: object)` → int

`__index__(self: parselmouth.Data.FileFormat)` → int

`__init__(self: parselmouth.Data.FileFormat, value: int)` → None

`__init__(self: parselmouth.Data.FileFormat, arg0: str)` → None

`__int__(self: parselmouth.Data.FileFormat)` → int

`__ne__(self: object, other: object)` → bool

`__new__(**kwargs)`

`__repr__(self: object)` → str

`__str__()`

`name(self: handle)` → str

`BINARY = <FileFormat.BINARY: 2>`

`SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>`

`TEXT = <FileFormat.TEXT: 0>`

property `name`

property `value`

`__copy__(self: parselmouth.Data)` → `parselmouth.Data`

```
__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
```

```
__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
```

```
__init__(*args, **kwargs)
```

```
__len__(self: parselmouth.Sampled) → int
```

```
__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
```

```
__new__(**kwargs)
```

```
__str__(self: parselmouth.Thing) → str
```

```
copy(self: parselmouth.Data) → parselmouth.Data
```

```
info(self: parselmouth.Thing) → str
```

```
static read(file_path: str) → parselmouth.Data
```

Read a file into a *parselmouth.Data* object.

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

```
save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
→ None
```

```
save_as_binary_file(self: parselmouth.Data, file_path: str) → None
```

```
save_as_short_text_file(self: parselmouth.Data, file_path: str) → None
```

```
save_as_text_file(self: parselmouth.Data, file_path: str) → None
```

```
scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None
```

```
scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None
```

```
shift_x_by(self: parselmouth.Function, shift: float) → None
```

```
shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None
```

```
x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
```

```
x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
```

```
xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
```

```
__hash__ = None
```

```
property class_name
```

`property dx`
`property full_name`
`property name`
`property nx`
`property x1`
`property xmax`
`property xmin`
`property xrange`

3.1.16 `parselmouth.SampledXY`

`class parselmouth.SampledXY`
Bases: *Sampled*

Methods

<code>__init__</code>	
<code>copy</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i><code>parselmouth.Data</code></i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>x_bins</code>	
<code>x_grid</code>	
<code>xs</code>	
<code>y_bins</code>	
<code>y_grid</code>	
<code>ys</code>	

Attributes

<code>class_name</code>
<code>dx</code>
<code>dy</code>
<code>full_name</code>
<code>name</code>
<code>nx</code>
<code>ny</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>
<code>y1</code>
<code>ymax</code>
<code>ymin</code>
<code>yrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object)` → `bool`

`__hash__(self: object)` → `int`

`__index__(self: parselmouth.Data.FileFormat)` → `int`

`__init__(self: parselmouth.Data.FileFormat, value: int)` → `None`

`__init__(self: parselmouth.Data.FileFormat, arg0: str)` → `None`

`__int__(self: parselmouth.Data.FileFormat)` → `int`

`__ne__(self: object, other: object)` → `bool`

`__new__(**kwargs)`

`__repr__(self: object)` → `str`

```

__str__()
    name(self: handle) -> str

BINARY = <FileFormat.BINARY: 2>

SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>

TEXT = <FileFormat.TEXT: 0>

property name

property value

__copy__(self: parselmouth.Data) -> parselmouth.Data

__deepcopy__(self: parselmouth.Data, memo: dict) -> parselmouth.Data

__eq__(self: parselmouth.Data, other: parselmouth.Data) -> bool

__init__(*args, **kwargs)

__len__(self: parselmouth.Sampled) -> int

__ne__(self: parselmouth.Data, other: parselmouth.Data) -> bool

__new__(**kwargs)

__str__(self: parselmouth.Thing) -> str

copy(self: parselmouth.Data) -> parselmouth.Data

info(self: parselmouth.Thing) -> str

static read(file_path: str) -> parselmouth.Data
    Read a file into a parselmouth.Data object.

    Parameters
        file_path (str) – The path of the file on disk to read.

    Returns
        The Praat Data object that was read.

    Return type
        parselmouth.Data

    See also:
        Praat: “Read from file...”

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    -> None

save_as_binary_file(self: parselmouth.Data, file_path: str) -> None

save_as_short_text_file(self: parselmouth.Data, file_path: str) -> None

save_as_text_file(self: parselmouth.Data, file_path: str) -> None

scale_x_by(self: parselmouth.Function, scale: Positive[float]) -> None

scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) -> None

```

```

shift_x_by(self: parselmouth.Function, shift: float) → None
shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None
x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
y_bins(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
y_grid(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
ys(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
__hash__ = None
property class_name
property dx
property dy
property full_name
property name
property nx
property ny
property x1
property xmax
property xmin
property xrange
property y1
property ymax
property ymin
property yrange

```

3.1.17 parselmouth.SignalOutsideTimeDomain

```

class parselmouth.SignalOutsideTimeDomain
    Bases: pybind11_object

```

Methods

`__init__`

Attributes

`SIMILAR`

`ZERO`

`name`

`value`

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.SignalOutsideTimeDomain) → int`

`__init__(self: parselmouth.SignalOutsideTimeDomain, value: int) → None`

`__init__(self: parselmouth.SignalOutsideTimeDomain, arg0: str) → None`

`__int__(self: parselmouth.SignalOutsideTimeDomain) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

`__repr__(self: object) → str`

`__str__()`

`name(self: handle) -> str`

`SIMILAR = <SignalOutsideTimeDomain.SIMILAR: 2>`

`ZERO = <SignalOutsideTimeDomain.ZERO: 1>`

property name

property value

3.1.18 parselmouth.Sound

class `parselmouth.Sound`

Bases: *TimeFrameSampled*, *Vector*

A fragment of audio, represented by one or multiple channels of floating point values between -1 and 1, sampled at a fixed sampling frequency.

Corresponds to a *Praat*: “*Sound*” object.

See also:

Praat: “*Sound*”, *Praat*: “*Sound files*”, *Praat*: “*Sound files 1. General structure*”

Methods

<code>__init__</code>	Create a new <i>Sound</i> object.
<code>add</code>	
<code>as_array</code>	
<code>at_xy</code>	
<code>autocorrelate</code>	
<code>combine_to_stereo</code>	
<code>concatenate</code>	
<code>convert_to_mono</code>	
<code>convert_to_stereo</code>	
<code>convolve</code>	
<code>copy</code>	
<code>cross_correlate</code>	
<code>de_emphasize</code>	
<code>deepen_band_modulation</code>	
<code>divide</code>	
<code>extract_all_channels</code>	
<code>extract_channel</code>	
<code>extract_left_channel</code>	
<code>extract_part</code>	

continues on next page

Table 10 – continued from previous page

<i>extract_part_for_overlap</i>
<i>extract_right_channel</i>
<i>formula</i>
<i>frame_number_to_time</i>
<i>get_column_distance</i>
<i>get_end_time</i>
<i>get_energy</i>
<i>get_energy_in_air</i>
<i>get_frame_number_from_time</i>
<i>get_highest_x</i>
<i>get_highest_y</i>
<i>get_index_from_time</i>
<i>get_intensity</i>
<i>get_lowest_x</i>
<i>get_lowest_y</i>
<i>get_maximum</i>
<i>get_minimum</i>
<i>get_nearest_zero_crossing</i>
<i>get_number_of_channels</i>
<i>get_number_of_columns</i>
<i>get_number_of_frames</i>
<i>get_number_of_rows</i>
<i>get_number_of_samples</i>
<i>get_power</i>
<i>get_power_in_air</i>
<i>get_rms</i>

continues on next page

Table 10 – continued from previous page

<code>get_root_mean_square</code>	
<code>get_row_distance</code>	
<code>get_sampling_frequency</code>	
<code>get_sampling_period</code>	
<code>get_start_time</code>	
<code>get_sum</code>	
<code>get_time_from_frame_number</code>	
<code>get_time_from_index</code>	
<code>get_time_step</code>	
<code>get_total_duration</code>	
<code>get_value</code>	
<code>get_value_at_xy</code>	
<code>get_value_in_cell</code>	
<code>get_x_of_column</code>	
<code>get_y_of_row</code>	
<code>info</code>	
<code>lengthen</code>	
<code>multiply</code>	
<code>multiply_by_window</code>	
<code>override_sampling_frequency</code>	
<code>pre_emphasize</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>resample</code>	
<code>reverse</code>	
<code>save</code>	Save a <i>Sound</i> object to an audio file on disk.
<code>save_as_binary_file</code>	
<code>save_as_headerless_spreadsheet_file</code>	

continues on next page

Table 10 – continued from previous page

<code>save_as_matrix_text_file</code>
<code>save_as_short_text_file</code>
<code>save_as_text_file</code>
<code>scale</code>
<code>scale_intensity</code>
<code>scale_peak</code>
<code>scale_times_by</code>
<code>scale_times_to</code>
<code>scale_x_by</code>
<code>scale_x_to</code>
<code>set_to_zero</code>
<code>set_value</code>
<code>shift_times_by</code>
<code>shift_times_to</code>
<code>shift_x_by</code>
<code>shift_x_to</code>
<code>subtract</code>
<code>subtract_mean</code>
<code>t_bins</code>
<code>t_grid</code>
<code>time_to_frame_number</code>
<code>to_formant_burg</code>
<code>to_harmonicity</code>
<code>to_harmonicity_ac</code>
<code>to_harmonicity_cc</code>
<code>to_harmonicity_gne</code>

continues on next page

Table 10 – continued from previous page

<code>to_intensity</code>
<code>to_mfcc</code>
<code>to_pitch</code>
<code>to_pitch_ac</code>
<code>to_pitch_cc</code>
<code>to_pitch_shs</code>
<code>to_pitch_spinet</code>
<code>to_spectrogram</code>
<code>to_spectrum</code>
<code>ts</code>
<code>x_bins</code>
<code>x_grid</code>
<code>xs</code>
<code>y_bins</code>
<code>y_grid</code>
<code>ys</code>

Attributes

<code>centre_time</code>
<code>class_name</code>
<code>dt</code>
<code>duration</code>
<code>dx</code>
<code>dy</code>
<code>end_time</code>

continues on next page

Table 11 – continued from previous page

<i>full_name</i>
<i>n_channels</i>
<i>n_columns</i>
<i>n_frames</i>
<i>n_rows</i>
<i>n_samples</i>
<i>name</i>
<i>nt</i>
<i>nx</i>
<i>ny</i>
<i>sampling_frequency</i>
<i>sampling_period</i>
<i>start_time</i>
<i>t1</i>
<i>time_range</i>
<i>time_step</i>
<i>tmax</i>
<i>tmin</i>
<i>total_duration</i>
<i>trange</i>
<i>values</i>
<i>x1</i>
<i>xmax</i>
<i>xmin</i>
<i>xrange</i>
<i>y1</i>

continues on next page

Table 11 – continued from previous page

<i>ymax</i>
<i>ymin</i>
<i>yrange</i>

class FileFormat

Bases: pybind11_object

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.Data.FileFormat) → int`

`__init__(self: parselmouth.Data.FileFormat, value: int) → None`

`__init__(self: parselmouth.Data.FileFormat, arg0: str) → None`

`__int__(self: parselmouth.Data.FileFormat) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

`__repr__(self: object) → str`

`__str__()`

name(self: handle) -> str

`BINARY = <FileFormat.BINARY: 2>`

`SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>`

`TEXT = <FileFormat.TEXT: 0>`

property name

property value

class ToHarmonicityMethod

Bases: pybind11_object

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.Sound.ToHarmonicityMethod) → int`

`__init__(self: parselmouth.Sound.ToHarmonicityMethod, value: int) → None`

`__init__(self: parselmouth.Sound.ToHarmonicityMethod, arg0: str) → None`

`__int__(self: parselmouth.Sound.ToHarmonicityMethod) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

```

    __repr__(self: object) → str
    __str__()
        name(self: handle) -> str
    AC = <ToHarmonicityMethod.AC: 1>
    CC = <ToHarmonicityMethod.CC: 0>
    GNE = <ToHarmonicityMethod.GNE: 2>
    property name
    property value
class ToPitchMethod
    Bases: pybind11_object
    __eq__(self: object, other: object) → bool
    __hash__(self: object) → int
    __index__(self: parselmouth.Sound.ToPitchMethod) → int
    __init__(self: parselmouth.Sound.ToPitchMethod, value: int) → None
    __init__(self: parselmouth.Sound.ToPitchMethod, arg0: str) → None
    __int__(self: parselmouth.Sound.ToPitchMethod) → int
    __ne__(self: object, other: object) → bool
    __new__(**kwargs)
    __repr__(self: object) → str
    __str__()
        name(self: handle) -> str
    AC = <ToPitchMethod.AC: 0>
    CC = <ToPitchMethod.CC: 1>
    SHS = <ToPitchMethod.SHS: 3>
    SPINET = <ToPitchMethod.SPINET: 2>
    property name
    property value
    __add__(self: parselmouth.Vector, number: float) → parselmouth.Vector
    __copy__(self: parselmouth.Data) → parselmouth.Data
    __deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
    __eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
    __iadd__(self: parselmouth.Vector, number: float) → parselmouth.Vector

```

`__imul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector`

`__init__(self: parselmouth.Sound, other: parselmouth.Sound) → None`

`__init__(self: parselmouth.Sound, values: numpy.ndarray[numpy.float64], sampling_frequency: Positive[float] = 44100.0, start_time: float = 0.0) → None`

`__init__(self: parselmouth.Sound, file_path: str) → None`

Create a new *Sound* object.

The new object can be created: - as a copy of an existing *Sound* object, - from an array of samples and a sampling frequency, or - by reading an audio file from disk.

Parameters

- **other** (*Sound*) – The *Sound* object to copy.
- **samples** (*array_like[float]*) – The samples of the new *Sound* object.
- **sampling_frequency** (*float*, *optional*) – The sampling frequency of the new *Sound* object (default: 44100).
- **start_time** (*float*, *optional*) – The start time (*xmin*) of the new *Sound* object (default: 0).
- **file_name** (*str*, *optional*) – The file name of the audio file to load.
- **file_path** (*str*) – The file path of an audio file to load from disk.

See also:

Praat: “Sound files 2. File types”, Praat: “Sound files 3. Files that Praat can read”

`__isub__(self: parselmouth.Vector, number: float) → parselmouth.Vector`

`__itruediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector`

`__len__(self: parselmouth.Sampled) → int`

`__mul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector`

`__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool`

`__new__(**kwargs)`

`__radd__(self: parselmouth.Vector, number: float) → parselmouth.Vector`

`__rmul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector`

`__str__(self: parselmouth.Thing) → str`

`__sub__(self: parselmouth.Vector, number: float) → parselmouth.Vector`

`__truediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector`

`add(self: parselmouth.Vector, number: float) → None`

`as_array(self: parselmouth.Matrix) → numpy.ndarray[numpy.float64]`

`at_xy(self: parselmouth.Matrix, x: float, y: float) → float`

```

autocorrelate(self: parselmouth.Sound, scaling: parselmouth.AmplitudeScaling =
    <AmplitudeScaling.PEAK_0_99: 4>, signal_outside_time_domain:
    parselmouth.SignalOutsideTimeDomain = <SignalOutsideTimeDomain.ZERO: 1>) →
    parselmouth.Sound

static combine_to_stereo(sounds: List[parselmouth.Sound]) → parselmouth.Sound

static concatenate(sounds: List[parselmouth.Sound], overlap: NonNegative[float] = 0.0) →
    parselmouth.Sound

convert_to_mono(self: parselmouth.Sound) → parselmouth.Sound

convert_to_stereo(self: parselmouth.Sound) → parselmouth.Sound

convolve(self: parselmouth.Sound, other: parselmouth.Sound, scaling: parselmouth.AmplitudeScaling =
    <AmplitudeScaling.PEAK_0_99: 4>, signal_outside_time_domain:
    parselmouth.SignalOutsideTimeDomain = <SignalOutsideTimeDomain.ZERO: 1>) →
    parselmouth.Sound

copy(self: parselmouth.Data) → parselmouth.Data

cross_correlate(self: parselmouth.Sound, other: parselmouth.Sound, scaling:
    parselmouth.AmplitudeScaling = <AmplitudeScaling.PEAK_0_99: 4>,
    signal_outside_time_domain: parselmouth.SignalOutsideTimeDomain =
    <SignalOutsideTimeDomain.ZERO: 1>) → parselmouth.Sound

de_emphasize(self: parselmouth.Sound, from_frequency: float = 50.0, normalize: bool = True) → None

deepen_band_modulation(self: parselmouth.Sound, enhancement: Positive[float] = 20.0, from_frequency:
    Positive[float] = 300.0, to_frequency: Positive[float] = 8000.0,
    slow_modulation: Positive[float] = 3.0, fast_modulation: Positive[float] = 30.0,
    band_smoothing: Positive[float] = 100.0) → parselmouth.Sound

divide(self: parselmouth.Vector, factor: float) → None

extract_all_channels(self: parselmouth.Sound) → List[parselmouth.Sound]

extract_channel(self: parselmouth.Sound, channel: int) → parselmouth.Sound

extract_channel(self: parselmouth.Sound, arg0: str) → parselmouth.Sound

extract_left_channel(self: parselmouth.Sound) → parselmouth.Sound

extract_part(self: parselmouth.Sound, from_time: Optional[float] = None, to_time: Optional[float] =
    None, window_shape: parselmouth.WindowShape = <WindowShape.RECTANGULAR: 0>,
    relative_width: Positive[float] = 1.0, preserve_times: bool = False) → parselmouth.Sound

extract_part_for_overlap(self: parselmouth.Sound, from_time: Optional[float] = None, to_time:
    Optional[float] = None, overlap: Positive[float]) → parselmouth.Sound

extract_right_channel(self: parselmouth.Sound) → parselmouth.Sound

formula(self: parselmouth.Matrix, formula: str, from_x: float | None = None, to_x: float | None = None,
    from_y: float | None = None, to_y: float | None = None) → None

formula(self: parselmouth.Matrix, formula: str, x_range: Tuple[float | None, float | None] = (None, None),
    y_range: Tuple[float | None, float | None] = (None, None)) → None

frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float
    
```

```

get_column_distance(self: parselmouth.Matrix) → float
get_end_time(self: parselmouth.Function) → float
get_energy(self: parselmouth.Sound, from_time: float | None = None, to_time: float | None = None) → float
get_energy_in_air(self: parselmouth.Sound) → float
get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float
get_highest_x(self: parselmouth.Matrix) → float
get_highest_y(self: parselmouth.Matrix) → float
get_index_from_time(self: parselmouth.Sound, time: float) → float
get_intensity(self: parselmouth.Sound) → float
get_lowest_x(self: parselmouth.Matrix) → float
get_lowest_y(self: parselmouth.Matrix) → float
get_maximum(self: parselmouth.Matrix) → float
get_minimum(self: parselmouth.Matrix) → float
get_nearest_zero_crossing(self: parselmouth.Sound, time: float, channel: int = 1) → float
get_number_of_channels(self: parselmouth.Sound) → int
get_number_of_columns(self: parselmouth.Matrix) → int
get_number_of_frames(self: parselmouth.Sampled) → int
get_number_of_rows(self: parselmouth.Matrix) → int
get_number_of_samples(self: parselmouth.Sound) → int
get_power(self: parselmouth.Sound, from_time: float | None = None, to_time: float | None = None) → float
get_power_in_air(self: parselmouth.Sound) → float
get_rms(self: parselmouth.Sound, from_time: float | None = None, to_time: float | None = None) → float
get_root_mean_square(self: parselmouth.Sound, from_time: float | None = None, to_time: float | None =
    None) → float
get_row_distance(self: parselmouth.Matrix) → float
get_sampling_frequency(self: parselmouth.Sound) → float
get_sampling_period(self: parselmouth.Sound) → float
get_start_time(self: parselmouth.Function) → float
get_sum(self: parselmouth.Matrix) → float
get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float
get_time_from_index(self: parselmouth.Sound, sample: int) → float

```

get_time_step(*self*: [parselmouth.Sampled](#)) → float
get_total_duration(*self*: [parselmouth.Function](#)) → float
get_value(*self*: [parselmouth.Vector](#), *x*: float, *channel*: *Optional[int] = None*, *interpolation*: [parselmouth.ValueInterpolation](#) = <[ValueInterpolation.CUBIC](#): 2>) → float
get_value_at_xy(*self*: [parselmouth.Matrix](#), *x*: float, *y*: float) → float
get_value_in_cell(*self*: [parselmouth.Matrix](#), *row_number*: *Positive[int]*, *column_number*: *Positive[int]*) → float
get_x_of_column(*self*: [parselmouth.Matrix](#), *column_number*: *Positive[int]*) → float
get_y_of_row(*self*: [parselmouth.Matrix](#), *row_number*: *Positive[int]*) → float
info(*self*: [parselmouth.Thing](#)) → str
lengthen(*self*: [parselmouth.Sound](#), *minimum_pitch*: *Positive[float] = 75.0*, *maximum_pitch*: *Positive[float] = 600.0*, *factor*: *Positive[float]*) → [parselmouth.Sound](#)
multiply(*self*: [parselmouth.Vector](#), *factor*: float) → None
multiply_by_window(*self*: [parselmouth.Sound](#), *window_shape*: [parselmouth.WindowShape](#)) → None
override_sampling_frequency(*self*: [parselmouth.Sound](#), *new_frequency*: *Positive[float]*) → None
pre_emphasize(*self*: [parselmouth.Sound](#), *from_frequency*: float = 50.0, *normalize*: bool = True) → None
static read(*file_path*: str) → [parselmouth.Data](#)

Read a file into a [parselmouth.Data](#) object.

Parameters

file_path (str) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

[parselmouth.Data](#)

See also:

Praat: “Read from file...”

resample(*self*: [parselmouth.Sound](#), *new_frequency*: float, *precision*: int = 50) → [parselmouth.Sound](#)
reverse(*self*: [parselmouth.Sound](#), *from_time*: float | None = None, *to_time*: float | None = None) → None
save(*self*: [parselmouth.Sound](#), *file_path*: str, *format*: [parselmouth.SoundFileFormat](#)) → None
 Save a [Sound](#) object to an audio file on disk.

Parameters

- **file_path** (str) – The file path of the audio file to save to disk.
- **file_format** ([SoundFileFormat](#)) – The audio file format to write to. This can either be a [SoundFileFormat](#) value (e.g., [SoundFileFormat.WAV](#)), or the string representation of the value (e.g., “WAV”).

See also:

Praat: “Sound files 2. File types”, *Praat*: “Sound files 4. Files that Praat can write”

```

save_as_binary_file(self: parselmouth.Data, file_path: str) → None
save_as_headerless_spreadsheet_file(self: parselmouth.Matrix, file_path: str) → None
save_as_matrix_text_file(self: parselmouth.Matrix, file_path: str) → None
save_as_short_text_file(self: parselmouth.Data, file_path: str) → None
save_as_text_file(self: parselmouth.Data, file_path: str) → None
scale(self: parselmouth.Vector, scale: Positive[float]) → None
scale_intensity(self: parselmouth.Sound, new_average_intensity: float) → None
scale_peak(self: parselmouth.Vector, new_peak: Positive[float] = 0.99) → None
scale_times_by(self: parselmouth.Function, scale: Positive[float]) → None
scale_times_to(self: parselmouth.Function, new_start_time: float, new_end_time: float) → None
scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None
scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None
set_to_zero(self: parselmouth.Sound, from_time: float | None = None, to_time: float | None = None,
             round_to_nearest_zero_crossing: bool = True) → None
set_value(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int], new_value:
            float) → None
shift_times_by(self: parselmouth.Function, seconds: float) → None
shift_times_to(self: parselmouth.Function, time: float, new_time: float) → None
shift_times_to(self: parselmouth.Function, time: str, new_time: float) → None
shift_x_by(self: parselmouth.Function, shift: float) → None
shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None
subtract(self: parselmouth.Vector, number: float) → None
subtract_mean(self: parselmouth.Vector) → None
t_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
t_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
time_to_frame_number(self: parselmouth.Sampled, time: float) → float
to_formant_burg(self: parselmouth.Sound, time_step: Positive[float] | None = None,
                 max_number_of_formants: Positive[float] = 5.0, maximum_formant: float = 5500.0,
                 window_length: Positive[float] = 0.025, pre_emphasis_from: Positive[float] = 50.0) →
                 parselmouth.Formant
to_harmonicity(self: parselmouth.Sound, method: parselmouth.Sound.ToHarmonicityMethod =
                 <ToHarmonicityMethod.CC: 0>, *args, **kwargs) → object

```

to_harmonicity_ac(self: `parselmouth.Sound`, time_step: `Positive[float] = 0.01`, minimum_pitch: `Positive[float] = 75.0`, silence_threshold: `float = 0.1`, periods_per_window: `Positive[float] = 1.0`) → `parselmouth.Harmonicity`

to_harmonicity_cc(self: `parselmouth.Sound`, time_step: `Positive[float] = 0.01`, minimum_pitch: `Positive[float] = 75.0`, silence_threshold: `float = 0.1`, periods_per_window: `Positive[float] = 1.0`) → `parselmouth.Harmonicity`

to_harmonicity_gne(self: `parselmouth.Sound`, minimum_frequency: `Positive[float] = 500.0`, maximum_frequency: `Positive[float] = 4500.0`, bandwidth: `Positive[float] = 1000.0`, step: `Positive[float] = 80.0`) → `parselmouth.Matrix`

to_intensity(self: `parselmouth.Sound`, minimum_pitch: `Positive[float] = 100.0`, time_step: `Positive[float]` | `None = None`, subtract_mean: `bool = True`) → `parselmouth.Intensity`

to_mfcc(self: `parselmouth.Sound`, number_of_coefficients: `Positive[int] = 12`, window_length: `Positive[float] = 0.015`, time_step: `Positive[float] = 0.005`, firstFilterFrequency: `Positive[float] = 100.0`, distance_between_filters: `Positive[float] = 100.0`, maximum_frequency: `Positive[float]` | `None = None`) → `parselmouth.MFCC`

to_pitch(self: `parselmouth.Sound`, time_step: `Positive[float]` | `None = None`, pitch_floor: `Positive[float] = 75.0`, pitch_ceiling: `Positive[float] = 600.0`) → `parselmouth.Pitch`

to_pitch(self: `parselmouth.Sound`, method: `parselmouth.Sound.ToPitchMethod`, *args, **kwargs) → object

to_pitch_ac(self: `parselmouth.Sound`, time_step: `Positive[float]` | `None = None`, pitch_floor: `Positive[float] = 75.0`, max_number_of_candidates: `Positive[int] = 15`, very_accurate: `bool = False`, silence_threshold: `float = 0.03`, voicing_threshold: `float = 0.45`, octave_cost: `float = 0.01`, octave_jump_cost: `float = 0.35`, voiced_unvoiced_cost: `float = 0.14`, pitch_ceiling: `Positive[float] = 600.0`) → `parselmouth.Pitch`

to_pitch_cc(self: `parselmouth.Sound`, time_step: `Positive[float]` | `None = None`, pitch_floor: `Positive[float] = 75.0`, max_number_of_candidates: `Positive[int] = 15`, very_accurate: `bool = False`, silence_threshold: `float = 0.03`, voicing_threshold: `float = 0.45`, octave_cost: `float = 0.01`, octave_jump_cost: `float = 0.35`, voiced_unvoiced_cost: `float = 0.14`, pitch_ceiling: `Positive[float] = 600.0`) → `parselmouth.Pitch`

to_pitch_shs(self: `parselmouth.Sound`, time_step: `Positive[float] = 0.01`, minimum_pitch: `Positive[float] = 50.0`, max_number_of_candidates: `Positive[int] = 15`, maximum_frequency_component: `Positive[float] = 1250.0`, max_number_of_subharmonics: `Positive[int] = 15`, compression_factor: `Positive[float] = 0.84`, ceiling: `Positive[float] = 600.0`, number_of_points_per_octave: `Positive[int] = 48`) → `parselmouth.Pitch`

to_pitch_spinet(self: `parselmouth.Sound`, time_step: `Positive[float] = 0.005`, window_length: `Positive[float] = 0.04`, minimum_filter_frequency: `Positive[float] = 70.0`, maximum_filter_frequency: `Positive[float] = 5000.0`, number_of_filters: `Positive[int] = 250`, ceiling: `Positive[float] = 500.0`, max_number_of_candidates: `Positive[int] = 15`) → `parselmouth.Pitch`

to_spectrogram(self: `parselmouth.Sound`, window_length: `Positive[float] = 0.005`, maximum_frequency: `Positive[float] = 5000.0`, time_step: `Positive[float] = 0.002`, frequency_step: `Positive[float] = 20.0`, window_shape: `parselmouth.SpectralAnalysisWindowShape = <SpectralAnalysisWindowShape.GAUSSIAN: 5>`) → `parselmouth.Spectrogram`

to_spectrum(self: `parselmouth.Sound`, fast: `bool = True`) → `parselmouth.Spectrum`

```

ts(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]
y_bins(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
y_grid(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]
ys(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

__hash__ = None
property centre_time
property class_name
property dt
property duration
property dx
property dy
property end_time
property full_name
property n_channels
property n_columns
property n_frames
property n_rows
property n_samples
property name
property nt
property nx
property ny
property sampling_frequency
property sampling_period
property start_time
property t1
property time_range
property time_step

```

```

property tmax
property tmin
property total_duration
property trange
property values
property x1
property xmax
property xmin
property xrange
property y1
property ymax
property ymin
property yrange

```

3.1.19 parselmouth.SoundFileFormat

class `parselmouth.SoundFileFormat`

Bases: `pybind11_object`

Methods

```
__init__
```

Attributes

<i>AIFC</i>
<i>AIFF</i>
<i>FLAC</i>
<i>KAY</i>
<i>NEXT_SUN</i>
<i>NIST</i>
<i>RAW_16_BE</i>
<i>RAW_16_LE</i>
<i>RAW_24_BE</i>
<i>RAW_24_LE</i>
<i>RAW_32_BE</i>
<i>RAW_32_LE</i>
<i>RAW_8_SIGNED</i>
<i>RAW_8_UNSIGNED</i>
<i>SESAM</i>
<i>WAV</i>
<i>WAV_24</i>
<i>WAV_32</i>
<i>name</i>
<i>value</i>

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.SoundFileFormat) → int`

`__init__(self: parselmouth.SoundFileFormat, value: int) → None`

`__init__(self: parselmouth.SoundFileFormat, arg0: str) → None`

`__int__(self: parselmouth.SoundFileFormat) → int`

```

__ne__(self: object, other: object) → bool

__new__(**kwargs)

__repr__(self: object) → str

__str__()
    name(self: handle) -> str

AIFC = <SoundFileFormat.AIFC: 2>

AIFF = <SoundFileFormat.AIFF: 1>

FLAC = <SoundFileFormat.FLAC: 5>

KAY = <SoundFileFormat.KAY: 6>

NEXT_SUN = <SoundFileFormat.NEXT_SUN: 3>

NIST = <SoundFileFormat.NIST: 4>

RAW_16_BE = <SoundFileFormat.RAW_16_BE: 12>

RAW_16_LE = <SoundFileFormat.RAW_16_LE: 13>

RAW_24_BE = <SoundFileFormat.RAW_24_BE: 14>

RAW_24_LE = <SoundFileFormat.RAW_24_LE: 15>

RAW_32_BE = <SoundFileFormat.RAW_32_BE: 16>

RAW_32_LE = <SoundFileFormat.RAW_32_LE: 17>

RAW_8_SIGNED = <SoundFileFormat.RAW_8_SIGNED: 10>

RAW_8_UNSIGNED = <SoundFileFormat.RAW_8_UNSIGNED: 11>

SESAM = <SoundFileFormat.SESAM: 7>

WAV = <SoundFileFormat.WAV: 0>

WAV_24 = <SoundFileFormat.WAV_24: 8>

WAV_32 = <SoundFileFormat.WAV_32: 9>

property name

property value
    
```

3.1.20 parselmouth.SpectralAnalysisWindowShape

```

class parselmouth.SpectralAnalysisWindowShape
    Bases: pybind11_object
    
```

Methods

<code>__init__</code>

Attributes

<code>BARTLETT</code>

<code>GAUSSIAN</code>

<code>HAMMING</code>

<code>HANNING</code>

<code>SQUARE</code>

<code>WELCH</code>

<code>name</code>

<code>value</code>

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.SpectralAnalysisWindowShape) → int`

`__init__(self: parselmouth.SpectralAnalysisWindowShape, value: int) → None`

`__init__(self: parselmouth.SpectralAnalysisWindowShape, arg0: str) → None`

`__int__(self: parselmouth.SpectralAnalysisWindowShape) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

`__repr__(self: object) → str`

`__str__()`

`name(self: handle) -> str`

`BARTLETT = <SpectralAnalysisWindowShape.BARTLETT: 2>`

`GAUSSIAN = <SpectralAnalysisWindowShape.GAUSSIAN: 5>`

`HAMMING = <SpectralAnalysisWindowShape.HAMMING: 1>`

`HANNING = <SpectralAnalysisWindowShape.HANNING: 4>`

SQUARE = <SpectralAnalysisWindowShape.SQUARE: 0>

WELCH = <SpectralAnalysisWindowShape.WELCH: 3>

property name

property value

3.1.21 parselmouth.Spectrogram

class parselmouth.Spectrogram

Bases: *TimeFrameSampled, Matrix*

Methods

`__init__`

`as_array`

`at_xy`

`copy`

`formula`

`frame_number_to_time`

`get_column_distance`

`get_end_time`

`get_frame_number_from_time`

`get_highest_x`

`get_highest_y`

`get_lowest_x`

`get_lowest_y`

`get_maximum`

`get_minimum`

`get_number_of_columns`

`get_number_of_frames`

`get_number_of_rows`

continues on next page

Table 12 – continued from previous page

<code>get_power_at</code>	
<code>get_row_distance</code>	
<code>get_start_time</code>	
<code>get_sum</code>	
<code>get_time_from_frame_number</code>	
<code>get_time_step</code>	
<code>get_total_duration</code>	
<code>get_value_at_xy</code>	
<code>get_value_in_cell</code>	
<code>get_x_of_column</code>	
<code>get_y_of_row</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_headerless_spreadsheet_file</code>	
<code>save_as_matrix_text_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_times_by</code>	
<code>scale_times_to</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>set_value</code>	
<code>shift_times_by</code>	
<code>shift_times_to</code>	

continues on next page

Table 12 – continued from previous page

<i>shift_x_by</i>
<i>shift_x_to</i>
<i>synthesize_sound</i>
<i>t_bins</i>
<i>t_grid</i>
<i>time_to_frame_number</i>
<i>to_sound</i>
<i>to_spectrum_slice</i>
<i>ts</i>
<i>x_bins</i>
<i>x_grid</i>
<i>xs</i>
<i>y_bins</i>
<i>y_grid</i>
<i>ys</i>

Attributes

<i>centre_time</i>
<i>class_name</i>
<i>dt</i>
<i>duration</i>
<i>dx</i>
<i>dy</i>
<i>end_time</i>
<i>full_name</i>

continues on next page

Table 13 – continued from previous page

<code>n_columns</code>
<code>n_frames</code>
<code>n_rows</code>
<code>name</code>
<code>nt</code>
<code>nx</code>
<code>ny</code>
<code>start_time</code>
<code>t1</code>
<code>time_range</code>
<code>time_step</code>
<code>tmax</code>
<code>tmin</code>
<code>total_duration</code>
<code>trange</code>
<code>values</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>
<code>y1</code>
<code>ymax</code>
<code>ymin</code>
<code>yrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object)` → `bool`

```

__hash__(self: object) → int

__index__(self: parselmouth.Data.FileFormat) → int

__init__(self: parselmouth.Data.FileFormat, value: int) → None
__init__(self: parselmouth.Data.FileFormat, arg0: str) → None

__int__(self: parselmouth.Data.FileFormat) → int

__ne__(self: object, other: object) → bool

__new__(**kwargs)

__repr__(self: object) → str

__str__()
    name(self: handle) -> str

BINARY = <FileFormat.BINARY: 2>

SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>

TEXT = <FileFormat.TEXT: 0>

property name

property value

__copy__(self: parselmouth.Data) → parselmouth.Data

__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data

__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool

__init__(*args, **kwargs)

__len__(self: parselmouth.Sampled) → int

__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool

__new__(**kwargs)

__str__(self: parselmouth.Thing) → str

as_array(self: parselmouth.Matrix) → numpy.ndarray[numpy.float64]

at_xy(self: parselmouth.Matrix, x: float, y: float) → float

copy(self: parselmouth.Data) → parselmouth.Data

formula(self: parselmouth.Matrix, formula: str, from_x: float | None = None, to_x: float | None = None,
        from_y: float | None = None, to_y: float | None = None) → None

formula(self: parselmouth.Matrix, formula: str, x_range: Tuple[float | None, float | None] = (None, None),
        y_range: Tuple[float | None, float | None] = (None, None)) → None

frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_column_distance(self: parselmouth.Matrix) → float
    
```

`get_end_time(self: parselmouth.Function) → float`
`get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float`
`get_highest_x(self: parselmouth.Matrix) → float`
`get_highest_y(self: parselmouth.Matrix) → float`
`get_lowest_x(self: parselmouth.Matrix) → float`
`get_lowest_y(self: parselmouth.Matrix) → float`
`get_maximum(self: parselmouth.Matrix) → float`
`get_minimum(self: parselmouth.Matrix) → float`
`get_number_of_columns(self: parselmouth.Matrix) → int`
`get_number_of_frames(self: parselmouth.Sampled) → int`
`get_number_of_rows(self: parselmouth.Matrix) → int`
`get_power_at(self: parselmouth.Spectrogram, time: float, frequency: float) → float`
`get_row_distance(self: parselmouth.Matrix) → float`
`get_start_time(self: parselmouth.Function) → float`
`get_sum(self: parselmouth.Matrix) → float`
`get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float`
`get_time_step(self: parselmouth.Sampled) → float`
`get_total_duration(self: parselmouth.Function) → float`
`get_value_at_xy(self: parselmouth.Matrix, x: float, y: float) → float`
`get_value_in_cell(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int]) → float`
`get_x_of_column(self: parselmouth.Matrix, column_number: Positive[int]) → float`
`get_y_of_row(self: parselmouth.Matrix, row_number: Positive[int]) → float`
`info(self: parselmouth.Thing) → str`

static read(*file_path: str*) → *parselmouth.Data*

Read a file into a *parselmouth.Data* object.

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

save(self: *parselmouth.Data*, file_path: *str*, format: *parselmouth.Data.FileFormat* = <*FileFormat.TEXT*: 0>) → *None*

save_as_binary_file(self: *parselmouth.Data*, file_path: *str*) → *None*

save_as_headerless_spreadsheet_file(self: *parselmouth.Matrix*, file_path: *str*) → *None*

save_as_matrix_text_file(self: *parselmouth.Matrix*, file_path: *str*) → *None*

save_as_short_text_file(self: *parselmouth.Data*, file_path: *str*) → *None*

save_as_text_file(self: *parselmouth.Data*, file_path: *str*) → *None*

scale_times_by(self: *parselmouth.Function*, scale: *Positive[float]*) → *None*

scale_times_to(self: *parselmouth.Function*, new_start_time: *float*, new_end_time: *float*) → *None*

scale_x_by(self: *parselmouth.Function*, scale: *Positive[float]*) → *None*

scale_x_to(self: *parselmouth.Function*, new_xmin: *float*, new_xmax: *float*) → *None*

set_value(self: *parselmouth.Matrix*, row_number: *Positive[int]*, column_number: *Positive[int]*, new_value: *float*) → *None*

shift_times_by(self: *parselmouth.Function*, seconds: *float*) → *None*

shift_times_to(self: *parselmouth.Function*, time: *float*, new_time: *float*) → *None*

shift_times_to(self: *parselmouth.Function*, time: *str*, new_time: *float*) → *None*

shift_x_by(self: *parselmouth.Function*, shift: *float*) → *None*

shift_x_to(self: *parselmouth.Function*, x: *float*, new_x: *float*) → *None*

synthesize_sound(self: *parselmouth.Spectrogram*, sampling_frequency: *Positive[float]* = 44100.0) → *parselmouth.Sound*

t_bins(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

t_grid(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

time_to_frame_number(self: *parselmouth.Sampled*, time: *float*) → *float*

to_sound(self: *parselmouth.Spectrogram*, sampling_frequency: *Positive[float]* = 44100.0) → *parselmouth.Sound*

to_spectrum_slice(self: *parselmouth.Spectrogram*, time: *float*) → *parselmouth.Spectrum*

ts(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

x_bins(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

x_grid(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

xs(self: *parselmouth.Sampled*) → *numpy.ndarray[numpy.float64]*

y_bins(self: *parselmouth.SampledXY*) → *numpy.ndarray[numpy.float64]*

y_grid(self: *parselmouth.SampledXY*) → *numpy.ndarray[numpy.float64]*

ys(self: *parselmouth.SampledXY*) → *numpy.ndarray[numpy.float64]*

```

__hash__ = None
property centre_time
property class_name
property dt
property duration
property dx
property dy
property end_time
property full_name
property n_columns
property n_frames
property n_rows
property name
property nt
property nx
property ny
property start_time
property t1
property time_range
property time_step
property tmax
property tmin
property total_duration
property trange
property values
property x1
property xmax
property xmin
property xrange
property y1
property ymax

```

property ymin

property yrange

3.1.22 parselmouth.Spectrum

class parselmouth.Spectrum

Bases: *Matrix*

Methods

<code>__init__</code>
<code>as_array</code>
<code>at_xy</code>
<code>cepstral_smoothing</code>
<code>copy</code>
<code>formula</code>
<code>get_band_density</code>
<code>get_band_density_difference</code>
<code>get_band_energy</code>
<code>get_band_energy_difference</code>
<code>get_bin_number_from_frequency</code>
<code>get_bin_width</code>
<code>get_center_of_gravity</code>
<code>get_central_moment</code>
<code>get_centre_of_gravity</code>
<code>get_column_distance</code>
<code>get_frequency_from_bin_number</code>
<code>get_highest_frequency</code>
<code>get_highest_x</code>

continues on next page

Table 14 – continued from previous page

<code>get_highest_y</code>	
<code>get_imaginary_value_in_bin</code>	
<code>get_kurtosis</code>	
<code>get_lowest_frequency</code>	
<code>get_lowest_x</code>	
<code>get_lowest_y</code>	
<code>get_maximum</code>	
<code>get_minimum</code>	
<code>get_number_of_bins</code>	
<code>get_number_of_columns</code>	
<code>get_number_of_rows</code>	
<code>get_real_value_in_bin</code>	
<code>get_row_distance</code>	
<code>get_skewness</code>	
<code>get_standard_deviation</code>	
<code>get_sum</code>	
<code>get_value_at_xy</code>	
<code>get_value_in_bin</code>	
<code>get_value_in_cell</code>	
<code>get_x_of_column</code>	
<code>get_y_of_row</code>	
<code>info</code>	
<code>lpc_smoothing</code>	
<code>read</code>	Read a file into a <code>parselmouth.Data</code> object.
<code>save</code>	
<code>save_as_binary_file</code>	

continues on next page

Table 14 – continued from previous page

<code>save_as_headerless_spreadsheet_file</code>
<code>save_as_matrix_text_file</code>
<code>save_as_short_text_file</code>
<code>save_as_text_file</code>
<code>scale_x_by</code>
<code>scale_x_to</code>
<code>set_imaginary_value_in_bin</code>
<code>set_real_value_in_bin</code>
<code>set_value</code>
<code>set_value_in_bin</code>
<code>shift_x_by</code>
<code>shift_x_to</code>
<code>to_sound</code>
<code>to_spectrogram</code>
<code>x_bins</code>
<code>x_grid</code>
<code>xs</code>
<code>y_bins</code>
<code>y_grid</code>
<code>ys</code>

Attributes

<i>bin_width</i>
<i>class_name</i>
<i>df</i>
<i>dx</i>
<i>dy</i>
<i>fmax</i>
<i>fmin</i>
<i>full_name</i>
<i>highest_frequency</i>
<i>lowest_frequency</i>
<i>n_bins</i>
<i>n_columns</i>
<i>n_rows</i>
<i>name</i>
<i>nf</i>
<i>nx</i>
<i>ny</i>
<i>values</i>
<i>x1</i>
<i>xmax</i>
<i>xmin</i>
<i>xrange</i>
<i>y1</i>
<i>ymax</i>
<i>ymin</i>
<i>yrange</i>

```

class FileFormat
    Bases: pybind11_object
    __eq__(self: object, other: object) → bool
    __hash__(self: object) → int
    __index__(self: parselmouth.Data.FileFormat) → int
    __init__(self: parselmouth.Data.FileFormat, value: int) → None
    __init__(self: parselmouth.Data.FileFormat, arg0: str) → None
    __int__(self: parselmouth.Data.FileFormat) → int
    __ne__(self: object, other: object) → bool
    __new__(**kwargs)
    __repr__(self: object) → str
    __str__()
        name(self: handle) -> str
    BINARY = <FileFormat.BINARY: 2>
    SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>
    TEXT = <FileFormat.TEXT: 0>
    property name
    property value
    __copy__(self: parselmouth.Data) → parselmouth.Data
    __deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
    __eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
    __getitem__(self: parselmouth.Spectrum, index: int) → complex
    __init__(self: parselmouth.Spectrum, values: numpy.ndarray[numpy.float64], maximum_frequency:
        Positive[float]) → None
    __init__(self: parselmouth.Spectrum, values: numpy.ndarray[numpy.complex128], maximum_frequency:
        Positive[float]) → None
    __len__(self: parselmouth.Sampled) → int
    __ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
    __new__(**kwargs)
    __setitem__(self: parselmouth.Spectrum, index: int, value: complex) → None
    __str__(self: parselmouth.Thing) → str
    as_array(self: parselmouth.Matrix) → numpy.ndarray[numpy.float64]
    at_xy(self: parselmouth.Matrix, x: float, y: float) → float

```

```

cepstral_smoothing(self: parselmouth.Spectrum, bandwidth: Positive[float] = 500.0) →
    parselmouth.Spectrum

copy(self: parselmouth.Data) → parselmouth.Data

formula(self: parselmouth.Matrix, formula: str, from_x: float | None = None, to_x: float | None = None,
    from_y: float | None = None, to_y: float | None = None) → None

formula(self: parselmouth.Matrix, formula: str, x_range: Tuple[float | None, float | None] = (None, None),
    y_range: Tuple[float | None, float | None] = (None, None)) → None

get_band_density(self: parselmouth.Spectrum, band_floor: float | None = None, band_ceiling: float | None
    = None) → float

get_band_density(self: parselmouth.Spectrum, band: Tuple[float | None, float | None] = (None, None)) →
    float

get_band_density_difference(self: parselmouth.Spectrum, low_band_floor: float | None = None,
    low_band_ceiling: float | None = None, high_band_floor: float | None =
    None, high_band_ceiling: float | None = None) → float

get_band_density_difference(self: parselmouth.Spectrum, low_band: Tuple[float | None, float | None] =
    (None, None), high_band: Tuple[float | None, float | None] = (None,
    None)) → float

get_band_energy(self: parselmouth.Spectrum, band_floor: float | None = None, band_ceiling: float | None
    = None) → float

get_band_energy(self: parselmouth.Spectrum, band: Tuple[float | None, float | None] = (None, None)) →
    float

get_band_energy_difference(self: parselmouth.Spectrum, low_band_floor: float | None = None,
    low_band_ceiling: float | None = None, high_band_floor: float | None =
    None, high_band_ceiling: float | None = None) → float

get_band_energy_difference(self: parselmouth.Spectrum, low_band: Tuple[float | None, float | None] =
    (None, None), high_band: Tuple[float | None, float | None] = (None,
    None)) → float

get_bin_number_from_frequency(self: parselmouth.Spectrum, frequency: float) → float

get_bin_width(self: parselmouth.Spectrum) → float

get_center_of_gravity(self: parselmouth.Spectrum, power: Positive[float] = 2.0) → float

get_central_moment(self: parselmouth.Spectrum, moment: Positive[float], power: Positive[float] = 2.0) →
    float

get_centre_of_gravity(self: parselmouth.Spectrum, power: Positive[float] = 2.0) → float

get_column_distance(self: parselmouth.Matrix) → float

get_frequency_from_bin_number(self: parselmouth.Spectrum, band_number: Positive[int]) → float

get_highest_frequency(self: parselmouth.Spectrum) → float

get_highest_x(self: parselmouth.Matrix) → float

get_highest_y(self: parselmouth.Matrix) → float
    
```

```

get_imaginary_value_in_bin(self: parselmouth.Spectrum, bin_number: Positive[int]) → float
get_kurtosis(self: parselmouth.Spectrum, power: Positive[float] = 2.0) → float
get_lowest_frequency(self: parselmouth.Spectrum) → float
get_lowest_x(self: parselmouth.Matrix) → float
get_lowest_y(self: parselmouth.Matrix) → float
get_maximum(self: parselmouth.Matrix) → float
get_minimum(self: parselmouth.Matrix) → float
get_number_of_bins(self: parselmouth.Spectrum) → int
get_number_of_columns(self: parselmouth.Matrix) → int
get_number_of_rows(self: parselmouth.Matrix) → int
get_real_value_in_bin(self: parselmouth.Spectrum, bin_number: Positive[int]) → float
get_row_distance(self: parselmouth.Matrix) → float
get_skewness(self: parselmouth.Spectrum, power: Positive[float] = 2.0) → float
get_standard_deviation(self: parselmouth.Spectrum, power: Positive[float] = 2.0) → float
get_sum(self: parselmouth.Matrix) → float
get_value_at_xy(self: parselmouth.Matrix, x: float, y: float) → float
get_value_in_bin(self: parselmouth.Spectrum, bin_number: Positive[int]) → complex
get_value_in_cell(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int])
    → float
get_x_of_column(self: parselmouth.Matrix, column_number: Positive[int]) → float
get_y_of_row(self: parselmouth.Matrix, row_number: Positive[int]) → float
info(self: parselmouth.Thing) → str
lpc_smoothing(self: parselmouth.Spectrum, num_peaks: Positive[int] = 5, pre_emphasis_from:
    Positive[float] = 50.0) → parselmouth.Spectrum

```

static read(file_path: str) → *parselmouth.Data*

Read a file into a *parselmouth.Data* object.

Parameters

file_path (str) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

```

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    → None

save_as_binary_file(self: parselmouth.Data, file_path: str) → None

save_as_headerless_spreadsheet_file(self: parselmouth.Matrix, file_path: str) → None

save_as_matrix_text_file(self: parselmouth.Matrix, file_path: str) → None

save_as_short_text_file(self: parselmouth.Data, file_path: str) → None

save_as_text_file(self: parselmouth.Data, file_path: str) → None

scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None

set_imaginary_value_in_bin(self: parselmouth.Spectrum, bin_number: Positive[int], value: float) →
    None

set_real_value_in_bin(self: parselmouth.Spectrum, bin_number: Positive[int], value: float) → None

set_value(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int], new_value:
    float) → None

set_value_in_bin(self: parselmouth.Spectrum, bin_number: Positive[int], value: complex) → None

shift_x_by(self: parselmouth.Function, shift: float) → None

shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None

to_sound(self: parselmouth.Spectrum) → parselmouth.Sound

to_spectrogram(self: parselmouth.Spectrum) → parselmouth.Spectrogram

x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

y_bins(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

y_grid(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

ys(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

__hash__ = None

property bin_width

property class_name

property df

property dx

property dy

property fmax
    
```

property fmin
property full_name
property highest_frequency
property lowest_frequency
property n_bins
property n_columns
property n_rows
property name
property nf
property nx
property ny
property values
property x1
property xmax
property xmin
property xrange
property y1
property ymax
property ymin
property yrange

3.1.23 parselmouth.TextGrid

class `parselmouth.TextGrid`

Bases: *Function*

Methods

<code>__init__</code>	
<code>copy</code>	
<code>from_tgt</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>to_tgt</code>	

Attributes

<code>class_name</code>
<code>full_name</code>
<code>name</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object)` → bool

```

__hash__(self: object) → int

__index__(self: parselmouth.Data.FileFormat) → int

__init__(self: parselmouth.Data.FileFormat, value: int) → None
__init__(self: parselmouth.Data.FileFormat, arg0: str) → None

__int__(self: parselmouth.Data.FileFormat) → int

__ne__(self: object, other: object) → bool

__new__(**kwargs)

__repr__(self: object) → str

__str__()
    name(self: handle) -> str

BINARY = <FileFormat.BINARY: 2>

SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>

TEXT = <FileFormat.TEXT: 0>

property name

property value

__copy__(self: parselmouth.Data) → parselmouth.Data

__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data

__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool

__init__(self: parselmouth.TextGrid, start_time: float, end_time: float, tier_names: str, point_tier_names:
    str) → None
__init__(self: parselmouth.TextGrid, start_time: float, end_time: float, tier_names: List[str] = [],
    point_tier_names: List[str] = []) → None
__init__(self: parselmouth.TextGrid, tgt_text_grid: tgt.core.TextGrid) → None

__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool

__new__(**kwargs)

__str__(self: parselmouth.Thing) → str

copy(self: parselmouth.Data) → parselmouth.Data

static from_tgt(tgt_text_grid: tgt.core.TextGrid) → parselmouth.TextGrid

info(self: parselmouth.Thing) → str

static read(file_path: str) → parselmouth.Data
    Read a file into a parselmouth.Data object.

```

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

save(*self*: *parselmouth.Data*, *file_path*: *str*, *format*: *parselmouth.Data.FileFormat* = <*FileFormat.TEXT*: 0>) → *None*

save_as_binary_file(*self*: *parselmouth.Data*, *file_path*: *str*) → *None*

save_as_short_text_file(*self*: *parselmouth.Data*, *file_path*: *str*) → *None*

save_as_text_file(*self*: *parselmouth.Data*, *file_path*: *str*) → *None*

scale_x_by(*self*: *parselmouth.Function*, *scale*: *Positive[float]*) → *None*

scale_x_to(*self*: *parselmouth.Function*, *new_xmin*: *float*, *new_xmax*: *float*) → *None*

shift_x_by(*self*: *parselmouth.Function*, *shift*: *float*) → *None*

shift_x_to(*self*: *parselmouth.Function*, *x*: *float*, *new_x*: *float*) → *None*

to_tgt(*self*: *parselmouth.TextGrid*, *, *include_empty_intervals*: *bool* = *False*) → *tgt.core.TextGrid*

__hash__ = *None*

property class_name

property full_name

property name

property xmax

property xmin

property xrange

3.1.24 parselmouth.Thing

class *parselmouth.Thing*

Bases: *pybind11_object*

Methods

__init__

info

Attributes

<code>class_name</code>
<code>full_name</code>
<code>name</code>

```

__init__(*args, **kwargs)
__new__(**kwargs)
__str__(self: parselmouth.Thing) → str
info(self: parselmouth.Thing) → str
property class_name
property full_name
property name

```

3.1.25 `parselmouth.TimeFrameSampled`

class `parselmouth.TimeFrameSampled`
 Bases: *TimeFunction*, *Sampled*

Methods

<code>__init__</code>
<code>copy</code>
<code>frame_number_to_time</code>
<code>get_end_time</code>
<code>get_frame_number_from_time</code>
<code>get_number_of_frames</code>
<code>get_start_time</code>
<code>get_time_from_frame_number</code>
<code>get_time_step</code>
<code>get_total_duration</code>

continues on next page

Table 15 – continued from previous page

<code>info</code>	
<code>read</code>	Read a file into a <i><code>parselmouth.Data</code></i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_times_by</code>	
<code>scale_times_to</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_times_by</code>	
<code>shift_times_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>t_bins</code>	
<code>t_grid</code>	
<code>time_to_frame_number</code>	
<code>ts</code>	
<code>x_bins</code>	
<code>x_grid</code>	
<code>xs</code>	

Attributes

<code>centre_time</code>
<code>class_name</code>
<code>dt</code>
<code>duration</code>
<code>dx</code>
<code>end_time</code>
<code>full_name</code>
<code>n_frames</code>
<code>name</code>
<code>nt</code>
<code>nx</code>
<code>start_time</code>
<code>t1</code>
<code>time_range</code>
<code>time_step</code>
<code>tmax</code>
<code>tmin</code>
<code>total_duration</code>
<code>trange</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__`(*self*: *object*, *other*: *object*) → bool

```

__hash__(self: object) → int

__index__(self: parselmouth.Data.FileFormat) → int

__init__(self: parselmouth.Data.FileFormat, value: int) → None
__init__(self: parselmouth.Data.FileFormat, arg0: str) → None

__int__(self: parselmouth.Data.FileFormat) → int

__ne__(self: object, other: object) → bool

__new__(**kwargs)

__repr__(self: object) → str

__str__()
    name(self: handle) -> str

BINARY = <FileFormat.BINARY: 2>

SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>

TEXT = <FileFormat.TEXT: 0>

property name

property value

__copy__(self: parselmouth.Data) → parselmouth.Data

__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data

__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool

__init__(*args, **kwargs)

__len__(self: parselmouth.Sampled) → int

__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool

__new__(**kwargs)

__str__(self: parselmouth.Thing) → str

copy(self: parselmouth.Data) → parselmouth.Data

frame_number_to_time(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_end_time(self: parselmouth.Function) → float

get_frame_number_from_time(self: parselmouth.Sampled, time: float) → float

get_number_of_frames(self: parselmouth.Sampled) → int

get_start_time(self: parselmouth.Function) → float

get_time_from_frame_number(self: parselmouth.Sampled, frame_number: Positive[int]) → float

get_time_step(self: parselmouth.Sampled) → float

```

get_total_duration(self: parselmouth.Function) → float

info(self: parselmouth.Thing) → str

static read(file_path: str) → parselmouth.Data

Read a file into a *parselmouth.Data* object.

Parameters

file_path (str) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>) → None

save_as_binary_file(self: parselmouth.Data, file_path: str) → None

save_as_short_text_file(self: parselmouth.Data, file_path: str) → None

save_as_text_file(self: parselmouth.Data, file_path: str) → None

scale_times_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_times_to(self: parselmouth.Function, new_start_time: float, new_end_time: float) → None

scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None

shift_times_by(self: parselmouth.Function, seconds: float) → None

shift_times_to(self: parselmouth.Function, time: float, new_time: float) → None

shift_times_to(self: parselmouth.Function, time: str, new_time: float) → None

shift_x_by(self: parselmouth.Function, shift: float) → None

shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None

t_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

t_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

time_to_frame_number(self: parselmouth.Sampled, time: float) → float

ts(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

__hash__ = None

```

property centre_time
property class_name
property dt
property duration
property dx
property end_time
property full_name
property n_frames
property name
property nt
property nx
property start_time
property t1
property time_range
property time_step
property tmax
property tmin
property total_duration
property trange
property x1
property xmax
property xmin
property xrange

```

3.1.26 `parselmouth.TimeFunction`

```
class parselmouth.TimeFunction
```

Bases: *Function*

Methods

<code>__init__</code>	
<code>copy</code>	
<code>get_end_time</code>	
<code>get_start_time</code>	
<code>get_total_duration</code>	
<code>info</code>	
<code>read</code>	Read a file into a <i>parselmouth.Data</i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale_times_by</code>	
<code>scale_times_to</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>shift_times_by</code>	
<code>shift_times_to</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	

Attributes

<code>centre_time</code>
<code>class_name</code>
<code>duration</code>
<code>end_time</code>
<code>full_name</code>
<code>name</code>
<code>start_time</code>
<code>time_range</code>
<code>tmax</code>
<code>tmin</code>
<code>total_duration</code>
<code>trange</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object)` → `bool`

`__hash__(self: object)` → `int`

`__index__(self: parselmouth.Data.FileFormat)` → `int`

`__init__(self: parselmouth.Data.FileFormat, value: int)` → `None`

`__init__(self: parselmouth.Data.FileFormat, arg0: str)` → `None`

`__int__(self: parselmouth.Data.FileFormat)` → `int`

`__ne__(self: object, other: object)` → `bool`

`__new__(**kwargs)`

`__repr__(self: object)` → `str`

```

__str__()
    name(self: handle) -> str

BINARY = <FileFormat.BINARY: 2>

SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>

TEXT = <FileFormat.TEXT: 0>

property name
property value

__copy__(self: parselmouth.Data) -> parselmouth.Data
__deepcopy__(self: parselmouth.Data, memo: dict) -> parselmouth.Data
__eq__(self: parselmouth.Data, other: parselmouth.Data) -> bool
__init__(*args, **kwargs)
__ne__(self: parselmouth.Data, other: parselmouth.Data) -> bool
__new__(**kwargs)
__str__(self: parselmouth.Thing) -> str

copy(self: parselmouth.Data) -> parselmouth.Data

get_end_time(self: parselmouth.Function) -> float
get_start_time(self: parselmouth.Function) -> float
get_total_duration(self: parselmouth.Function) -> float

info(self: parselmouth.Thing) -> str

static read(file_path: str) -> parselmouth.Data
    Read a file into a parselmouth.Data object.

    Parameters
        file_path (str) – The path of the file on disk to read.

    Returns
        The Praat Data object that was read.

    Return type
        parselmouth.Data

    See also:
        Praat: “Read from file...”

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    -> None

save_as_binary_file(self: parselmouth.Data, file_path: str) -> None

save_as_short_text_file(self: parselmouth.Data, file_path: str) -> None

save_as_text_file(self: parselmouth.Data, file_path: str) -> None

```

```

scale_times_by(self: parselmouth.Function, scale: Positive[float]) → None
scale_times_to(self: parselmouth.Function, new_start_time: float, new_end_time: float) → None
scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None
scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None
shift_times_by(self: parselmouth.Function, seconds: float) → None
shift_times_to(self: parselmouth.Function, time: float, new_time: float) → None
shift_times_to(self: parselmouth.Function, time: str, new_time: float) → None
shift_x_by(self: parselmouth.Function, shift: float) → None
shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None
__hash__ = None
property centre_time
property class_name
property duration
property end_time
property full_name
property name
property start_time
property time_range
property tmax
property tmin
property total_duration
property trange
property xmax
property xmin
property xrange
    
```

3.1.27 parselmouth.ValueInterpolation

```

class parselmouth.ValueInterpolation
    Bases: pybind11_object
    
```

Methods

`__init__`

Attributes

`CUBIC`

`LINEAR`

`NEAREST`

`SINC70`

`SINC700`

`name`

`value`

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.ValueInterpolation) → int`

`__init__(self: parselmouth.ValueInterpolation, value: int) → None`

`__init__(self: parselmouth.ValueInterpolation, arg0: str) → None`

`__int__(self: parselmouth.ValueInterpolation) → int`

`__ne__(self: object, other: object) → bool`

`__new__(**kwargs)`

`__repr__(self: object) → str`

`__str__()`

`name(self: handle) -> str`

`CUBIC = <ValueInterpolation.CUBIC: 2>`

`LINEAR = <ValueInterpolation.LINEAR: 1>`

`NEAREST = <ValueInterpolation.NEAREST: 0>`

`SINC70 = <ValueInterpolation.SINC70: 3>`

`SINC700 = <ValueInterpolation.SINC700: 4>`

`property name`

`property value`

3.1.28 parselmouth.Vector

class `parselmouth.Vector`

Bases: *Matrix*

Methods

<code>__init__</code>
<code>add</code>
<code>as_array</code>
<code>at_xy</code>
<code>copy</code>
<code>divide</code>
<code>formula</code>
<code>get_column_distance</code>
<code>get_highest_x</code>
<code>get_highest_y</code>
<code>get_lowest_x</code>
<code>get_lowest_y</code>
<code>get_maximum</code>
<code>get_minimum</code>
<code>get_number_of_columns</code>
<code>get_number_of_rows</code>
<code>get_row_distance</code>
<code>get_sum</code>
<code>get_value</code>
<code>get_value_at_xy</code>
<code>get_value_in_cell</code>
<code>get_x_of_column</code>

continues on next page

Table 16 – continued from previous page

<code>get_y_of_row</code>	
<code>info</code>	
<code>multiply</code>	
<code>read</code>	Read a file into a <i><code>parselmouth.Data</code></i> object.
<code>save</code>	
<code>save_as_binary_file</code>	
<code>save_as_headerless_spreadsheet_file</code>	
<code>save_as_matrix_text_file</code>	
<code>save_as_short_text_file</code>	
<code>save_as_text_file</code>	
<code>scale</code>	
<code>scale_peak</code>	
<code>scale_x_by</code>	
<code>scale_x_to</code>	
<code>set_value</code>	
<code>shift_x_by</code>	
<code>shift_x_to</code>	
<code>subtract</code>	
<code>subtract_mean</code>	
<code>x_bins</code>	
<code>x_grid</code>	
<code>xs</code>	
<code>y_bins</code>	
<code>y_grid</code>	
<code>ys</code>	

Attributes

<code>class_name</code>
<code>dx</code>
<code>dy</code>
<code>full_name</code>
<code>n_columns</code>
<code>n_rows</code>
<code>name</code>
<code>nx</code>
<code>ny</code>
<code>values</code>
<code>x1</code>
<code>xmax</code>
<code>xmin</code>
<code>xrange</code>
<code>y1</code>
<code>ymax</code>
<code>ymin</code>
<code>yrange</code>

class FileFormat

Bases: `pybind11_object`

`__eq__(self: object, other: object) → bool`

`__hash__(self: object) → int`

`__index__(self: parselmouth.Data.FileFormat) → int`

`__init__(self: parselmouth.Data.FileFormat, value: int) → None`

`__init__(self: parselmouth.Data.FileFormat, arg0: str) → None`

`__int__(self: parselmouth.Data.FileFormat) → int`

```

__ne__(self: object, other: object) → bool

__new__(**kwargs)

__repr__(self: object) → str

__str__()
    name(self: handle) -> str

BINARY = <FileFormat.BINARY: 2>

SHORT_TEXT = <FileFormat.SHORT_TEXT: 1>

TEXT = <FileFormat.TEXT: 0>

property name
property value

__add__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__copy__(self: parselmouth.Data) → parselmouth.Data
__deepcopy__(self: parselmouth.Data, memo: dict) → parselmouth.Data
__eq__(self: parselmouth.Data, other: parselmouth.Data) → bool
__iadd__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__imul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__init__(*args, **kwargs)
__isub__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__itruediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__len__(self: parselmouth.Sampled) → int
__mul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__ne__(self: parselmouth.Data, other: parselmouth.Data) → bool
__new__(**kwargs)
__radd__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__rmul__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
__str__(self: parselmouth.Thing) → str
__sub__(self: parselmouth.Vector, number: float) → parselmouth.Vector
__truediv__(self: parselmouth.Vector, factor: float) → parselmouth.Vector
add(self: parselmouth.Vector, number: float) → None
as_array(self: parselmouth.Matrix) → numpy.ndarray[numpy.float64]
at_xy(self: parselmouth.Matrix, x: float, y: float) → float

```

```

copy(self: parselmouth.Data) → parselmouth.Data

divide(self: parselmouth.Vector, factor: float) → None

formula(self: parselmouth.Matrix, formula: str, from_x: float | None = None, to_x: float | None = None,
        from_y: float | None = None, to_y: float | None = None) → None

formula(self: parselmouth.Matrix, formula: str, x_range: Tuple[float | None, float | None] = (None, None),
        y_range: Tuple[float | None, float | None] = (None, None)) → None

get_column_distance(self: parselmouth.Matrix) → float

get_highest_x(self: parselmouth.Matrix) → float

get_highest_y(self: parselmouth.Matrix) → float

get_lowest_x(self: parselmouth.Matrix) → float

get_lowest_y(self: parselmouth.Matrix) → float

get_maximum(self: parselmouth.Matrix) → float

get_minimum(self: parselmouth.Matrix) → float

get_number_of_columns(self: parselmouth.Matrix) → int

get_number_of_rows(self: parselmouth.Matrix) → int

get_row_distance(self: parselmouth.Matrix) → float

get_sum(self: parselmouth.Matrix) → float

get_value(self: parselmouth.Vector, x: float, channel: Optional[int] = None, interpolation:
        parselmouth.ValueInterpolation = <ValueInterpolation.CUBIC: 2>) → float

get_value_at_xy(self: parselmouth.Matrix, x: float, y: float) → float

get_value_in_cell(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int])
        → float

get_x_of_column(self: parselmouth.Matrix, column_number: Positive[int]) → float

get_y_of_row(self: parselmouth.Matrix, row_number: Positive[int]) → float

info(self: parselmouth.Thing) → str

multiply(self: parselmouth.Vector, factor: float) → None

static read(file_path: str) → parselmouth.Data
    Read a file into a parselmouth.Data object.

```

Parameters

file_path (*str*) – The path of the file on disk to read.

Returns

The Praat Data object that was read.

Return type

parselmouth.Data

See also:

Praat: “Read from file...”

```

save(self: parselmouth.Data, file_path: str, format: parselmouth.Data.FileFormat = <FileFormat.TEXT: 0>)
    → None

save_as_binary_file(self: parselmouth.Data, file_path: str) → None

save_as_headerless_spreadsheet_file(self: parselmouth.Matrix, file_path: str) → None

save_as_matrix_text_file(self: parselmouth.Matrix, file_path: str) → None

save_as_short_text_file(self: parselmouth.Data, file_path: str) → None

save_as_text_file(self: parselmouth.Data, file_path: str) → None

scale(self: parselmouth.Vector, scale: Positive[float]) → None

scale_peak(self: parselmouth.Vector, new_peak: Positive[float] = 0.99) → None

scale_x_by(self: parselmouth.Function, scale: Positive[float]) → None

scale_x_to(self: parselmouth.Function, new_xmin: float, new_xmax: float) → None

set_value(self: parselmouth.Matrix, row_number: Positive[int], column_number: Positive[int], new_value:
    float) → None

shift_x_by(self: parselmouth.Function, shift: float) → None

shift_x_to(self: parselmouth.Function, x: float, new_x: float) → None

subtract(self: parselmouth.Vector, number: float) → None

subtract_mean(self: parselmouth.Vector) → None

x_bins(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

x_grid(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

xs(self: parselmouth.Sampled) → numpy.ndarray[numpy.float64]

y_bins(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

y_grid(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

ys(self: parselmouth.SampledXY) → numpy.ndarray[numpy.float64]

__hash__ = None

property class_name

property dx

property dy

property full_name

property n_columns

property n_rows

property name

```

property `nx`
property `ny`
property `values`
property `x1`
property `xmax`
property `xmin`
property `xrange`
property `y1`
property `ymax`
property `ymin`
property `yrange`

3.1.29 `parselmouth.WindowShape`

class `parselmouth.WindowShape`

Bases: `pybind11_object`

Methods

<code>__init__</code>

Attributes

<code>GAUSSIAN1</code>
<code>GAUSSIAN2</code>
<code>GAUSSIAN3</code>
<code>GAUSSIAN4</code>
<code>GAUSSIAN5</code>
<code>HAMMING</code>
<code>HANNING</code>
<code>KAISER1</code>
<code>KAISER2</code>
<code>PARABOLIC</code>
<code>RECTANGULAR</code>
<code>TRIANGULAR</code>
<code>name</code>
<code>value</code>

```

__eq__(self: object, other: object) → bool
__hash__(self: object) → int
__index__(self: parselmouth.WindowShape) → int
__init__(self: parselmouth.WindowShape, value: int) → None
__init__(self: parselmouth.WindowShape, arg0: str) → None
__int__(self: parselmouth.WindowShape) → int
__ne__(self: object, other: object) → bool
__new__(**kwargs)
__repr__(self: object) → str
__str__()
    name(self: handle) -> str
GAUSSIAN1 = <WindowShape.GAUSSIAN1: 5>
GAUSSIAN2 = <WindowShape.GAUSSIAN2: 6>

```

```

GAUSSIAN3 = <WindowShape.GAUSSIAN3: 7>
GAUSSIAN4 = <WindowShape.GAUSSIAN4: 8>
GAUSSIAN5 = <WindowShape.GAUSSIAN5: 9>
HAMMING = <WindowShape.HAMMING: 4>
HANNING = <WindowShape.HANNING: 3>
KAISER1 = <WindowShape.KAISER1: 10>
KAISER2 = <WindowShape.KAISER2: 11>
PARABOLIC = <WindowShape.PARABOLIC: 2>
RECTANGULAR = <WindowShape.RECTANGULAR: 0>
TRIANGULAR = <WindowShape.TRIANGULAR: 1>

property name
property value

```

Exceptions

<i>PraatError</i>
<i>PraatFatal</i>
<i>PraatWarning</i>

3.1.30 parselmouth.PraatError

exception `parselmouth.PraatError`

3.1.31 parselmouth.PraatFatal

exception `parselmouth.PraatFatal`

3.1.32 parselmouth.PraatWarning

exception `parselmouth.PraatWarning`

3.2 parselmouth.praat

Submodule with functions to call Praat commands and run Praat scripts.

In addition to the Python interface provided in the top-level *parselmouth* module, this module provides a Python interface to access any Praat command or action through the *parselmouth.praat.call* function from Python. Moreover, existing Praat scripts can be run through the *parselmouth.praat.run* and *parselmouth.praat.run_file* functions.

Note: Any command or script accessing the graphical user interface or other interactive functionality of Praat will not work, since the internal Praat source is in Parselmouth is run in *batch mode*. This is similar to running a Praat script from the command line (see *Praat: “Scripting 6.9. Calling from the command line”*).

Functions

<i>call</i>	Call a Praat command.
<i>run</i>	Run a Praat script.
<i>run_file</i>	Run a Praat script from file.

3.2.1 parselmouth.praat.call

parselmouth.praat.call(*command: str, *args, **kwargs*) → *object*

parselmouth.praat.call(*object: parselmouth.Data, command: str, *args, **kwargs*) → *object*

parselmouth.praat.call(*objects: List[parselmouth.Data], command: str, *args, **kwargs*) → *object*

Call a Praat command.

This function provides a Python interface to call available Praat commands based on the label in the Praat user interface and documentation, similar to the Praat scripting language.

Calling a Praat command through this function roughly corresponds to the following scenario in the Praat user interface or scripting language:

1. Zero, one, or multiple *parselmouth.Data* objects are put into Praat’s global object list and are ‘selected’.
2. The Python argument values are converted into Praat values; see below.
3. The Praat command is executed on the selected objects with the converted values as arguments.
4. The result of the command is returned. The type of the result depends on the result of the Praat command; see below.
5. Praat’s object list is emptied again, such that a future execution of this function is independent from the current call.

The use of *call* is demonstrated in the *Pitch manipulation and Praat commands* example.

Parameters

- **object** (*parselmouth.Data*) – A single object to add to the Praat object list, which will be selected when the Praat command is called.
- **objects** (*List[parselmouth.Data]*) – Multiple objects to be added to the Praat object list, which will be selected when the Praat command is called.

- **command** (*str*) – The Praat action to call. This is the same command name as one would use in a Praat script and corresponds to the label on the button in the Praat user interface.
- ***args** – The list of values to be passed as arguments to the Praat command. Allowed types for these arguments are:
 - *int* or *float*: passed as a Praat numeric value
 - *bool*: converted into "yes"/"no"
 - *str*: passed as Praat string value
 - *numpy.ndarray*: passed as Praat vector or matrix, if the array contains numeric values and is 1D or 2D, respectively.

Keyword Arguments

- **extra_objects** (*List[parselmouth.Data]*) – Extra objects added to the Praat object list that will not be selected when the command is called (default value: []).
- **return_string** (*bool*) – Return the raw string written in the Praat info window instead of the converted Python object (default value: *False*).

Returns

The result of the Praat command. The actual value returned depends on what the Praat command does. The following types can be returned:

- If *return_string=True* was passed, a *str* value is returned, which contains the text that would have been written to the Praat info window.
- A *float*, *int*, *bool*, or *complex* value is returned when the Praat command would write such a value to the Praat info window.
- A *numpy.ndarray* value is returned if the command returns a Praat vector or matrix.
- A *parselmouth.Data* object is returned if the command always creates exactly one object. If the actual type of the Praat object is available in Parselmouth, an object of a subtype of *parselmouth.Data* is returned.
- A list of *parselmouth.Data* objects is returned if the command can create multiple new objects (even if this particular execution of the command only added one object to the Praat object list).
- A *str* is returned when a string or info text would be written to the Praat info window.

Return type

object

See also:

parselmouth.praat.run, *parselmouth.praat.run_file*, *Praat*: “Scripting”

3.2.2 `parselmouth.praat.run`

`parselmouth.praat.run(script: str, *args, **kwargs) → object`

`parselmouth.praat.run(object: parselmouth.Data, script: str, *args, **kwargs) → object`

`parselmouth.praat.run(objects: List[parselmouth.Data], script: str, *args, **kwargs) → object`

Run a Praat script.

Given a string with the contents of a Praat script, run this script as if it was run inside Praat itself. Similarly to [`parselmouth.praat.call`](#), Parselmouth objects and Python argument values can be passed into the script.

Calling this function roughly corresponds to the following sequence of steps in Praat:

1. Zero, one, or multiple [`parselmouth.Data`](#) objects are put into Praat’s global object list and are ‘selected’.
2. The Python argument values are converted into Praat values; see [`call`](#).
3. The Praat script is opened and run with the converted values as arguments; see *Praat: “Scripting 6.1. Arguments to the script”*.
4. The results of the execution of the script are returned; see below.
5. Praat’s object list is emptied again, such that a future execution of this function is independent from the current call.

Note that the script will be run in Praat’s so-called ‘batch’ mode; see *Praat: “Scripting 6.9. Calling from the command line”*. Since the script is run from inside a Python program, the Praat functionality is run without graphical user interface and no windows (such as “View & Edit”) can be opened by the Praat script. However, the functionality in these windows is also available in different ways: for example, opening a *Sound* object in a “View & Edit” window, making a selection, and choosing “Extract selected sound (windowed)...” can also be achieved by directly using the “Extract part...” command of the *Sound* object.

Parameters

- **object** ([`parselmouth.Data`](#)) – A single object to add to the Praat object list, which will be selected when the Praat script is run.
- **objects** ([List\[`parselmouth.Data`\]](#)) – Multiple objects to be added to the Praat object list, which will be selected when the Praat script is run.
- **script** ([str](#)) – The content of the Praat script to be run.
- ***args** – The list of values to be passed as arguments to the Praat script. For more details on the allowed types of these argument, see [`call`](#).

Keyword Arguments

- **extra_objects** ([List\[`parselmouth.Data`\]](#)) – Extra objects added to the Praat object list that will not be selected when the command is called (default value: []).
- **capture_output** ([bool](#)) – Intercept and also return the output written to the Praat info window, instead of forwarding it to the Python standard output; see below (default value: False).
- **return_variables** ([bool](#)) – Also return a [dict](#) of the Praat variables and their values at the end of the script’s execution; see below (default value: False).

Returns

A list of [`parselmouth.Data`](#) objects selected at the end of the script’s execution.

Optionally, extra values are returned:

- A [str](#) containing the intercepted output if `capture_output=True` was passed.

- A `dict` mapping variable names (`str`) to their values (`object`) if `return_variables` is `True`. The values of Praat’s variables get converted to Python values:
 - A Praat string variable, with a name ending in `$`, is returned as `str` value.
 - A Praat vector or matrix variable, respectively ending in `#` or `##`, is returned as `numpy.ndarray`.
 - A numeric variable, without variable name suffix, is converted to a Python `float`.

Return type
`object`

See also:

`parselmouth.praat.run_file`, `parselmouth.praat.call`, Praat: “Scripting”

3.2.3 `parselmouth.praat.run_file`

`parselmouth.praat.run_file(path: str, *args, **kwargs) → object`

`parselmouth.praat.run_file(object: parselmouth.Data, path: str, *args, **kwargs) → object`

`parselmouth.praat.run_file(objects: List[parselmouth.Data], path: str, *args, **kwargs) → object`

Run a Praat script from file.

Given the filename of a Praat script, the script is read and run the same way as a script string passed to `parselmouth.praat.run`. See `run` for details on the manner in which the script gets executed.

One thing to note is that relative filenames in the Praat script (including those in potential ‘include’ statements in the script; see Praat: “Scripting 5.8. Including other scripts”) will be resolved relative to the path of the script file, just like in Praat. Also note that Praat accomplishes this by temporarily changing the current working during the execution of the script.

Parameters

- **object** (`parselmouth.Data`) – A single object to add to the Praat object list, which will be selected when the Praat script is run.
- **objects** (`List[parselmouth.Data]`) – Multiple objects to be added to the Praat object list, which will be selected when the Praat script is run.
- **path** (`str`) – The filename of the Praat script to run.
- ***args** – The list of values to be passed as arguments to the Praat script. For more details on the allowed types of these argument, see `call`.

Keyword Arguments

- **keep_cwd** (`bool`) – Keep the current working directory (see `os.getcwd`) when running the script, rather than changing it to the script’s parent directory, as Praat does by default (default value: `False`). Note that even when set to `True`, the filenames in the Praat script’s include statements will be resolved relatively to the directory containing the script.
- ****kwargs** – See `parselmouth.praat.run`.

Returns

See `parselmouth.praat.run`.

Return type
`object`

See also:

*`parselmouth.praat.run`, `parselmouth.praat.call`, *Praat: “Scripting”**

CITING PARSELMOUTH

A manuscript introducing Parselmouth (and supplementary material) has been published in the *Journal of Phonetics*. Scientific work and publications can for now cite Parselmouth in the following way:

Jadoul, Y., Thompson, B., & de Boer, B. (2018). Introducing Parselmouth: A Python interface to Praat. *Journal of Phonetics*, 71, 1-15. <https://doi.org/10.1016/j.wocn.2018.07.001>

```
@article{parselmouth,  
  author = "Yannick Jadoul and Bill Thompson and Bart de Boer",  
  title = "Introducing {P}arselmouth: A {P}ython interface to {P}raat",  
  journal = "Journal of Phonetics",  
  volume = "71",  
  pages = "1--15",  
  year = "2018",  
  doi = "https://doi.org/10.1016/j.wocn.2018.07.001"  
}
```

Since Parselmouth exposes existing Praat functionality and algorithm implementations, we suggest also citing Praat when using Parselmouth in scientific research:

Boersma, P., & Weenink, D. (2021). Praat: doing phonetics by computer [Computer program]. Version 6.1.38, retrieved 2 January 2021 from <http://www.praat.org/>

```
@misc{praat,  
  author = "Paul Boersma and David Weenink",  
  title = "{P}raat: doing phonetics by computer [{C}omputer program]",  
  howpublished = "Version 6.1.38, retrieved 2 January 2021 \url{http://www.praat.org/}"  
  ↪ ,  
  year = "2021"  
}
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`parselmouth`, [27](#)

`parselmouth.praat`, [160](#)

Symbols

- `__add__()` (*parselmouth.Harmonicity* method), 54
- `__add__()` (*parselmouth.Intensity* method), 63
- `__add__()` (*parselmouth.Sound* method), 107
- `__add__()` (*parselmouth.Vector* method), 154
- `__copy__()` (*parselmouth.CC* method), 34
- `__copy__()` (*parselmouth.Data* method), 38
- `__copy__()` (*parselmouth.Formant* method), 43
- `__copy__()` (*parselmouth.Function* method), 48
- `__copy__()` (*parselmouth.Harmonicity* method), 54
- `__copy__()` (*parselmouth.Intensity* method), 63
- `__copy__()` (*parselmouth.MFCC* method), 71
- `__copy__()` (*parselmouth.Matrix* method), 78
- `__copy__()` (*parselmouth.Pitch* method), 86
- `__copy__()` (*parselmouth.Sampled* method), 92
- `__copy__()` (*parselmouth.SampledXY* method), 97
- `__copy__()` (*parselmouth.Sound* method), 107
- `__copy__()` (*parselmouth.Spectrogram* method), 123
- `__copy__()` (*parselmouth.Spectrum* method), 132
- `__copy__()` (*parselmouth.TextGrid* method), 138
- `__copy__()` (*parselmouth.TimeFrameSampled* method), 143
- `__copy__()` (*parselmouth.TimeFunction* method), 148
- `__copy__()` (*parselmouth.Vector* method), 154
- `__deepcopy__()` (*parselmouth.CC* method), 34
- `__deepcopy__()` (*parselmouth.Data* method), 38
- `__deepcopy__()` (*parselmouth.Formant* method), 43
- `__deepcopy__()` (*parselmouth.Function* method), 48
- `__deepcopy__()` (*parselmouth.Harmonicity* method), 54
- `__deepcopy__()` (*parselmouth.Intensity* method), 63
- `__deepcopy__()` (*parselmouth.MFCC* method), 71
- `__deepcopy__()` (*parselmouth.Matrix* method), 78
- `__deepcopy__()` (*parselmouth.Pitch* method), 86
- `__deepcopy__()` (*parselmouth.Sampled* method), 92
- `__deepcopy__()` (*parselmouth.SampledXY* method), 97
- `__deepcopy__()` (*parselmouth.Sound* method), 107
- `__deepcopy__()` (*parselmouth.Spectrogram* method), 123
- `__deepcopy__()` (*parselmouth.Spectrum* method), 132
- `__deepcopy__()` (*parselmouth.TextGrid* method), 138
- `__deepcopy__()` (*parselmouth.TimeFrameSampled* method), 143
- `__deepcopy__()` (*parselmouth.TimeFunction* method), 148
- `__deepcopy__()` (*parselmouth.Vector* method), 154
- `__eq__()` (*parselmouth.AmplitudeScaling* method), 30
- `__eq__()` (*parselmouth.CC* method), 34
- `__eq__()` (*parselmouth.CC.FileFormat* method), 34
- `__eq__()` (*parselmouth.Data* method), 38
- `__eq__()` (*parselmouth.Data.FileFormat* method), 38
- `__eq__()` (*parselmouth.Formant* method), 43
- `__eq__()` (*parselmouth.Formant.FileFormat* method), 42
- `__eq__()` (*parselmouth.FormantUnit* method), 46
- `__eq__()` (*parselmouth.Function* method), 48
- `__eq__()` (*parselmouth.Function.FileFormat* method), 47
- `__eq__()` (*parselmouth.Harmonicity* method), 54
- `__eq__()` (*parselmouth.Harmonicity.FileFormat* method), 53
- `__eq__()` (*parselmouth.Intensity* method), 63
- `__eq__()` (*parselmouth.Intensity.AveragingMethod* method), 62
- `__eq__()` (*parselmouth.Intensity.FileFormat* method), 62
- `__eq__()` (*parselmouth.MFCC* method), 71
- `__eq__()` (*parselmouth.MFCC.FileFormat* method), 71
- `__eq__()` (*parselmouth.Matrix* method), 78
- `__eq__()` (*parselmouth.Matrix.FileFormat* method), 77
- `__eq__()` (*parselmouth.Pitch* method), 86
- `__eq__()` (*parselmouth.Pitch.FileFormat* method), 85
- `__eq__()` (*parselmouth.PitchUnit* method), 90
- `__eq__()` (*parselmouth.Sampled* method), 93
- `__eq__()` (*parselmouth.Sampled.FileFormat* method), 92
- `__eq__()` (*parselmouth.SampledXY* method), 97
- `__eq__()` (*parselmouth.SampledXY.FileFormat* method), 96
- `__eq__()` (*parselmouth.SignalOutsideTimeDomain* method), 99
- `__eq__()` (*parselmouth.Sound* method), 107
- `__eq__()` (*parselmouth.Sound.FileFormat* method), 106
- `__eq__()` (*parselmouth.Sound.ToHarmonicityMethod* method), 143

`method`), 106
`__eq__()` (`parselmouth.Sound.ToPitchMethod` `method`), 107
`__eq__()` (`parselmouth.SoundFileFormat` `method`), 116
`__eq__()` (`parselmouth.SpectralAnalysisWindowShape` `method`), 118
`__eq__()` (`parselmouth.Spectrogram` `method`), 123
`__eq__()` (`parselmouth.Spectrogram.FileFormat` `method`), 122
`__eq__()` (`parselmouth.Spectrum` `method`), 132
`__eq__()` (`parselmouth.Spectrum.FileFormat` `method`), 132
`__eq__()` (`parselmouth.TextGrid` `method`), 138
`__eq__()` (`parselmouth.TextGrid.FileFormat` `method`), 137
`__eq__()` (`parselmouth.TimeFrameSampled` `method`), 143
`__eq__()` (`parselmouth.TimeFrameSampled.FileFormat` `method`), 142
`__eq__()` (`parselmouth.TimeFunction` `method`), 148
`__eq__()` (`parselmouth.TimeFunction.FileFormat` `method`), 147
`__eq__()` (`parselmouth.ValueInterpolation` `method`), 150
`__eq__()` (`parselmouth.Vector` `method`), 154
`__eq__()` (`parselmouth.Vector.FileFormat` `method`), 153
`__eq__()` (`parselmouth.WindowShape` `method`), 158
`__getitem__()` (`parselmouth.CC` `method`), 34
`__getitem__()` (`parselmouth.CC.Frame` `method`), 34
`__getitem__()` (`parselmouth.MFCC` `method`), 71
`__getitem__()` (`parselmouth.MFCC.Frame` `method`), 71
`__getitem__()` (`parselmouth.Pitch` `method`), 86
`__getitem__()` (`parselmouth.Pitch.Frame` `method`), 85
`__getitem__()` (`parselmouth.Spectrum` `method`), 132
`__hash__` (`parselmouth.CC` `attribute`), 36
`__hash__` (`parselmouth.Data` `attribute`), 39
`__hash__` (`parselmouth.Formant` `attribute`), 45
`__hash__` (`parselmouth.Function` `attribute`), 49
`__hash__` (`parselmouth.Harmonicity` `attribute`), 56
`__hash__` (`parselmouth.Intensity` `attribute`), 66
`__hash__` (`parselmouth.MFCC` `attribute`), 73
`__hash__` (`parselmouth.Matrix` `attribute`), 80
`__hash__` (`parselmouth.Pitch` `attribute`), 88
`__hash__` (`parselmouth.Sampled` `attribute`), 93
`__hash__` (`parselmouth.SampledXY` `attribute`), 98
`__hash__` (`parselmouth.Sound` `attribute`), 114
`__hash__` (`parselmouth.Spectrogram` `attribute`), 125
`__hash__` (`parselmouth.Spectrum` `attribute`), 135
`__hash__` (`parselmouth.TextGrid` `attribute`), 139
`__hash__` (`parselmouth.TimeFrameSampled` `attribute`), 144
`__hash__` (`parselmouth.TimeFunction` `attribute`), 149
`__hash__` (`parselmouth.Vector` `attribute`), 156
`__hash__()` (`parselmouth.AmplitudeScaling` `method`), 30
`__hash__()` (`parselmouth.CC.FileFormat` `method`), 34
`__hash__()` (`parselmouth.Data.FileFormat` `method`), 38
`__hash__()` (`parselmouth.Formant.FileFormat` `method`), 43
`__hash__()` (`parselmouth.FormantUnit` `method`), 46
`__hash__()` (`parselmouth.Function.FileFormat` `method`), 48
`__hash__()` (`parselmouth.Harmonicity.FileFormat` `method`), 53
`__hash__()` (`parselmouth.Intensity.AveragingMethod` `method`), 62
`__hash__()` (`parselmouth.Intensity.FileFormat` `method`), 62
`__hash__()` (`parselmouth.MFCC.FileFormat` `method`), 71
`__hash__()` (`parselmouth.Matrix.FileFormat` `method`), 77
`__hash__()` (`parselmouth.Pitch.FileFormat` `method`), 85
`__hash__()` (`parselmouth.PitchUnit` `method`), 90
`__hash__()` (`parselmouth.Sampled.FileFormat` `method`), 92
`__hash__()` (`parselmouth.SampledXY.FileFormat` `method`), 96
`__hash__()` (`parselmouth.SignalOutsideTimeDomain` `method`), 99
`__hash__()` (`parselmouth.Sound.FileFormat` `method`), 106
`__hash__()` (`parselmouth.Sound.ToHarmonicityMethod` `method`), 106
`__hash__()` (`parselmouth.Sound.ToPitchMethod` `method`), 107
`__hash__()` (`parselmouth.SoundFileFormat` `method`), 116
`__hash__()` (`parselmouth.SpectralAnalysisWindowShape` `method`), 118
`__hash__()` (`parselmouth.Spectrogram.FileFormat` `method`), 123
`__hash__()` (`parselmouth.Spectrum.FileFormat` `method`), 132
`__hash__()` (`parselmouth.TextGrid.FileFormat` `method`), 137
`__hash__()` (`parselmouth.TimeFrameSampled.FileFormat` `method`), 143
`__hash__()` (`parselmouth.TimeFunction.FileFormat` `method`), 147
`__hash__()` (`parselmouth.ValueInterpolation` `method`), 150
`__hash__()` (`parselmouth.Vector.FileFormat` `method`), 153
`__hash__()` (`parselmouth.WindowShape` `method`), 158
`__iadd__()` (`parselmouth.Harmonicity` `method`), 54
`__iadd__()` (`parselmouth.Intensity` `method`), 63

__iadd__() (parselmouth.Sound method), 107
 __iadd__() (parselmouth.Vector method), 154
 __imul__() (parselmouth.Harmonicity method), 54
 __imul__() (parselmouth.Intensity method), 63
 __imul__() (parselmouth.Sound method), 107
 __imul__() (parselmouth.Vector method), 154
 __index__() (parselmouth.AmplitudeScaling method), 30
 __index__() (parselmouth.CC.FileFormat method), 34
 __index__() (parselmouth.Data.FileFormat method), 38
 __index__() (parselmouth.Formant.FileFormat method), 43
 __index__() (parselmouth.FormantUnit method), 46
 __index__() (parselmouth.Function.FileFormat method), 48
 __index__() (parselmouth.Harmonicity.FileFormat method), 53
 __index__() (parselmouth.Intensity.AveragingMethod method), 62
 __index__() (parselmouth.Intensity.FileFormat method), 62
 __index__() (parselmouth.MFCC.FileFormat method), 71
 __index__() (parselmouth.Matrix.FileFormat method), 77
 __index__() (parselmouth.Pitch.FileFormat method), 85
 __index__() (parselmouth.PitchUnit method), 90
 __index__() (parselmouth.Sampled.FileFormat method), 92
 __index__() (parselmouth.SampledXY.FileFormat method), 96
 __index__() (parselmouth.SignalOutsideTimeDomain method), 99
 __index__() (parselmouth.Sound.FileFormat method), 106
 __index__() (parselmouth.Sound.ToHarmonicityMethod method), 106
 __index__() (parselmouth.Sound.ToPitchMethod method), 107
 __index__() (parselmouth.SoundFileFormat method), 116
 __index__() (parselmouth.SpectralAnalysisWindowShape method), 118
 __index__() (parselmouth.Spectrogram.FileFormat method), 123
 __index__() (parselmouth.Spectrum.FileFormat method), 132
 __index__() (parselmouth.TextGrid.FileFormat method), 138
 __index__() (parselmouth.TimeFrameSampled.FileFormat method), 143
 __index__() (parselmouth.TimeFunction.FileFormat method), 147
 __index__() (parselmouth.ValueInterpolation method), 150
 __index__() (parselmouth.Vector.FileFormat method), 153
 __index__() (parselmouth.WindowShape method), 158
 __init__() (parselmouth.AmplitudeScaling method), 30
 __init__() (parselmouth.CC method), 34
 __init__() (parselmouth.CC.FileFormat method), 34
 __init__() (parselmouth.CC.Frame method), 34
 __init__() (parselmouth.Data method), 38
 __init__() (parselmouth.Data.FileFormat method), 38
 __init__() (parselmouth.Formant method), 43
 __init__() (parselmouth.Formant.FileFormat method), 43
 __init__() (parselmouth.FormantUnit method), 46
 __init__() (parselmouth.Function method), 48
 __init__() (parselmouth.Function.FileFormat method), 48
 __init__() (parselmouth.Harmonicity method), 54
 __init__() (parselmouth.Harmonicity.FileFormat method), 53
 __init__() (parselmouth.Intensity method), 63
 __init__() (parselmouth.Intensity.AveragingMethod method), 62
 __init__() (parselmouth.Intensity.FileFormat method), 62
 __init__() (parselmouth.MFCC method), 71
 __init__() (parselmouth.MFCC.FileFormat method), 71
 __init__() (parselmouth.MFCC.Frame method), 71
 __init__() (parselmouth.Matrix method), 78
 __init__() (parselmouth.Matrix.FileFormat method), 77
 __init__() (parselmouth.Pitch method), 86
 __init__() (parselmouth.Pitch.Candidate method), 85
 __init__() (parselmouth.Pitch.FileFormat method), 85
 __init__() (parselmouth.Pitch.Frame method), 85
 __init__() (parselmouth.PitchUnit method), 90
 __init__() (parselmouth.Sampled method), 93
 __init__() (parselmouth.Sampled.FileFormat method), 92
 __init__() (parselmouth.SampledXY method), 97
 __init__() (parselmouth.SampledXY.FileFormat method), 96
 __init__() (parselmouth.SignalOutsideTimeDomain method), 99
 __init__() (parselmouth.Sound method), 108
 __init__() (parselmouth.Sound.FileFormat method), 106
 __init__() (parselmouth.Sound.ToHarmonicityMethod method), 106
 __init__() (parselmouth.Sound.ToPitchMethod

method), 107

`__init__()` (*parselmouth.SoundFileFormat* method), 116

`__init__()` (*parselmouth.SpectralAnalysisWindowShape* method), 118

`__init__()` (*parselmouth.Spectrogram* method), 123

`__init__()` (*parselmouth.Spectrogram.FileFormat* method), 123

`__init__()` (*parselmouth.Spectrum* method), 132

`__init__()` (*parselmouth.Spectrum.FileFormat* method), 132

`__init__()` (*parselmouth.TextGrid* method), 138

`__init__()` (*parselmouth.TextGrid.FileFormat* method), 138

`__init__()` (*parselmouth.Thing* method), 140

`__init__()` (*parselmouth.TimeFrameSampled* method), 143

`__init__()` (*parselmouth.TimeFrameSampled.FileFormat* method), 143

`__init__()` (*parselmouth.TimeFunction* method), 148

`__init__()` (*parselmouth.TimeFunction.FileFormat* method), 147

`__init__()` (*parselmouth.ValueInterpolation* method), 150

`__init__()` (*parselmouth.Vector* method), 154

`__init__()` (*parselmouth.Vector.FileFormat* method), 153

`__init__()` (*parselmouth.WindowShape* method), 158

`__int__()` (*parselmouth.AmplitudeScaling* method), 30

`__int__()` (*parselmouth.CC.FileFormat* method), 34

`__int__()` (*parselmouth.Data.FileFormat* method), 38

`__int__()` (*parselmouth.Formant.FileFormat* method), 43

`__int__()` (*parselmouth.FormantUnit* method), 46

`__int__()` (*parselmouth.Function.FileFormat* method), 48

`__int__()` (*parselmouth.Harmonicity.FileFormat* method), 53

`__int__()` (*parselmouth.Intensity.AveragingMethod* method), 62

`__int__()` (*parselmouth.Intensity.FileFormat* method), 62

`__int__()` (*parselmouth.MFCC.FileFormat* method), 71

`__int__()` (*parselmouth.Matrix.FileFormat* method), 77

`__int__()` (*parselmouth.Pitch.FileFormat* method), 85

`__int__()` (*parselmouth.PitchUnit* method), 90

`__int__()` (*parselmouth.Sampled.FileFormat* method), 92

`__int__()` (*parselmouth.SampledXY.FileFormat* method), 96

`__int__()` (*parselmouth.SignalOutsideTimeDomain* method), 99

`__int__()` (*parselmouth.Sound.FileFormat* method), 106

`__int__()` (*parselmouth.Sound.ToHarmonicityMethod* method), 106

`__int__()` (*parselmouth.Sound.ToPitchMethod* method), 107

`__int__()` (*parselmouth.SoundFileFormat* method), 116

`__int__()` (*parselmouth.SpectralAnalysisWindowShape* method), 118

`__int__()` (*parselmouth.Spectrogram.FileFormat* method), 123

`__int__()` (*parselmouth.Spectrum.FileFormat* method), 132

`__int__()` (*parselmouth.TextGrid.FileFormat* method), 138

`__int__()` (*parselmouth.TimeFrameSampled.FileFormat* method), 143

`__int__()` (*parselmouth.TimeFunction.FileFormat* method), 147

`__int__()` (*parselmouth.ValueInterpolation* method), 150

`__int__()` (*parselmouth.Vector.FileFormat* method), 153

`__int__()` (*parselmouth.WindowShape* method), 158

`__isub__()` (*parselmouth.Harmonicity* method), 54

`__isub__()` (*parselmouth.Intensity* method), 63

`__isub__()` (*parselmouth.Sound* method), 108

`__isub__()` (*parselmouth.Vector* method), 154

`__iter__()` (*parselmouth.CC* method), 35

`__iter__()` (*parselmouth.MFCC* method), 72

`__iter__()` (*parselmouth.Pitch* method), 86

`__itruediv__()` (*parselmouth.Harmonicity* method), 54

`__itruediv__()` (*parselmouth.Intensity* method), 63

`__itruediv__()` (*parselmouth.Sound* method), 108

`__itruediv__()` (*parselmouth.Vector* method), 154

`__len__()` (*parselmouth.CC* method), 35

`__len__()` (*parselmouth.CC.Frame* method), 34

`__len__()` (*parselmouth.Formant* method), 43

`__len__()` (*parselmouth.Harmonicity* method), 54

`__len__()` (*parselmouth.Intensity* method), 63

`__len__()` (*parselmouth.MFCC* method), 72

`__len__()` (*parselmouth.MFCC.Frame* method), 71

`__len__()` (*parselmouth.Matrix* method), 78

`__len__()` (*parselmouth.Pitch* method), 86

`__len__()` (*parselmouth.Pitch.Frame* method), 85

`__len__()` (*parselmouth.Sampled* method), 93

`__len__()` (*parselmouth.SampledXY* method), 97

`__len__()` (*parselmouth.Sound* method), 108

`__len__()` (*parselmouth.Spectrogram* method), 123

`__len__()` (*parselmouth.Spectrum* method), 132

`__len__()` (*parselmouth.TimeFrameSampled* method), 143

`__len__()` (*parselmouth.Vector* method), 154

`__mul__()` (*parselmouth.Harmonicity* method), 54

`__mul__()` (*parselmouth.Intensity* method), 63

- `__mul__()` (*parselmouth.Sound* method), 108
- `__mul__()` (*parselmouth.Vector* method), 154
- `__ne__()` (*parselmouth.AmplitudeScaling* method), 30
- `__ne__()` (*parselmouth.CC* method), 35
- `__ne__()` (*parselmouth.CC.FileFormat* method), 34
- `__ne__()` (*parselmouth.Data* method), 38
- `__ne__()` (*parselmouth.Data.FileFormat* method), 38
- `__ne__()` (*parselmouth.Formant* method), 43
- `__ne__()` (*parselmouth.Formant.FileFormat* method), 43
- `__ne__()` (*parselmouth.FormantUnit* method), 46
- `__ne__()` (*parselmouth.Function* method), 48
- `__ne__()` (*parselmouth.Function.FileFormat* method), 48
- `__ne__()` (*parselmouth.Harmonicity* method), 54
- `__ne__()` (*parselmouth.Harmonicity.FileFormat* method), 53
- `__ne__()` (*parselmouth.Intensity* method), 63
- `__ne__()` (*parselmouth.Intensity.AveragingMethod* method), 62
- `__ne__()` (*parselmouth.Intensity.FileFormat* method), 62
- `__ne__()` (*parselmouth.MFCC* method), 72
- `__ne__()` (*parselmouth.MFCC.FileFormat* method), 71
- `__ne__()` (*parselmouth.Matrix* method), 78
- `__ne__()` (*parselmouth.Matrix.FileFormat* method), 77
- `__ne__()` (*parselmouth.Pitch* method), 86
- `__ne__()` (*parselmouth.Pitch.FileFormat* method), 85
- `__ne__()` (*parselmouth.PitchUnit* method), 90
- `__ne__()` (*parselmouth.Sampled* method), 93
- `__ne__()` (*parselmouth.Sampled.FileFormat* method), 92
- `__ne__()` (*parselmouth.SampledXY* method), 97
- `__ne__()` (*parselmouth.SampledXY.FileFormat* method), 96
- `__ne__()` (*parselmouth.SignalOutsideTimeDomain* method), 99
- `__ne__()` (*parselmouth.Sound* method), 108
- `__ne__()` (*parselmouth.Sound.FileFormat* method), 106
- `__ne__()` (*parselmouth.Sound.ToHarmonicityMethod* method), 106
- `__ne__()` (*parselmouth.Sound.ToPitchMethod* method), 107
- `__ne__()` (*parselmouth.SoundFileFormat* method), 116
- `__ne__()` (*parselmouth.SpectralAnalysisWindowShape* method), 118
- `__ne__()` (*parselmouth.Spectrogram* method), 123
- `__ne__()` (*parselmouth.Spectrogram.FileFormat* method), 123
- `__ne__()` (*parselmouth.Spectrum* method), 132
- `__ne__()` (*parselmouth.Spectrum.FileFormat* method), 132
- `__ne__()` (*parselmouth.TextGrid* method), 138
- `__ne__()` (*parselmouth.TextGrid.FileFormat* method), 138
- `__ne__()` (*parselmouth.TimeFrameSampled* method), 143
- `__ne__()` (*parselmouth.TimeFrameSampled.FileFormat* method), 143
- `__ne__()` (*parselmouth.TimeFunction* method), 148
- `__ne__()` (*parselmouth.TimeFunction.FileFormat* method), 147
- `__ne__()` (*parselmouth.ValueInterpolation* method), 150
- `__ne__()` (*parselmouth.Vector* method), 154
- `__ne__()` (*parselmouth.Vector.FileFormat* method), 153
- `__ne__()` (*parselmouth.WindowShape* method), 158
- `__new__()` (*parselmouth.AmplitudeScaling* method), 30
- `__new__()` (*parselmouth.CC* method), 35
- `__new__()` (*parselmouth.CC.FileFormat* method), 34
- `__new__()` (*parselmouth.CC.Frame* method), 34
- `__new__()` (*parselmouth.Data* method), 38
- `__new__()` (*parselmouth.Data.FileFormat* method), 38
- `__new__()` (*parselmouth.Formant* method), 43
- `__new__()` (*parselmouth.Formant.FileFormat* method), 43
- `__new__()` (*parselmouth.FormantUnit* method), 46
- `__new__()` (*parselmouth.Function* method), 48
- `__new__()` (*parselmouth.Function.FileFormat* method), 48
- `__new__()` (*parselmouth.Harmonicity* method), 54
- `__new__()` (*parselmouth.Harmonicity.FileFormat* method), 53
- `__new__()` (*parselmouth.Intensity* method), 63
- `__new__()` (*parselmouth.Intensity.AveragingMethod* method), 62
- `__new__()` (*parselmouth.Intensity.FileFormat* method), 62
- `__new__()` (*parselmouth.MFCC* method), 72
- `__new__()` (*parselmouth.MFCC.FileFormat* method), 71
- `__new__()` (*parselmouth.MFCC.Frame* method), 71
- `__new__()` (*parselmouth.Matrix* method), 78
- `__new__()` (*parselmouth.Matrix.FileFormat* method), 78
- `__new__()` (*parselmouth.Pitch* method), 86
- `__new__()` (*parselmouth.Pitch.Candidate* method), 85
- `__new__()` (*parselmouth.Pitch.FileFormat* method), 85
- `__new__()` (*parselmouth.Pitch.Frame* method), 85
- `__new__()` (*parselmouth.PitchUnit* method), 90
- `__new__()` (*parselmouth.Sampled* method), 93
- `__new__()` (*parselmouth.Sampled.FileFormat* method), 92
- `__new__()` (*parselmouth.SampledXY* method), 97
- `__new__()` (*parselmouth.SampledXY.FileFormat* method), 96
- `__new__()` (*parselmouth.SignalOutsideTimeDomain* method), 99
- `__new__()` (*parselmouth.Sound* method), 108

`__new__()` (*parselmouth.Sound.FileFormat* method), 106
`__new__()` (*parselmouth.Sound.ToHarmonicityMethod* method), 106
`__new__()` (*parselmouth.Sound.ToPitchMethod* method), 107
`__new__()` (*parselmouth.SoundFileFormat* method), 117
`__new__()` (*parselmouth.SpectralAnalysisWindowShape* method), 118
`__new__()` (*parselmouth.Spectrogram* method), 123
`__new__()` (*parselmouth.Spectrogram.FileFormat* method), 123
`__new__()` (*parselmouth.Spectrum* method), 132
`__new__()` (*parselmouth.Spectrum.FileFormat* method), 132
`__new__()` (*parselmouth.TextGrid* method), 138
`__new__()` (*parselmouth.TextGrid.FileFormat* method), 138
`__new__()` (*parselmouth.Thing* method), 140
`__new__()` (*parselmouth.TimeFrameSampled* method), 143
`__new__()` (*parselmouth.TimeFrameSampled.FileFormat* method), 143
`__new__()` (*parselmouth.TimeFunction* method), 148
`__new__()` (*parselmouth.TimeFunction.FileFormat* method), 147
`__new__()` (*parselmouth.ValueInterpolation* method), 150
`__new__()` (*parselmouth.Vector* method), 154
`__new__()` (*parselmouth.Vector.FileFormat* method), 154
`__new__()` (*parselmouth.WindowShape* method), 158
`__radd__()` (*parselmouth.Harmonicity* method), 54
`__radd__()` (*parselmouth.Intensity* method), 63
`__radd__()` (*parselmouth.Sound* method), 108
`__radd__()` (*parselmouth.Vector* method), 154
`__repr__()` (*parselmouth.AmplitudeScaling* method), 30
`__repr__()` (*parselmouth.CC.FileFormat* method), 34
`__repr__()` (*parselmouth.Data.FileFormat* method), 38
`__repr__()` (*parselmouth.Formant.FileFormat* method), 43
`__repr__()` (*parselmouth.FormantUnit* method), 46
`__repr__()` (*parselmouth.Function.FileFormat* method), 48
`__repr__()` (*parselmouth.Harmonicity.FileFormat* method), 53
`__repr__()` (*parselmouth.Intensity.AveragingMethod* method), 62
`__repr__()` (*parselmouth.Intensity.FileFormat* method), 62
`__repr__()` (*parselmouth.MFCC.FileFormat* method), 71
`__repr__()` (*parselmouth.Matrix.FileFormat* method), 78
`__repr__()` (*parselmouth.Pitch.FileFormat* method), 85
`__repr__()` (*parselmouth.PitchUnit* method), 90
`__repr__()` (*parselmouth.Sampled.FileFormat* method), 92
`__repr__()` (*parselmouth.SampledXY.FileFormat* method), 96
`__repr__()` (*parselmouth.SignalOutsideTimeDomain* method), 99
`__repr__()` (*parselmouth.Sound.FileFormat* method), 106
`__repr__()` (*parselmouth.Sound.ToHarmonicityMethod* method), 106
`__repr__()` (*parselmouth.Sound.ToPitchMethod* method), 107
`__repr__()` (*parselmouth.SoundFileFormat* method), 117
`__repr__()` (*parselmouth.SpectralAnalysisWindowShape* method), 118
`__repr__()` (*parselmouth.Spectrogram.FileFormat* method), 123
`__repr__()` (*parselmouth.Spectrum.FileFormat* method), 132
`__repr__()` (*parselmouth.TextGrid.FileFormat* method), 138
`__repr__()` (*parselmouth.TimeFrameSampled.FileFormat* method), 143
`__repr__()` (*parselmouth.TimeFunction.FileFormat* method), 147
`__repr__()` (*parselmouth.ValueInterpolation* method), 150
`__repr__()` (*parselmouth.Vector.FileFormat* method), 154
`__repr__()` (*parselmouth.WindowShape* method), 158
`__rmul__()` (*parselmouth.Harmonicity* method), 54
`__rmul__()` (*parselmouth.Intensity* method), 63
`__rmul__()` (*parselmouth.Sound* method), 108
`__rmul__()` (*parselmouth.Vector* method), 154
`__setitem__()` (*parselmouth.CC* method), 35
`__setitem__()` (*parselmouth.CC.Frame* method), 34
`__setitem__()` (*parselmouth.MFCC* method), 72
`__setitem__()` (*parselmouth.MFCC.Frame* method), 71
`__setitem__()` (*parselmouth.Spectrum* method), 132
`__str__()` (*parselmouth.AmplitudeScaling* method), 30
`__str__()` (*parselmouth.CC* method), 35
`__str__()` (*parselmouth.CC.FileFormat* method), 34
`__str__()` (*parselmouth.Data* method), 38
`__str__()` (*parselmouth.Data.FileFormat* method), 38
`__str__()` (*parselmouth.Formant* method), 43
`__str__()` (*parselmouth.Formant.FileFormat* method), 43
`__str__()` (*parselmouth.FormantUnit* method), 46
`__str__()` (*parselmouth.Function* method), 48

__str__() (parselmouth.Function.FileFormat method), 48
 __str__() (parselmouth.Harmonicity method), 54
 __str__() (parselmouth.Harmonicity.FileFormat method), 53
 __str__() (parselmouth.Intensity method), 63
 __str__() (parselmouth.Intensity.AveragingMethod method), 62
 __str__() (parselmouth.Intensity.FileFormat method), 63
 __str__() (parselmouth.MFCC method), 72
 __str__() (parselmouth.MFCC.FileFormat method), 71
 __str__() (parselmouth.Matrix method), 78
 __str__() (parselmouth.Matrix.FileFormat method), 78
 __str__() (parselmouth.Pitch method), 86
 __str__() (parselmouth.Pitch.FileFormat method), 85
 __str__() (parselmouth.PitchUnit method), 90
 __str__() (parselmouth.Sampled method), 93
 __str__() (parselmouth.Sampled.FileFormat method), 92
 __str__() (parselmouth.SampledXY method), 97
 __str__() (parselmouth.SampledXY.FileFormat method), 96
 __str__() (parselmouth.SignalOutsideTimeDomain method), 99
 __str__() (parselmouth.Sound method), 108
 __str__() (parselmouth.Sound.FileFormat method), 106
 __str__() (parselmouth.Sound.ToHarmonicityMethod method), 107
 __str__() (parselmouth.Sound.ToPitchMethod method), 107
 __str__() (parselmouth.SoundFileFormat method), 117
 __str__() (parselmouth.SpectralAnalysisWindowShape method), 118
 __str__() (parselmouth.Spectrogram method), 123
 __str__() (parselmouth.Spectrogram.FileFormat method), 123
 __str__() (parselmouth.Spectrum method), 132
 __str__() (parselmouth.Spectrum.FileFormat method), 132
 __str__() (parselmouth.TextGrid method), 138
 __str__() (parselmouth.TextGrid.FileFormat method), 138
 __str__() (parselmouth.Thing method), 140
 __str__() (parselmouth.TimeFrameSampled method), 143
 __str__() (parselmouth.TimeFrameSampled.FileFormat method), 143
 __str__() (parselmouth.TimeFunction method), 148
 __str__() (parselmouth.TimeFunction.FileFormat method), 147
 __str__() (parselmouth.ValueInterpolation method), 150

__str__() (parselmouth.Vector method), 154
 __str__() (parselmouth.Vector.FileFormat method), 154
 __str__() (parselmouth.WindowShape method), 158
 __sub__() (parselmouth.Harmonicity method), 54
 __sub__() (parselmouth.Intensity method), 63
 __sub__() (parselmouth.Sound method), 108
 __sub__() (parselmouth.Vector method), 154
 __truediv__() (parselmouth.Harmonicity method), 54
 __truediv__() (parselmouth.Intensity method), 63
 __truediv__() (parselmouth.Sound method), 108
 __truediv__() (parselmouth.Vector method), 154

A

AC (parselmouth.Sound.ToHarmonicityMethod attribute), 107
 AC (parselmouth.Sound.ToPitchMethod attribute), 107
 add() (parselmouth.Harmonicity method), 54
 add() (parselmouth.Intensity method), 63
 add() (parselmouth.Sound method), 108
 add() (parselmouth.Vector method), 154
 AIFC (parselmouth.SoundFileFormat attribute), 117
 AIFF (parselmouth.SoundFileFormat attribute), 117
 AmplitudeScaling (class in parselmouth), 30
 as_array() (parselmouth.Harmonicity method), 54
 as_array() (parselmouth.Intensity method), 63
 as_array() (parselmouth.Matrix method), 78
 as_array() (parselmouth.Pitch.Frame method), 85
 as_array() (parselmouth.Sound method), 108
 as_array() (parselmouth.Spectrogram method), 123
 as_array() (parselmouth.Spectrum method), 132
 as_array() (parselmouth.Vector method), 154
 at_xy() (parselmouth.Harmonicity method), 54
 at_xy() (parselmouth.Intensity method), 63
 at_xy() (parselmouth.Matrix method), 78
 at_xy() (parselmouth.Sound method), 108
 at_xy() (parselmouth.Spectrogram method), 123
 at_xy() (parselmouth.Spectrum method), 132
 at_xy() (parselmouth.Vector method), 154
 autocorrelate() (parselmouth.Sound method), 108

B

BARK (parselmouth.FormantUnit attribute), 46
 BARTLETT (parselmouth.SpectralAnalysisWindowShape attribute), 118
 bin_width (parselmouth.Spectrum property), 135
 BINARY (parselmouth.CC.FileFormat attribute), 34
 BINARY (parselmouth.Data.FileFormat attribute), 38
 BINARY (parselmouth.Formant.FileFormat attribute), 43
 BINARY (parselmouth.Function.FileFormat attribute), 48
 BINARY (parselmouth.Harmonicity.FileFormat attribute), 53
 BINARY (parselmouth.Intensity.FileFormat attribute), 63
 BINARY (parselmouth.Matrix.FileFormat attribute), 78

BINARY (*parselmouth.MFCC.FileFormat* attribute), 71
 BINARY (*parselmouth.Pitch.FileFormat* attribute), 85
 BINARY (*parselmouth.Sampled.FileFormat* attribute), 92
 BINARY (*parselmouth.SampledXY.FileFormat* attribute), 97
 BINARY (*parselmouth.Sound.FileFormat* attribute), 106
 BINARY (*parselmouth.Spectrogram.FileFormat* attribute), 123
 BINARY (*parselmouth.Spectrum.FileFormat* attribute), 132
 BINARY (*parselmouth.TextGrid.FileFormat* attribute), 138
 BINARY (*parselmouth.TimeFrameSampled.FileFormat* attribute), 143
 BINARY (*parselmouth.TimeFunction.FileFormat* attribute), 148
 BINARY (*parselmouth.Vector.FileFormat* attribute), 154

C

c (*parselmouth.CC.Frame* property), 34
c (*parselmouth.MFCC.Frame* property), 71
c0 (*parselmouth.CC.Frame* property), 34
c0 (*parselmouth.MFCC.Frame* property), 71
call() (in module *parselmouth.praat*), 160
candidates (*parselmouth.Pitch.Frame* property), 86
CC (class in *parselmouth*), 31
CC (*parselmouth.Sound.ToHarmonicityMethod* attribute), 107
CC (*parselmouth.Sound.ToPitchMethod* attribute), 107
CC.FileFormat (class in *parselmouth*), 34
CC.Frame (class in *parselmouth*), 34
ceiling (*parselmouth.Pitch* property), 88
centre_time (*parselmouth.CC* property), 36
centre_time (*parselmouth.Formant* property), 45
centre_time (*parselmouth.Harmonicity* property), 57
centre_time (*parselmouth.Intensity* property), 66
centre_time (*parselmouth.MFCC* property), 73
centre_time (*parselmouth.Pitch* property), 88
centre_time (*parselmouth.Sound* property), 114
centre_time (*parselmouth.Spectrogram* property), 126
centre_time (*parselmouth.TimeFrameSampled* property), 144
centre_time (*parselmouth.TimeFunction* property), 149
cepstral_smoothing() (*parselmouth.Spectrum* method), 132
class_name (*parselmouth.CC* property), 36
class_name (*parselmouth.Data* property), 39
class_name (*parselmouth.Formant* property), 45
class_name (*parselmouth.Function* property), 49
class_name (*parselmouth.Harmonicity* property), 57
class_name (*parselmouth.Intensity* property), 66
class_name (*parselmouth.Matrix* property), 80
class_name (*parselmouth.MFCC* property), 74
class_name (*parselmouth.Pitch* property), 88

class_name (*parselmouth.Sampled* property), 93
class_name (*parselmouth.SampledXY* property), 98
class_name (*parselmouth.Sound* property), 114
class_name (*parselmouth.Spectrogram* property), 126
class_name (*parselmouth.Spectrum* property), 135
class_name (*parselmouth.TextGrid* property), 139
class_name (*parselmouth.Thing* property), 140
class_name (*parselmouth.TimeFrameSampled* property), 145
class_name (*parselmouth.TimeFunction* property), 149
class_name (*parselmouth.Vector* property), 156
combine_to_stereo() (*parselmouth.Sound* static method), 109
concatenate() (*parselmouth.Sound* static method), 109
convert_to_mono() (*parselmouth.Sound* method), 109
convert_to_stereo() (*parselmouth.Sound* method), 109
convolve() (*parselmouth.MFCC* method), 72
convolve() (*parselmouth.Sound* method), 109
copy() (*parselmouth.CC* method), 35
copy() (*parselmouth.Data* method), 38
copy() (*parselmouth.Formant* method), 43
copy() (*parselmouth.Function* method), 48
copy() (*parselmouth.Harmonicity* method), 54
copy() (*parselmouth.Intensity* method), 63
copy() (*parselmouth.Matrix* method), 78
copy() (*parselmouth.MFCC* method), 72
copy() (*parselmouth.Pitch* method), 86
copy() (*parselmouth.Sampled* method), 93
copy() (*parselmouth.SampledXY* method), 97
copy() (*parselmouth.Sound* method), 109
copy() (*parselmouth.Spectrogram* method), 123
copy() (*parselmouth.Spectrum* method), 133
copy() (*parselmouth.TextGrid* method), 138
copy() (*parselmouth.TimeFrameSampled* method), 143
copy() (*parselmouth.TimeFunction* method), 148
copy() (*parselmouth.Vector* method), 154
count_differences() (*parselmouth.Pitch* method), 86
count_voiced_frames() (*parselmouth.Pitch* method), 86
cross_correlate() (*parselmouth.MFCC* method), 72
cross_correlate() (*parselmouth.Sound* method), 109
CUBIC (*parselmouth.ValueInterpolation* attribute), 150

D

Data (class in *parselmouth*), 37
Data.FileFormat (class in *parselmouth*), 38
DB (*parselmouth.Intensity.AveragingMethod* attribute), 62
de_emphasize() (*parselmouth.Sound* method), 109
deepen_band_modulation() (*parselmouth.Sound* method), 109
df (*parselmouth.Spectrum* property), 135
divide() (*parselmouth.Harmonicity* method), 54
divide() (*parselmouth.Intensity* method), 63

[divide\(\)](#) (*parselmouth.Sound* method), 109
[divide\(\)](#) (*parselmouth.Vector* method), 155
[dt](#) (*parselmouth.CC* property), 36
[dt](#) (*parselmouth.Formant* property), 45
[dt](#) (*parselmouth.Harmonicity* property), 57
[dt](#) (*parselmouth.Intensity* property), 66
[dt](#) (*parselmouth.MFCC* property), 74
[dt](#) (*parselmouth.Pitch* property), 88
[dt](#) (*parselmouth.Sound* property), 114
[dt](#) (*parselmouth.Spectrogram* property), 126
[dt](#) (*parselmouth.TimeFrameSampled* property), 145
[duration](#) (*parselmouth.CC* property), 36
[duration](#) (*parselmouth.Formant* property), 45
[duration](#) (*parselmouth.Harmonicity* property), 57
[duration](#) (*parselmouth.Intensity* property), 66
[duration](#) (*parselmouth.MFCC* property), 74
[duration](#) (*parselmouth.Pitch* property), 88
[duration](#) (*parselmouth.Sound* property), 114
[duration](#) (*parselmouth.Spectrogram* property), 126
[duration](#) (*parselmouth.TimeFrameSampled* property), 145
[duration](#) (*parselmouth.TimeFunction* property), 149
[dx](#) (*parselmouth.CC* property), 36
[dx](#) (*parselmouth.Formant* property), 45
[dx](#) (*parselmouth.Harmonicity* property), 57
[dx](#) (*parselmouth.Intensity* property), 66
[dx](#) (*parselmouth.Matrix* property), 80
[dx](#) (*parselmouth.MFCC* property), 74
[dx](#) (*parselmouth.Pitch* property), 88
[dx](#) (*parselmouth.Sampled* property), 93
[dx](#) (*parselmouth.SampledXY* property), 98
[dx](#) (*parselmouth.Sound* property), 114
[dx](#) (*parselmouth.Spectrogram* property), 126
[dx](#) (*parselmouth.Spectrum* property), 135
[dx](#) (*parselmouth.TimeFrameSampled* property), 145
[dx](#) (*parselmouth.Vector* property), 156
[dy](#) (*parselmouth.Harmonicity* property), 57
[dy](#) (*parselmouth.Intensity* property), 66
[dy](#) (*parselmouth.Matrix* property), 80
[dy](#) (*parselmouth.SampledXY* property), 98
[dy](#) (*parselmouth.Sound* property), 114
[dy](#) (*parselmouth.Spectrogram* property), 126
[dy](#) (*parselmouth.Spectrum* property), 135
[dy](#) (*parselmouth.Vector* property), 156

E

[end_time](#) (*parselmouth.CC* property), 36
[end_time](#) (*parselmouth.Formant* property), 45
[end_time](#) (*parselmouth.Harmonicity* property), 57
[end_time](#) (*parselmouth.Intensity* property), 66
[end_time](#) (*parselmouth.MFCC* property), 74
[end_time](#) (*parselmouth.Pitch* property), 89
[end_time](#) (*parselmouth.Sound* property), 114
[end_time](#) (*parselmouth.Spectrogram* property), 126

[end_time](#) (*parselmouth.TimeFrameSampled* property), 145
[end_time](#) (*parselmouth.TimeFunction* property), 149
[ENERGY](#) (*parselmouth.Intensity.AveragingMethod* attribute), 62
[ERB](#) (*parselmouth.PitchUnit* attribute), 90
[extract_all_channels\(\)](#) (*parselmouth.Sound* method), 109
[extract_channel\(\)](#) (*parselmouth.Sound* method), 109
[extract_features\(\)](#) (*parselmouth.MFCC* method), 72
[extract_left_channel\(\)](#) (*parselmouth.Sound* method), 109
[extract_part\(\)](#) (*parselmouth.Sound* method), 109
[extract_part_for_overlap\(\)](#) (*parselmouth.Sound* method), 109
[extract_right_channel\(\)](#) (*parselmouth.Sound* method), 109

F

[fifth_down\(\)](#) (*parselmouth.Pitch* method), 86
[fifth_up\(\)](#) (*parselmouth.Pitch* method), 86
[FLAC](#) (*parselmouth.SoundFileFormat* attribute), 117
[fmax](#) (*parselmouth.CC* property), 36
[fmax](#) (*parselmouth.MFCC* property), 74
[fmax](#) (*parselmouth.Spectrum* property), 135
[fmin](#) (*parselmouth.CC* property), 36
[fmin](#) (*parselmouth.MFCC* property), 74
[fmin](#) (*parselmouth.Spectrum* property), 135
[Formant](#) (class in *parselmouth*), 39
[Formant.FileFormat](#) (class in *parselmouth*), 42
[FormantUnit](#) (class in *parselmouth*), 46
[formula\(\)](#) (*parselmouth.Harmonicity* method), 54
[formula\(\)](#) (*parselmouth.Intensity* method), 64
[formula\(\)](#) (*parselmouth.Matrix* method), 78
[formula\(\)](#) (*parselmouth.Pitch* method), 86
[formula\(\)](#) (*parselmouth.Sound* method), 109
[formula\(\)](#) (*parselmouth.Spectrogram* method), 123
[formula\(\)](#) (*parselmouth.Spectrum* method), 133
[formula\(\)](#) (*parselmouth.Vector* method), 155
[frame_number_to_time\(\)](#) (*parselmouth.CC* method), 35
[frame_number_to_time\(\)](#) (*parselmouth.Formant* method), 43
[frame_number_to_time\(\)](#) (*parselmouth.Harmonicity* method), 54
[frame_number_to_time\(\)](#) (*parselmouth.Intensity* method), 64
[frame_number_to_time\(\)](#) (*parselmouth.MFCC* method), 72
[frame_number_to_time\(\)](#) (*parselmouth.Pitch* method), 86
[frame_number_to_time\(\)](#) (*parselmouth.Sound* method), 109

[frame_number_to_time\(\)](#) (*parselmouth.Spectrogram method*), 123
[frame_number_to_time\(\)](#) (*parselmouth.TimeFrameSampled method*), 143
[frequency](#) (*parselmouth.Pitch.Candidate property*), 85
[from_tgt\(\)](#) (*parselmouth.TextGrid static method*), 138
[full_name](#) (*parselmouth.CC property*), 36
[full_name](#) (*parselmouth.Data property*), 39
[full_name](#) (*parselmouth.Formant property*), 45
[full_name](#) (*parselmouth.Function property*), 49
[full_name](#) (*parselmouth.Harmonicity property*), 57
[full_name](#) (*parselmouth.Intensity property*), 66
[full_name](#) (*parselmouth.Matrix property*), 80
[full_name](#) (*parselmouth.MFCC property*), 74
[full_name](#) (*parselmouth.Pitch property*), 89
[full_name](#) (*parselmouth.Sampled property*), 94
[full_name](#) (*parselmouth.SampledXY property*), 98
[full_name](#) (*parselmouth.Sound property*), 114
[full_name](#) (*parselmouth.Spectrogram property*), 126
[full_name](#) (*parselmouth.Spectrum property*), 136
[full_name](#) (*parselmouth.TextGrid property*), 139
[full_name](#) (*parselmouth.Thing property*), 140
[full_name](#) (*parselmouth.TimeFrameSampled property*), 145
[full_name](#) (*parselmouth.TimeFunction property*), 149
[full_name](#) (*parselmouth.Vector property*), 156
[Function](#) (class in *parselmouth*), 47
[Function.FileFormat](#) (class in *parselmouth*), 47

G

[GAUSSIAN](#) (*parselmouth.SpectralAnalysisWindowShape attribute*), 118
[GAUSSIAN1](#) (*parselmouth.WindowShape attribute*), 158
[GAUSSIAN2](#) (*parselmouth.WindowShape attribute*), 158
[GAUSSIAN3](#) (*parselmouth.WindowShape attribute*), 158
[GAUSSIAN4](#) (*parselmouth.WindowShape attribute*), 159
[GAUSSIAN5](#) (*parselmouth.WindowShape attribute*), 159
[get_average\(\)](#) (*parselmouth.Intensity method*), 64
[get_band_density\(\)](#) (*parselmouth.Spectrum method*), 133
[get_band_density_difference\(\)](#) (*parselmouth.Spectrum method*), 133
[get_band_energy\(\)](#) (*parselmouth.Spectrum method*), 133
[get_band_energy_difference\(\)](#) (*parselmouth.Spectrum method*), 133
[get_bandwidth_at_time\(\)](#) (*parselmouth.Formant method*), 43
[get_bin_number_from_frequency\(\)](#) (*parselmouth.Spectrum method*), 133
[get_bin_width\(\)](#) (*parselmouth.Spectrum method*), 133
[get_c0_value_in_frame\(\)](#) (*parselmouth.CC method*), 35
[get_c0_value_in_frame\(\)](#) (*parselmouth.MFCC method*), 72
[get_center_of_gravity\(\)](#) (*parselmouth.Spectrum method*), 133
[get_central_moment\(\)](#) (*parselmouth.Spectrum method*), 133
[get_centre_of_gravity\(\)](#) (*parselmouth.Spectrum method*), 133
[get_column_distance\(\)](#) (*parselmouth.Harmonicity method*), 54
[get_column_distance\(\)](#) (*parselmouth.Intensity method*), 64
[get_column_distance\(\)](#) (*parselmouth.Matrix method*), 78
[get_column_distance\(\)](#) (*parselmouth.Sound method*), 109
[get_column_distance\(\)](#) (*parselmouth.Spectrogram method*), 123
[get_column_distance\(\)](#) (*parselmouth.Spectrum method*), 133
[get_column_distance\(\)](#) (*parselmouth.Vector method*), 155
[get_end_time\(\)](#) (*parselmouth.CC method*), 35
[get_end_time\(\)](#) (*parselmouth.Formant method*), 43
[get_end_time\(\)](#) (*parselmouth.Harmonicity method*), 54
[get_end_time\(\)](#) (*parselmouth.Intensity method*), 64
[get_end_time\(\)](#) (*parselmouth.MFCC method*), 72
[get_end_time\(\)](#) (*parselmouth.Pitch method*), 86
[get_end_time\(\)](#) (*parselmouth.Sound method*), 110
[get_end_time\(\)](#) (*parselmouth.Spectrogram method*), 123
[get_end_time\(\)](#) (*parselmouth.TimeFrameSampled method*), 143
[get_end_time\(\)](#) (*parselmouth.TimeFunction method*), 148
[get_energy\(\)](#) (*parselmouth.Sound method*), 110
[get_energy_in_air\(\)](#) (*parselmouth.Sound method*), 110
[get_frame\(\)](#) (*parselmouth.CC method*), 35
[get_frame\(\)](#) (*parselmouth.MFCC method*), 72
[get_frame\(\)](#) (*parselmouth.Pitch method*), 86
[get_frame_number_from_time\(\)](#) (*parselmouth.CC method*), 35
[get_frame_number_from_time\(\)](#) (*parselmouth.Formant method*), 43
[get_frame_number_from_time\(\)](#) (*parselmouth.Harmonicity method*), 54
[get_frame_number_from_time\(\)](#) (*parselmouth.Intensity method*), 64
[get_frame_number_from_time\(\)](#) (*parselmouth.MFCC method*), 72
[get_frame_number_from_time\(\)](#) (*parselmouth.Pitch method*), 86

[get_frame_number_from_time\(\)](#) (*parselmouth.Sound method*), 110
[get_frame_number_from_time\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_frame_number_from_time\(\)](#) (*parselmouth.TimeFrameSampled method*), 143
[get_frequency_from_bin_number\(\)](#) (*parselmouth.Spectrum method*), 133
[get_highest_frequency\(\)](#) (*parselmouth.Spectrum method*), 133
[get_highest_x\(\)](#) (*parselmouth.Harmonicity method*), 54
[get_highest_x\(\)](#) (*parselmouth.Intensity method*), 64
[get_highest_x\(\)](#) (*parselmouth.Matrix method*), 78
[get_highest_x\(\)](#) (*parselmouth.Sound method*), 110
[get_highest_x\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_highest_x\(\)](#) (*parselmouth.Spectrum method*), 133
[get_highest_x\(\)](#) (*parselmouth.Vector method*), 155
[get_highest_y\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_highest_y\(\)](#) (*parselmouth.Intensity method*), 64
[get_highest_y\(\)](#) (*parselmouth.Matrix method*), 78
[get_highest_y\(\)](#) (*parselmouth.Sound method*), 110
[get_highest_y\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_highest_y\(\)](#) (*parselmouth.Spectrum method*), 133
[get_highest_y\(\)](#) (*parselmouth.Vector method*), 155
[get_imaginary_value_in_bin\(\)](#) (*parselmouth.Spectrum method*), 133
[get_index_from_time\(\)](#) (*parselmouth.Sound method*), 110
[get_intensity\(\)](#) (*parselmouth.Sound method*), 110
[get_kurtosis\(\)](#) (*parselmouth.Spectrum method*), 134
[get_lowest_frequency\(\)](#) (*parselmouth.Spectrum method*), 134
[get_lowest_x\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_lowest_x\(\)](#) (*parselmouth.Intensity method*), 64
[get_lowest_x\(\)](#) (*parselmouth.Matrix method*), 78
[get_lowest_x\(\)](#) (*parselmouth.Sound method*), 110
[get_lowest_x\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_lowest_x\(\)](#) (*parselmouth.Spectrum method*), 134
[get_lowest_x\(\)](#) (*parselmouth.Vector method*), 155
[get_lowest_y\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_lowest_y\(\)](#) (*parselmouth.Intensity method*), 64
[get_lowest_y\(\)](#) (*parselmouth.Matrix method*), 78
[get_lowest_y\(\)](#) (*parselmouth.Sound method*), 110
[get_lowest_y\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_lowest_y\(\)](#) (*parselmouth.Spectrum method*), 134
[get_lowest_y\(\)](#) (*parselmouth.Vector method*), 155
[get_maximum\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_maximum\(\)](#) (*parselmouth.Intensity method*), 64
[get_maximum\(\)](#) (*parselmouth.Matrix method*), 78
[get_maximum\(\)](#) (*parselmouth.Sound method*), 110
[get_maximum\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_maximum\(\)](#) (*parselmouth.Spectrum method*), 134
[get_maximum\(\)](#) (*parselmouth.Vector method*), 155
[get_mean_absolute_slope\(\)](#) (*parselmouth.Pitch method*), 86
[get_minimum\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_minimum\(\)](#) (*parselmouth.Intensity method*), 64
[get_minimum\(\)](#) (*parselmouth.Matrix method*), 78
[get_minimum\(\)](#) (*parselmouth.Sound method*), 110
[get_minimum\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_minimum\(\)](#) (*parselmouth.Spectrum method*), 134
[get_minimum\(\)](#) (*parselmouth.Vector method*), 155
[get_nearest_zero_crossing\(\)](#) (*parselmouth.Sound method*), 110
[get_number_of_bins\(\)](#) (*parselmouth.Spectrum method*), 134
[get_number_of_channels\(\)](#) (*parselmouth.Sound method*), 110
[get_number_of_coefficients\(\)](#) (*parselmouth.CC method*), 35
[get_number_of_coefficients\(\)](#) (*parselmouth.MFCC method*), 72
[get_number_of_columns\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_number_of_columns\(\)](#) (*parselmouth.Intensity method*), 64
[get_number_of_columns\(\)](#) (*parselmouth.Matrix method*), 78
[get_number_of_columns\(\)](#) (*parselmouth.Sound method*), 110
[get_number_of_columns\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_number_of_columns\(\)](#) (*parselmouth.Spectrum method*), 134
[get_number_of_columns\(\)](#) (*parselmouth.Vector method*), 155
[get_number_of_frames\(\)](#) (*parselmouth.CC method*), 35
[get_number_of_frames\(\)](#) (*parselmouth.Formant method*), 43
[get_number_of_frames\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_number_of_frames\(\)](#) (*parselmouth.Intensity method*), 64
[get_number_of_frames\(\)](#) (*parselmouth.MFCC method*), 72
[get_number_of_frames\(\)](#) (*parselmouth.Pitch method*), 86

[get_number_of_frames\(\)](#) (*parselmouth.Sound method*), 110
[get_number_of_frames\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_number_of_frames\(\)](#) (*parselmouth.TimeFrameSampled method*), 143
[get_number_of_rows\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_number_of_rows\(\)](#) (*parselmouth.Intensity method*), 64
[get_number_of_rows\(\)](#) (*parselmouth.Matrix method*), 79
[get_number_of_rows\(\)](#) (*parselmouth.Sound method*), 110
[get_number_of_rows\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_number_of_rows\(\)](#) (*parselmouth.Spectrum method*), 134
[get_number_of_rows\(\)](#) (*parselmouth.Vector method*), 155
[get_number_of_samples\(\)](#) (*parselmouth.Sound method*), 110
[get_power\(\)](#) (*parselmouth.Sound method*), 110
[get_power_at\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_power_in_air\(\)](#) (*parselmouth.Sound method*), 110
[get_real_value_in_bin\(\)](#) (*parselmouth.Spectrum method*), 134
[get_rms\(\)](#) (*parselmouth.Sound method*), 110
[get_root_mean_square\(\)](#) (*parselmouth.Sound method*), 110
[get_row_distance\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_row_distance\(\)](#) (*parselmouth.Intensity method*), 64
[get_row_distance\(\)](#) (*parselmouth.Matrix method*), 79
[get_row_distance\(\)](#) (*parselmouth.Sound method*), 110
[get_row_distance\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_row_distance\(\)](#) (*parselmouth.Spectrum method*), 134
[get_row_distance\(\)](#) (*parselmouth.Vector method*), 155
[get_sampling_frequency\(\)](#) (*parselmouth.Sound method*), 110
[get_sampling_period\(\)](#) (*parselmouth.Sound method*), 110
[get_skewness\(\)](#) (*parselmouth.Spectrum method*), 134
[get_slope_without_octave_jumps\(\)](#) (*parselmouth.Pitch method*), 86
[get_standard_deviation\(\)](#) (*parselmouth.Spectrum method*), 134
[get_start_time\(\)](#) (*parselmouth.CC method*), 35
[get_start_time\(\)](#) (*parselmouth.Formant method*), 43
[get_start_time\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_start_time\(\)](#) (*parselmouth.Intensity method*), 64
[get_start_time\(\)](#) (*parselmouth.MFCC method*), 72
[get_start_time\(\)](#) (*parselmouth.Pitch method*), 86
[get_start_time\(\)](#) (*parselmouth.Sound method*), 110
[get_start_time\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_start_time\(\)](#) (*parselmouth.TimeFrameSampled method*), 143
[get_start_time\(\)](#) (*parselmouth.TimeFunction method*), 148
[get_sum\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_sum\(\)](#) (*parselmouth.Intensity method*), 64
[get_sum\(\)](#) (*parselmouth.Matrix method*), 79
[get_sum\(\)](#) (*parselmouth.Sound method*), 110
[get_sum\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_sum\(\)](#) (*parselmouth.Spectrum method*), 134
[get_sum\(\)](#) (*parselmouth.Vector method*), 155
[get_time_from_frame_number\(\)](#) (*parselmouth.CC method*), 35
[get_time_from_frame_number\(\)](#) (*parselmouth.Formant method*), 43
[get_time_from_frame_number\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_time_from_frame_number\(\)](#) (*parselmouth.Intensity method*), 64
[get_time_from_frame_number\(\)](#) (*parselmouth.MFCC method*), 72
[get_time_from_frame_number\(\)](#) (*parselmouth.Pitch method*), 86
[get_time_from_frame_number\(\)](#) (*parselmouth.Sound method*), 110
[get_time_from_frame_number\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_time_from_frame_number\(\)](#) (*parselmouth.TimeFrameSampled method*), 143
[get_time_from_index\(\)](#) (*parselmouth.Sound method*), 110
[get_time_step\(\)](#) (*parselmouth.CC method*), 35
[get_time_step\(\)](#) (*parselmouth.Formant method*), 43
[get_time_step\(\)](#) (*parselmouth.Harmonicity method*), 55
[get_time_step\(\)](#) (*parselmouth.Intensity method*), 64
[get_time_step\(\)](#) (*parselmouth.MFCC method*), 72
[get_time_step\(\)](#) (*parselmouth.Pitch method*), 86
[get_time_step\(\)](#) (*parselmouth.Sound method*), 110
[get_time_step\(\)](#) (*parselmouth.Spectrogram method*), 124
[get_time_step\(\)](#) (*parselmouth.TimeFrameSampled method*), 143
[get_total_duration\(\)](#) (*parselmouth.CC method*), 35

- `get_total_duration()` (*parselmouth.Formant method*), 44
 - `get_total_duration()` (*parselmouth.Harmonicity method*), 55
 - `get_total_duration()` (*parselmouth.Intensity method*), 64
 - `get_total_duration()` (*parselmouth.MFCC method*), 72
 - `get_total_duration()` (*parselmouth.Pitch method*), 87
 - `get_total_duration()` (*parselmouth.Sound method*), 111
 - `get_total_duration()` (*parselmouth.Spectrogram method*), 124
 - `get_total_duration()` (*parselmouth.TimeFrameSampled method*), 143
 - `get_total_duration()` (*parselmouth.TimeFunction method*), 148
 - `get_value()` (*parselmouth.Harmonicity method*), 55
 - `get_value()` (*parselmouth.Intensity method*), 64
 - `get_value()` (*parselmouth.Sound method*), 111
 - `get_value()` (*parselmouth.Vector method*), 155
 - `get_value_at_time()` (*parselmouth.Formant method*), 44
 - `get_value_at_time()` (*parselmouth.Pitch method*), 87
 - `get_value_at_xy()` (*parselmouth.Harmonicity method*), 55
 - `get_value_at_xy()` (*parselmouth.Intensity method*), 64
 - `get_value_at_xy()` (*parselmouth.Matrix method*), 79
 - `get_value_at_xy()` (*parselmouth.Sound method*), 111
 - `get_value_at_xy()` (*parselmouth.Spectrogram method*), 124
 - `get_value_at_xy()` (*parselmouth.Spectrum method*), 134
 - `get_value_at_xy()` (*parselmouth.Vector method*), 155
 - `get_value_in_bin()` (*parselmouth.Spectrum method*), 134
 - `get_value_in_cell()` (*parselmouth.Harmonicity method*), 55
 - `get_value_in_cell()` (*parselmouth.Intensity method*), 64
 - `get_value_in_cell()` (*parselmouth.Matrix method*), 79
 - `get_value_in_cell()` (*parselmouth.Sound method*), 111
 - `get_value_in_cell()` (*parselmouth.Spectrogram method*), 124
 - `get_value_in_cell()` (*parselmouth.Spectrum method*), 134
 - `get_value_in_cell()` (*parselmouth.Vector method*), 155
 - `get_value_in_frame()` (*parselmouth.CC method*), 35
 - `get_value_in_frame()` (*parselmouth.MFCC method*), 72
 - `get_value_in_frame()` (*parselmouth.Pitch method*), 87
 - `get_x_of_column()` (*parselmouth.Harmonicity method*), 55
 - `get_x_of_column()` (*parselmouth.Intensity method*), 64
 - `get_x_of_column()` (*parselmouth.Matrix method*), 79
 - `get_x_of_column()` (*parselmouth.Sound method*), 111
 - `get_x_of_column()` (*parselmouth.Spectrogram method*), 124
 - `get_x_of_column()` (*parselmouth.Spectrum method*), 134
 - `get_x_of_column()` (*parselmouth.Vector method*), 155
 - `get_y_of_row()` (*parselmouth.Harmonicity method*), 55
 - `get_y_of_row()` (*parselmouth.Intensity method*), 64
 - `get_y_of_row()` (*parselmouth.Matrix method*), 79
 - `get_y_of_row()` (*parselmouth.Sound method*), 111
 - `get_y_of_row()` (*parselmouth.Spectrogram method*), 124
 - `get_y_of_row()` (*parselmouth.Spectrum method*), 134
 - `get_y_of_row()` (*parselmouth.Vector method*), 155
 - GNE (*parselmouth.Sound.ToHarmonicityMethod attribute*), 107
- ## H
- HAMMING (*parselmouth.SpectralAnalysisWindowShape attribute*), 118
 - HAMMING (*parselmouth.WindowShape attribute*), 159
 - HANNING (*parselmouth.SpectralAnalysisWindowShape attribute*), 118
 - HANNING (*parselmouth.WindowShape attribute*), 159
 - Harmonicity (*class in parselmouth*), 49
 - Harmonicity.FileFormat (*class in parselmouth*), 53
 - HERTZ (*parselmouth.FormantUnit attribute*), 46
 - HERTZ (*parselmouth.PitchUnit attribute*), 90
 - HERTZ_LOGARITHMIC (*parselmouth.PitchUnit attribute*), 90
 - highest_frequency (*parselmouth.Spectrum property*), 136
- ## I
- info() (*parselmouth.CC method*), 35
 - info() (*parselmouth.Data method*), 38
 - info() (*parselmouth.Formant method*), 44
 - info() (*parselmouth.Function method*), 48
 - info() (*parselmouth.Harmonicity method*), 55
 - info() (*parselmouth.Intensity method*), 65
 - info() (*parselmouth.Matrix method*), 79
 - info() (*parselmouth.MFCC method*), 72
 - info() (*parselmouth.Pitch method*), 87
 - info() (*parselmouth.Sampled method*), 93
 - info() (*parselmouth.SampledXY method*), 97
 - info() (*parselmouth.Sound method*), 111
 - info() (*parselmouth.Spectrogram method*), 124

info() (*parselmouth.Spectrum method*), 134
 info() (*parselmouth.TextGrid method*), 138
 info() (*parselmouth.Thing method*), 140
 info() (*parselmouth.TimeFrameSampled method*), 144
 info() (*parselmouth.TimeFunction method*), 148
 info() (*parselmouth.Vector method*), 155
 INTEGRAL (*parselmouth.AmplitudeScaling attribute*), 30
 Intensity (*class in parselmouth*), 58
 intensity (*parselmouth.Pitch.Frame property*), 86
 Intensity.AveragingMethod (*class in parselmouth*), 62
 Intensity.FileFormat (*class in parselmouth*), 62
 interpolate() (*parselmouth.Pitch method*), 87
 Interpolation (*in module parselmouth*), 67

K

KAISER1 (*parselmouth.WindowShape attribute*), 159
 KAISER2 (*parselmouth.WindowShape attribute*), 159
 KAY (*parselmouth.SoundFileFormat attribute*), 117
 kill_octave_jumps() (*parselmouth.Pitch method*), 87

L

lengthen() (*parselmouth.Sound method*), 111
 LINEAR (*parselmouth.ValueInterpolation attribute*), 150
 LOG_HERTZ (*parselmouth.PitchUnit attribute*), 90
 lowest_frequency (*parselmouth.Spectrum property*), 136
 lpc_smoothing() (*parselmouth.Spectrum method*), 134

M

Matrix (*class in parselmouth*), 75
 Matrix.FileFormat (*class in parselmouth*), 77
 max_n_candidates (*parselmouth.Pitch property*), 89
 max_n_coefficients (*parselmouth.CC property*), 36
 max_n_coefficients (*parselmouth.MFCC property*), 74
 MEDIAN (*parselmouth.Intensity.AveragingMethod attribute*), 62
 MEL (*parselmouth.PitchUnit attribute*), 90
 MFCC (*class in parselmouth*), 67
 MFCC.FileFormat (*class in parselmouth*), 71
 MFCC.Frame (*class in parselmouth*), 71
 module
 parselmouth, 27
 parselmouth.praat, 160
 multiply() (*parselmouth.Harmonicity method*), 55
 multiply() (*parselmouth.Intensity method*), 65
 multiply() (*parselmouth.Sound method*), 111
 multiply() (*parselmouth.Vector method*), 155
 multiply_by_window() (*parselmouth.Sound method*), 111

N

n_bins (*parselmouth.Spectrum property*), 136

n_channels (*parselmouth.Sound property*), 114
 n_columns (*parselmouth.Harmonicity property*), 57
 n_columns (*parselmouth.Intensity property*), 66
 n_columns (*parselmouth.Matrix property*), 80
 n_columns (*parselmouth.Sound property*), 114
 n_columns (*parselmouth.Spectrogram property*), 126
 n_columns (*parselmouth.Spectrum property*), 136
 n_columns (*parselmouth.Vector property*), 156
 n_frames (*parselmouth.CC property*), 37
 n_frames (*parselmouth.Formant property*), 45
 n_frames (*parselmouth.Harmonicity property*), 57
 n_frames (*parselmouth.Intensity property*), 66
 n_frames (*parselmouth.MFCC property*), 74
 n_frames (*parselmouth.Pitch property*), 89
 n_frames (*parselmouth.Sound property*), 114
 n_frames (*parselmouth.Spectrogram property*), 126
 n_frames (*parselmouth.TimeFrameSampled property*), 145
 n_rows (*parselmouth.Harmonicity property*), 57
 n_rows (*parselmouth.Intensity property*), 66
 n_rows (*parselmouth.Matrix property*), 80
 n_rows (*parselmouth.Sound property*), 114
 n_rows (*parselmouth.Spectrogram property*), 126
 n_rows (*parselmouth.Spectrum property*), 136
 n_rows (*parselmouth.Vector property*), 156
 n_samples (*parselmouth.Sound property*), 114
 name (*parselmouth.AmplitudeScaling property*), 31
 name (*parselmouth.CC property*), 37
 name (*parselmouth.CC.FileFormat property*), 34
 name (*parselmouth.Data property*), 39
 name (*parselmouth.Data.FileFormat property*), 38
 name (*parselmouth.Formant property*), 45
 name (*parselmouth.Formant.FileFormat property*), 43
 name (*parselmouth.FormantUnit property*), 46
 name (*parselmouth.Function property*), 49
 name (*parselmouth.Function.FileFormat property*), 48
 name (*parselmouth.Harmonicity property*), 57
 name (*parselmouth.Harmonicity.FileFormat property*), 53
 name (*parselmouth.Intensity property*), 66
 name (*parselmouth.Intensity.AveragingMethod property*), 62
 name (*parselmouth.Intensity.FileFormat property*), 63
 name (*parselmouth.Matrix property*), 80
 name (*parselmouth.Matrix.FileFormat property*), 78
 name (*parselmouth.MFCC property*), 74
 name (*parselmouth.MFCC.FileFormat property*), 71
 name (*parselmouth.Pitch property*), 89
 name (*parselmouth.Pitch.FileFormat property*), 85
 name (*parselmouth.PitchUnit property*), 91
 name (*parselmouth.Sampled property*), 94
 name (*parselmouth.Sampled.FileFormat property*), 92
 name (*parselmouth.SampledXY property*), 98
 name (*parselmouth.SampledXY.FileFormat property*), 97

name (*parselmouth.SignalOutsideTimeDomain* property), 99
 name (*parselmouth.Sound* property), 114
 name (*parselmouth.Sound.FileFormat* property), 106
 name (*parselmouth.Sound.ToHarmonicityMethod* property), 107
 name (*parselmouth.Sound.ToPitchMethod* property), 107
 name (*parselmouth.SoundFileFormat* property), 117
 name (*parselmouth.SpectralAnalysisWindowShape* property), 119
 name (*parselmouth.Spectrogram* property), 126
 name (*parselmouth.Spectrogram.FileFormat* property), 123
 name (*parselmouth.Spectrum* property), 136
 name (*parselmouth.Spectrum.FileFormat* property), 132
 name (*parselmouth.TextGrid* property), 139
 name (*parselmouth.TextGrid.FileFormat* property), 138
 name (*parselmouth.Thing* property), 140
 name (*parselmouth.TimeFrameSampled* property), 145
 name (*parselmouth.TimeFrameSampled.FileFormat* property), 143
 name (*parselmouth.TimeFunction* property), 149
 name (*parselmouth.TimeFunction.FileFormat* property), 148
 name (*parselmouth.ValueInterpolation* property), 150
 name (*parselmouth.Vector* property), 156
 name (*parselmouth.Vector.FileFormat* property), 154
 name (*parselmouth.WindowShape* property), 159
 NEAREST (*parselmouth.ValueInterpolation* attribute), 150
 NEXT_SUN (*parselmouth.SoundFileFormat* attribute), 117
 nf (*parselmouth.Spectrum* property), 136
 NIST (*parselmouth.SoundFileFormat* attribute), 117
 NORMALIZE (*parselmouth.AmplitudeScaling* attribute), 30
 nt (*parselmouth.CC* property), 37
 nt (*parselmouth.Formant* property), 45
 nt (*parselmouth.Harmonicity* property), 57
 nt (*parselmouth.Intensity* property), 66
 nt (*parselmouth.MFCC* property), 74
 nt (*parselmouth.Pitch* property), 89
 nt (*parselmouth.Sound* property), 114
 nt (*parselmouth.Spectrogram* property), 126
 nt (*parselmouth.TimeFrameSampled* property), 145
 nx (*parselmouth.CC* property), 37
 nx (*parselmouth.Formant* property), 45
 nx (*parselmouth.Harmonicity* property), 57
 nx (*parselmouth.Intensity* property), 66
 nx (*parselmouth.Matrix* property), 80
 nx (*parselmouth.MFCC* property), 74
 nx (*parselmouth.Pitch* property), 89
 nx (*parselmouth.Sampled* property), 94
 nx (*parselmouth.SampledXY* property), 98
 nx (*parselmouth.Sound* property), 114
 nx (*parselmouth.Spectrogram* property), 126
 nx (*parselmouth.Spectrum* property), 136

nx (*parselmouth.TimeFrameSampled* property), 145
 nx (*parselmouth.Vector* property), 156
 ny (*parselmouth.Harmonicity* property), 57
 ny (*parselmouth.Intensity* property), 66
 ny (*parselmouth.Matrix* property), 80
 ny (*parselmouth.SampledXY* property), 98
 ny (*parselmouth.Sound* property), 114
 ny (*parselmouth.Spectrogram* property), 126
 ny (*parselmouth.Spectrum* property), 136
 ny (*parselmouth.Vector* property), 157

O

octave_down() (*parselmouth.Pitch* method), 87
 octave_up() (*parselmouth.Pitch* method), 87
 override_sampling_frequency() (*parselmouth.Sound* method), 111

P

PARABOLIC (*parselmouth.WindowShape* attribute), 159
 parselmouth
 module, 27
 parselmouth.praat
 module, 160
 path_finder() (*parselmouth.Pitch* method), 87
 PEAK_0_99 (*parselmouth.AmplitudeScaling* attribute), 30
 Pitch (class in *parselmouth*), 80
 Pitch.Candidate (class in *parselmouth*), 85
 Pitch.FileFormat (class in *parselmouth*), 85
 Pitch.Frame (class in *parselmouth*), 85
 PitchUnit (class in *parselmouth*), 89
 PRAAT_VERSION (in module *parselmouth*), 27
 PRAAT_VERSION_DATE (in module *parselmouth*), 27
 PraatError, 159
 PraatFatal, 159
 PraatWarning, 159
 pre_emphasize() (*parselmouth.Sound* method), 111

R

RAW_16_BE (*parselmouth.SoundFileFormat* attribute), 117
 RAW_16_LE (*parselmouth.SoundFileFormat* attribute), 117
 RAW_24_BE (*parselmouth.SoundFileFormat* attribute), 117
 RAW_24_LE (*parselmouth.SoundFileFormat* attribute), 117
 RAW_32_BE (*parselmouth.SoundFileFormat* attribute), 117
 RAW_32_LE (*parselmouth.SoundFileFormat* attribute), 117
 RAW_8_SIGNED (*parselmouth.SoundFileFormat* attribute), 117
 RAW_8_UNSIGNED (*parselmouth.SoundFileFormat* attribute), 117

[read\(\)](#) (in module *parselmouth*), 27
[read\(\)](#) (*parselmouth.CC* static method), 35
[read\(\)](#) (*parselmouth.Data* static method), 39
[read\(\)](#) (*parselmouth.Formant* static method), 44
[read\(\)](#) (*parselmouth.Function* static method), 48
[read\(\)](#) (*parselmouth.Harmonicity* static method), 55
[read\(\)](#) (*parselmouth.Intensity* static method), 65
[read\(\)](#) (*parselmouth.Matrix* static method), 79
[read\(\)](#) (*parselmouth.MFCC* static method), 72
[read\(\)](#) (*parselmouth.Pitch* static method), 87
[read\(\)](#) (*parselmouth.Sampled* static method), 93
[read\(\)](#) (*parselmouth.SampledXY* static method), 97
[read\(\)](#) (*parselmouth.Sound* static method), 111
[read\(\)](#) (*parselmouth.Spectrogram* static method), 124
[read\(\)](#) (*parselmouth.Spectrum* static method), 134
[read\(\)](#) (*parselmouth.TextGrid* static method), 138
[read\(\)](#) (*parselmouth.TimeFrameSampled* static method), 144
[read\(\)](#) (*parselmouth.TimeFunction* static method), 148
[read\(\)](#) (*parselmouth.Vector* static method), 155
[RECTANGULAR](#) (*parselmouth.WindowShape* attribute), 159
[resample\(\)](#) (*parselmouth.Sound* method), 111
[reverse\(\)](#) (*parselmouth.Sound* method), 111
[run\(\)](#) (in module *parselmouth.praat*), 162
[run_file\(\)](#) (in module *parselmouth.praat*), 163

S

[Sampled](#) (class in *parselmouth*), 91
[Sampled.FileFormat](#) (class in *parselmouth*), 92
[SampledXY](#) (class in *parselmouth*), 94
[SampledXY.FileFormat](#) (class in *parselmouth*), 96
[sampling_frequency](#) (*parselmouth.Sound* property), 114
[sampling_period](#) (*parselmouth.Sound* property), 114
[save\(\)](#) (*parselmouth.CC* method), 35
[save\(\)](#) (*parselmouth.Data* method), 39
[save\(\)](#) (*parselmouth.Formant* method), 44
[save\(\)](#) (*parselmouth.Function* method), 48
[save\(\)](#) (*parselmouth.Harmonicity* method), 55
[save\(\)](#) (*parselmouth.Intensity* method), 65
[save\(\)](#) (*parselmouth.Matrix* method), 79
[save\(\)](#) (*parselmouth.MFCC* method), 73
[save\(\)](#) (*parselmouth.Pitch* method), 87
[save\(\)](#) (*parselmouth.Sampled* method), 93
[save\(\)](#) (*parselmouth.SampledXY* method), 97
[save\(\)](#) (*parselmouth.Sound* method), 111
[save\(\)](#) (*parselmouth.Spectrogram* method), 124
[save\(\)](#) (*parselmouth.Spectrum* method), 134
[save\(\)](#) (*parselmouth.TextGrid* method), 139
[save\(\)](#) (*parselmouth.TimeFrameSampled* method), 144
[save\(\)](#) (*parselmouth.TimeFunction* method), 148
[save\(\)](#) (*parselmouth.Vector* method), 155

[save_as_binary_file\(\)](#) (*parselmouth.CC* method), 35
[save_as_binary_file\(\)](#) (*parselmouth.Data* method), 39
[save_as_binary_file\(\)](#) (*parselmouth.Formant* method), 44
[save_as_binary_file\(\)](#) (*parselmouth.Function* method), 49
[save_as_binary_file\(\)](#) (*parselmouth.Harmonicity* method), 56
[save_as_binary_file\(\)](#) (*parselmouth.Intensity* method), 65
[save_as_binary_file\(\)](#) (*parselmouth.Matrix* method), 79
[save_as_binary_file\(\)](#) (*parselmouth.MFCC* method), 73
[save_as_binary_file\(\)](#) (*parselmouth.Pitch* method), 87
[save_as_binary_file\(\)](#) (*parselmouth.Sampled* method), 93
[save_as_binary_file\(\)](#) (*parselmouth.SampledXY* method), 97
[save_as_binary_file\(\)](#) (*parselmouth.Sound* method), 111
[save_as_binary_file\(\)](#) (*parselmouth.Spectrogram* method), 125
[save_as_binary_file\(\)](#) (*parselmouth.Spectrum* method), 135
[save_as_binary_file\(\)](#) (*parselmouth.TextGrid* method), 139
[save_as_binary_file\(\)](#) (*parselmouth.TimeFrameSampled* method), 144
[save_as_binary_file\(\)](#) (*parselmouth.TimeFunction* method), 148
[save_as_binary_file\(\)](#) (*parselmouth.Vector* method), 156
[save_as_headerless_spreadsheet_file\(\)](#) (*parselmouth.Harmonicity* method), 56
[save_as_headerless_spreadsheet_file\(\)](#) (*parselmouth.Intensity* method), 65
[save_as_headerless_spreadsheet_file\(\)](#) (*parselmouth.Matrix* method), 79
[save_as_headerless_spreadsheet_file\(\)](#) (*parselmouth.Sound* method), 112
[save_as_headerless_spreadsheet_file\(\)](#) (*parselmouth.Spectrogram* method), 125
[save_as_headerless_spreadsheet_file\(\)](#) (*parselmouth.Spectrum* method), 135
[save_as_headerless_spreadsheet_file\(\)](#) (*parselmouth.Vector* method), 156
[save_as_matrix_text_file\(\)](#) (*parselmouth.Harmonicity* method), 56
[save_as_matrix_text_file\(\)](#) (*parselmouth.Intensity* method), 65

`save_as_matrix_text_file()` (*parselmouth.Matrix method*), 79
`save_as_matrix_text_file()` (*parselmouth.Sound method*), 112
`save_as_matrix_text_file()` (*parselmouth.Spectrogram method*), 125
`save_as_matrix_text_file()` (*parselmouth.Spectrum method*), 135
`save_as_matrix_text_file()` (*parselmouth.Vector method*), 156
`save_as_short_text_file()` (*parselmouth.CC method*), 36
`save_as_short_text_file()` (*parselmouth.Data method*), 39
`save_as_short_text_file()` (*parselmouth.Formant method*), 44
`save_as_short_text_file()` (*parselmouth.Function method*), 49
`save_as_short_text_file()` (*parselmouth.Harmonicity method*), 56
`save_as_short_text_file()` (*parselmouth.Intensity method*), 65
`save_as_short_text_file()` (*parselmouth.Matrix method*), 79
`save_as_short_text_file()` (*parselmouth.MFCC method*), 73
`save_as_short_text_file()` (*parselmouth.Pitch method*), 87
`save_as_short_text_file()` (*parselmouth.Sampled method*), 93
`save_as_short_text_file()` (*parselmouth.SampledXY method*), 97
`save_as_short_text_file()` (*parselmouth.Sound method*), 112
`save_as_short_text_file()` (*parselmouth.Spectrogram method*), 125
`save_as_short_text_file()` (*parselmouth.Spectrum method*), 135
`save_as_short_text_file()` (*parselmouth.TextGrid method*), 139
`save_as_short_text_file()` (*parselmouth.TimeFrameSampled method*), 144
`save_as_short_text_file()` (*parselmouth.TimeFunction method*), 148
`save_as_short_text_file()` (*parselmouth.Vector method*), 156
`save_as_text_file()` (*parselmouth.CC method*), 36
`save_as_text_file()` (*parselmouth.Data method*), 39
`save_as_text_file()` (*parselmouth.Formant method*), 44
`save_as_text_file()` (*parselmouth.Function method*), 49
`save_as_text_file()` (*parselmouth.Harmonicity method*), 56
`save_as_text_file()` (*parselmouth.Intensity method*), 65
`save_as_text_file()` (*parselmouth.Matrix method*), 79
`save_as_text_file()` (*parselmouth.MFCC method*), 73
`save_as_text_file()` (*parselmouth.Pitch method*), 87
`save_as_text_file()` (*parselmouth.Sampled method*), 93
`save_as_text_file()` (*parselmouth.SampledXY method*), 97
`save_as_text_file()` (*parselmouth.Sound method*), 112
`save_as_text_file()` (*parselmouth.Spectrogram method*), 125
`save_as_text_file()` (*parselmouth.Spectrum method*), 135
`save_as_text_file()` (*parselmouth.TextGrid method*), 139
`save_as_text_file()` (*parselmouth.TimeFrameSampled method*), 144
`save_as_text_file()` (*parselmouth.TimeFunction method*), 148
`save_as_text_file()` (*parselmouth.Vector method*), 156
`scale()` (*parselmouth.Harmonicity method*), 56
`scale()` (*parselmouth.Intensity method*), 65
`scale()` (*parselmouth.Sound method*), 112
`scale()` (*parselmouth.Vector method*), 156
`scale_intensity()` (*parselmouth.Sound method*), 112
`scale_peak()` (*parselmouth.Harmonicity method*), 56
`scale_peak()` (*parselmouth.Intensity method*), 65
`scale_peak()` (*parselmouth.Sound method*), 112
`scale_peak()` (*parselmouth.Vector method*), 156
`scale_times_by()` (*parselmouth.CC method*), 36
`scale_times_by()` (*parselmouth.Formant method*), 44
`scale_times_by()` (*parselmouth.Harmonicity method*), 56
`scale_times_by()` (*parselmouth.Intensity method*), 65
`scale_times_by()` (*parselmouth.MFCC method*), 73
`scale_times_by()` (*parselmouth.Pitch method*), 87
`scale_times_by()` (*parselmouth.Sound method*), 112
`scale_times_by()` (*parselmouth.Spectrogram method*), 125
`scale_times_by()` (*parselmouth.TimeFrameSampled method*), 144
`scale_times_by()` (*parselmouth.TimeFunction method*), 148
`scale_times_to()` (*parselmouth.CC method*), 36
`scale_times_to()` (*parselmouth.Formant method*), 44
`scale_times_to()` (*parselmouth.Harmonicity method*), 56
`scale_times_to()` (*parselmouth.Intensity method*), 65
`scale_times_to()` (*parselmouth.MFCC method*), 73

- `scale_times_to()` (*parselmouth.Pitch method*), 87
- `scale_times_to()` (*parselmouth.Sound method*), 112
- `scale_times_to()` (*parselmouth.Spectrogram method*), 125
- `scale_times_to()` (*parselmouth.TimeFrameSampled method*), 144
- `scale_times_to()` (*parselmouth.TimeFunction method*), 149
- `scale_x_by()` (*parselmouth.CC method*), 36
- `scale_x_by()` (*parselmouth.Formant method*), 44
- `scale_x_by()` (*parselmouth.Function method*), 49
- `scale_x_by()` (*parselmouth.Harmonicity method*), 56
- `scale_x_by()` (*parselmouth.Intensity method*), 65
- `scale_x_by()` (*parselmouth.Matrix method*), 79
- `scale_x_by()` (*parselmouth.MFCC method*), 73
- `scale_x_by()` (*parselmouth.Pitch method*), 87
- `scale_x_by()` (*parselmouth.Sampled method*), 93
- `scale_x_by()` (*parselmouth.SampledXY method*), 97
- `scale_x_by()` (*parselmouth.Sound method*), 112
- `scale_x_by()` (*parselmouth.Spectrogram method*), 125
- `scale_x_by()` (*parselmouth.Spectrum method*), 135
- `scale_x_by()` (*parselmouth.TextGrid method*), 139
- `scale_x_by()` (*parselmouth.TimeFrameSampled method*), 144
- `scale_x_by()` (*parselmouth.TimeFunction method*), 149
- `scale_x_by()` (*parselmouth.Vector method*), 156
- `scale_x_to()` (*parselmouth.CC method*), 36
- `scale_x_to()` (*parselmouth.Formant method*), 44
- `scale_x_to()` (*parselmouth.Function method*), 49
- `scale_x_to()` (*parselmouth.Harmonicity method*), 56
- `scale_x_to()` (*parselmouth.Intensity method*), 65
- `scale_x_to()` (*parselmouth.Matrix method*), 79
- `scale_x_to()` (*parselmouth.MFCC method*), 73
- `scale_x_to()` (*parselmouth.Pitch method*), 87
- `scale_x_to()` (*parselmouth.Sampled method*), 93
- `scale_x_to()` (*parselmouth.SampledXY method*), 97
- `scale_x_to()` (*parselmouth.Sound method*), 112
- `scale_x_to()` (*parselmouth.Spectrogram method*), 125
- `scale_x_to()` (*parselmouth.Spectrum method*), 135
- `scale_x_to()` (*parselmouth.TextGrid method*), 139
- `scale_x_to()` (*parselmouth.TimeFrameSampled method*), 144
- `scale_x_to()` (*parselmouth.TimeFunction method*), 149
- `scale_x_to()` (*parselmouth.Vector method*), 156
- `select()` (*parselmouth.Pitch.Frame method*), 85
- `selected` (*parselmouth.Pitch property*), 89
- `selected` (*parselmouth.Pitch.Frame property*), 86
- `selected_array` (*parselmouth.Pitch property*), 89
- `SEMITONES_1` (*parselmouth.PitchUnit attribute*), 90
- `SEMITONES_100` (*parselmouth.PitchUnit attribute*), 91
- `SEMITONES_200` (*parselmouth.PitchUnit attribute*), 91
- `SEMITONES_440` (*parselmouth.PitchUnit attribute*), 91
- `SESAM` (*parselmouth.SoundFileFormat attribute*), 117
- `set_imaginary_value_in_bin()` (*parselmouth.Spectrum method*), 135
- `set_real_value_in_bin()` (*parselmouth.Spectrum method*), 135
- `set_to_zero()` (*parselmouth.Sound method*), 112
- `set_value()` (*parselmouth.Harmonicity method*), 56
- `set_value()` (*parselmouth.Intensity method*), 65
- `set_value()` (*parselmouth.Matrix method*), 79
- `set_value()` (*parselmouth.Sound method*), 112
- `set_value()` (*parselmouth.Spectrogram method*), 125
- `set_value()` (*parselmouth.Spectrum method*), 135
- `set_value()` (*parselmouth.Vector method*), 156
- `set_value_in_bin()` (*parselmouth.Spectrum method*), 135
- `shift_times_by()` (*parselmouth.CC method*), 36
- `shift_times_by()` (*parselmouth.Formant method*), 44
- `shift_times_by()` (*parselmouth.Harmonicity method*), 56
- `shift_times_by()` (*parselmouth.Intensity method*), 65
- `shift_times_by()` (*parselmouth.MFCC method*), 73
- `shift_times_by()` (*parselmouth.Pitch method*), 87
- `shift_times_by()` (*parselmouth.Sound method*), 112
- `shift_times_by()` (*parselmouth.Spectrogram method*), 125
- `shift_times_by()` (*parselmouth.TimeFrameSampled method*), 144
- `shift_times_by()` (*parselmouth.TimeFunction method*), 149
- `shift_times_to()` (*parselmouth.CC method*), 36
- `shift_times_to()` (*parselmouth.Formant method*), 44
- `shift_times_to()` (*parselmouth.Harmonicity method*), 56
- `shift_times_to()` (*parselmouth.Intensity method*), 65
- `shift_times_to()` (*parselmouth.MFCC method*), 73
- `shift_times_to()` (*parselmouth.Pitch method*), 88
- `shift_times_to()` (*parselmouth.Sound method*), 112
- `shift_times_to()` (*parselmouth.Spectrogram method*), 125
- `shift_times_to()` (*parselmouth.TimeFrameSampled method*), 144
- `shift_times_to()` (*parselmouth.TimeFunction method*), 149
- `shift_x_by()` (*parselmouth.CC method*), 36
- `shift_x_by()` (*parselmouth.Formant method*), 44
- `shift_x_by()` (*parselmouth.Function method*), 49
- `shift_x_by()` (*parselmouth.Harmonicity method*), 56
- `shift_x_by()` (*parselmouth.Intensity method*), 65
- `shift_x_by()` (*parselmouth.Matrix method*), 79
- `shift_x_by()` (*parselmouth.MFCC method*), 73
- `shift_x_by()` (*parselmouth.Pitch method*), 88
- `shift_x_by()` (*parselmouth.Sampled method*), 93
- `shift_x_by()` (*parselmouth.SampledXY method*), 97
- `shift_x_by()` (*parselmouth.Sound method*), 112
- `shift_x_by()` (*parselmouth.Spectrogram method*), 125

- `shift_x_by()` (*parselmouth.Spectrum* method), 135
- `shift_x_by()` (*parselmouth.TextGrid* method), 139
- `shift_x_by()` (*parselmouth.TimeFrameSampled* method), 144
- `shift_x_by()` (*parselmouth.TimeFunction* method), 149
- `shift_x_by()` (*parselmouth.Vector* method), 156
- `shift_x_to()` (*parselmouth.CC* method), 36
- `shift_x_to()` (*parselmouth.Formant* method), 44
- `shift_x_to()` (*parselmouth.Function* method), 49
- `shift_x_to()` (*parselmouth.Harmonicity* method), 56
- `shift_x_to()` (*parselmouth.Intensity* method), 65
- `shift_x_to()` (*parselmouth.Matrix* method), 79
- `shift_x_to()` (*parselmouth.MFCC* method), 73
- `shift_x_to()` (*parselmouth.Pitch* method), 88
- `shift_x_to()` (*parselmouth.Sampled* method), 93
- `shift_x_to()` (*parselmouth.SampledXY* method), 98
- `shift_x_to()` (*parselmouth.Sound* method), 112
- `shift_x_to()` (*parselmouth.Spectrogram* method), 125
- `shift_x_to()` (*parselmouth.Spectrum* method), 135
- `shift_x_to()` (*parselmouth.TextGrid* method), 139
- `shift_x_to()` (*parselmouth.TimeFrameSampled* method), 144
- `shift_x_to()` (*parselmouth.TimeFunction* method), 149
- `shift_x_to()` (*parselmouth.Vector* method), 156
- `SHORT_TEXT` (*parselmouth.CC.FileFormat* attribute), 34
- `SHORT_TEXT` (*parselmouth.Data.FileFormat* attribute), 38
- `SHORT_TEXT` (*parselmouth.Formant.FileFormat* attribute), 43
- `SHORT_TEXT` (*parselmouth.Function.FileFormat* attribute), 48
- `SHORT_TEXT` (*parselmouth.Harmonicity.FileFormat* attribute), 53
- `SHORT_TEXT` (*parselmouth.Intensity.FileFormat* attribute), 63
- `SHORT_TEXT` (*parselmouth.Matrix.FileFormat* attribute), 78
- `SHORT_TEXT` (*parselmouth.MFCC.FileFormat* attribute), 71
- `SHORT_TEXT` (*parselmouth.Pitch.FileFormat* attribute), 85
- `SHORT_TEXT` (*parselmouth.Sampled.FileFormat* attribute), 92
- `SHORT_TEXT` (*parselmouth.SampledXY.FileFormat* attribute), 97
- `SHORT_TEXT` (*parselmouth.Sound.FileFormat* attribute), 106
- `SHORT_TEXT` (*parselmouth.Spectrogram.FileFormat* attribute), 123
- `SHORT_TEXT` (*parselmouth.Spectrum.FileFormat* attribute), 132
- `SHORT_TEXT` (*parselmouth.TextGrid.FileFormat* attribute), 138
- `SHORT_TEXT` (*parselmouth.TimeFrameSampled.FileFormat* attribute), 143
- `SHORT_TEXT` (*parselmouth.TimeFunction.FileFormat* attribute), 148
- `SHORT_TEXT` (*parselmouth.Vector.FileFormat* attribute), 154
- `SHS` (*parselmouth.Sound.ToPitchMethod* attribute), 107
- `SignalOutsideTimeDomain` (class in *parselmouth*), 98
- `SIMILAR` (*parselmouth.SignalOutsideTimeDomain* attribute), 99
- `SINC70` (*parselmouth.ValueInterpolation* attribute), 150
- `SINC700` (*parselmouth.ValueInterpolation* attribute), 150
- `smooth()` (*parselmouth.Pitch* method), 88
- `SONES` (*parselmouth.Intensity.AveragingMethod* attribute), 62
- `Sound` (class in *parselmouth*), 100
- `Sound.FileFormat` (class in *parselmouth*), 106
- `Sound.ToHarmonicityMethod` (class in *parselmouth*), 106
- `Sound.ToPitchMethod` (class in *parselmouth*), 107
- `SoundFileFormat` (class in *parselmouth*), 115
- `SpectralAnalysisWindowShape` (class in *parselmouth*), 117
- `Spectrogram` (class in *parselmouth*), 119
- `Spectrogram.FileFormat` (class in *parselmouth*), 122
- `Spectrum` (class in *parselmouth*), 127
- `Spectrum.FileFormat` (class in *parselmouth*), 132
- `SPINET` (*parselmouth.Sound.ToPitchMethod* attribute), 107
- `SQUARE` (*parselmouth.SpectralAnalysisWindowShape* attribute), 118
- `start_time` (*parselmouth.CC* property), 37
- `start_time` (*parselmouth.Formant* property), 45
- `start_time` (*parselmouth.Harmonicity* property), 57
- `start_time` (*parselmouth.Intensity* property), 66
- `start_time` (*parselmouth.MFCC* property), 74
- `start_time` (*parselmouth.Pitch* property), 89
- `start_time` (*parselmouth.Sound* property), 114
- `start_time` (*parselmouth.Spectrogram* property), 126
- `start_time` (*parselmouth.TimeFrameSampled* property), 145
- `start_time` (*parselmouth.TimeFunction* property), 149
- `step()` (*parselmouth.Pitch* method), 88
- `strength` (*parselmouth.Pitch.Candidate* property), 85
- `subtract()` (*parselmouth.Harmonicity* method), 56
- `subtract()` (*parselmouth.Intensity* method), 65
- `subtract()` (*parselmouth.Sound* method), 112
- `subtract()` (*parselmouth.Vector* method), 156
- `subtract_linear_fit()` (*parselmouth.Pitch* method), 88
- `subtract_mean()` (*parselmouth.Harmonicity* method), 56
- `subtract_mean()` (*parselmouth.Intensity* method), 65
- `subtract_mean()` (*parselmouth.Sound* method), 112
- `subtract_mean()` (*parselmouth.Vector* method), 156

SUM (*parselmouth.AmplitudeScaling* attribute), 30
 synthesize_sound() (*parselmouth.Spectrogram*
 method), 125

T

t1 (*parselmouth.CC* property), 37
 t1 (*parselmouth.Formant* property), 45
 t1 (*parselmouth.Harmonicity* property), 57
 t1 (*parselmouth.Intensity* property), 66
 t1 (*parselmouth.MFCC* property), 74
 t1 (*parselmouth.Pitch* property), 89
 t1 (*parselmouth.Sound* property), 114
 t1 (*parselmouth.Spectrogram* property), 126
 t1 (*parselmouth.TimeFrameSampled* property), 145
 t_bins() (*parselmouth.CC* method), 36
 t_bins() (*parselmouth.Formant* method), 44
 t_bins() (*parselmouth.Harmonicity* method), 56
 t_bins() (*parselmouth.Intensity* method), 66
 t_bins() (*parselmouth.MFCC* method), 73
 t_bins() (*parselmouth.Pitch* method), 88
 t_bins() (*parselmouth.Sound* method), 112
 t_bins() (*parselmouth.Spectrogram* method), 125
 t_bins() (*parselmouth.TimeFrameSampled* method),
 144
 t_grid() (*parselmouth.CC* method), 36
 t_grid() (*parselmouth.Formant* method), 44
 t_grid() (*parselmouth.Harmonicity* method), 56
 t_grid() (*parselmouth.Intensity* method), 66
 t_grid() (*parselmouth.MFCC* method), 73
 t_grid() (*parselmouth.Pitch* method), 88
 t_grid() (*parselmouth.Sound* method), 112
 t_grid() (*parselmouth.Spectrogram* method), 125
 t_grid() (*parselmouth.TimeFrameSampled* method),
 144
 TEXT (*parselmouth.CC.FileFormat* attribute), 34
 TEXT (*parselmouth.Data.FileFormat* attribute), 38
 TEXT (*parselmouth.Formant.FileFormat* attribute), 43
 TEXT (*parselmouth.Function.FileFormat* attribute), 48
 TEXT (*parselmouth.Harmonicity.FileFormat* attribute),
 53
 TEXT (*parselmouth.Intensity.FileFormat* attribute), 63
 TEXT (*parselmouth.Matrix.FileFormat* attribute), 78
 TEXT (*parselmouth.MFCC.FileFormat* attribute), 71
 TEXT (*parselmouth.Pitch.FileFormat* attribute), 85
 TEXT (*parselmouth.Sampled.FileFormat* attribute), 92
 TEXT (*parselmouth.SampledXY.FileFormat* attribute), 97
 TEXT (*parselmouth.Sound.FileFormat* attribute), 106
 TEXT (*parselmouth.Spectrogram.FileFormat* attribute),
 123
 TEXT (*parselmouth.Spectrum.FileFormat* attribute), 132
 TEXT (*parselmouth.TextGrid.FileFormat* attribute), 138
 TEXT (*parselmouth.TimeFrameSampled.FileFormat* at-
 tribute), 143

TEXT (*parselmouth.TimeFunction.FileFormat* attribute),
 148
 TEXT (*parselmouth.Vector.FileFormat* attribute), 154
 TextGrid (class in *parselmouth*), 136
 TextGrid.FileFormat (class in *parselmouth*), 137
 Thing (class in *parselmouth*), 139
 time_range (*parselmouth.CC* property), 37
 time_range (*parselmouth.Formant* property), 45
 time_range (*parselmouth.Harmonicity* property), 57
 time_range (*parselmouth.Intensity* property), 66
 time_range (*parselmouth.MFCC* property), 74
 time_range (*parselmouth.Pitch* property), 89
 time_range (*parselmouth.Sound* property), 114
 time_range (*parselmouth.Spectrogram* property), 126
 time_range (*parselmouth.TimeFrameSampled* prop-
 erty), 145
 time_range (*parselmouth.TimeFunction* property), 149
 time_step (*parselmouth.CC* property), 37
 time_step (*parselmouth.Formant* property), 45
 time_step (*parselmouth.Harmonicity* property), 57
 time_step (*parselmouth.Intensity* property), 66
 time_step (*parselmouth.MFCC* property), 74
 time_step (*parselmouth.Pitch* property), 89
 time_step (*parselmouth.Sound* property), 114
 time_step (*parselmouth.Spectrogram* property), 126
 time_step (*parselmouth.TimeFrameSampled* property),
 145
 time_to_frame_number() (*parselmouth.CC* method),
 36
 time_to_frame_number() (*parselmouth.Formant*
 method), 44
 time_to_frame_number() (*parselmouth.Harmonicity*
 method), 56
 time_to_frame_number() (*parselmouth.Intensity*
 method), 66
 time_to_frame_number() (*parselmouth.MFCC*
 method), 73
 time_to_frame_number() (*parselmouth.Pitch*
 method), 88
 time_to_frame_number() (*parselmouth.Sound*
 method), 112
 time_to_frame_number() (*parselmouth.Spectrogram*
 method), 125
 time_to_frame_number() (*parselmouth.TimeFrameSampled* method), 144
 TimeFrameSampled (class in *parselmouth*), 140
 TimeFrameSampled.FileFormat (class in *parselmouth*), 142
 TimeFunction (class in *parselmouth*), 145
 TimeFunction.FileFormat (class in *parselmouth*), 147
 tmax (*parselmouth.CC* property), 37
 tmax (*parselmouth.Formant* property), 45
 tmax (*parselmouth.Harmonicity* property), 57
 tmax (*parselmouth.Intensity* property), 66

[tmax \(parselmouth.MFCC property\)](#), 74
[tmax \(parselmouth.Pitch property\)](#), 89
[tmax \(parselmouth.Sound property\)](#), 114
[tmax \(parselmouth.Spectrogram property\)](#), 126
[tmax \(parselmouth.TimeFrameSampled property\)](#), 145
[tmax \(parselmouth.TimeFunction property\)](#), 149
[tmin \(parselmouth.CC property\)](#), 37
[tmin \(parselmouth.Formant property\)](#), 45
[tmin \(parselmouth.Harmonicity property\)](#), 57
[tmin \(parselmouth.Intensity property\)](#), 67
[tmin \(parselmouth.MFCC property\)](#), 74
[tmin \(parselmouth.Pitch property\)](#), 89
[tmin \(parselmouth.Sound property\)](#), 115
[tmin \(parselmouth.Spectrogram property\)](#), 126
[tmin \(parselmouth.TimeFrameSampled property\)](#), 145
[tmin \(parselmouth.TimeFunction property\)](#), 149
[to_array\(\) \(parselmouth.CC method\)](#), 36
[to_array\(\) \(parselmouth.CC.Frame method\)](#), 34
[to_array\(\) \(parselmouth.MFCC method\)](#), 73
[to_array\(\) \(parselmouth.MFCC.Frame method\)](#), 71
[to_array\(\) \(parselmouth.Pitch method\)](#), 88
[to_formant_burg\(\) \(parselmouth.Sound method\)](#), 112
[to_harmonicity\(\) \(parselmouth.Sound method\)](#), 112
[to_harmonicity_ac\(\) \(parselmouth.Sound method\)](#), 112
[to_harmonicity_cc\(\) \(parselmouth.Sound method\)](#), 113
[to_harmonicity_gne\(\) \(parselmouth.Sound method\)](#), 113
[to_intensity\(\) \(parselmouth.Sound method\)](#), 113
[to_matrix\(\) \(parselmouth.CC method\)](#), 36
[to_matrix\(\) \(parselmouth.MFCC method\)](#), 73
[to_matrix\(\) \(parselmouth.Pitch method\)](#), 88
[to_matrix_features\(\) \(parselmouth.MFCC method\)](#), 73
[to_mfcc\(\) \(parselmouth.Sound method\)](#), 113
[to_pitch\(\) \(parselmouth.Sound method\)](#), 113
[to_pitch_ac\(\) \(parselmouth.Sound method\)](#), 113
[to_pitch_cc\(\) \(parselmouth.Sound method\)](#), 113
[to_pitch_shs\(\) \(parselmouth.Sound method\)](#), 113
[to_pitch_spinet\(\) \(parselmouth.Sound method\)](#), 113
[to_sound\(\) \(parselmouth.MFCC method\)](#), 73
[to_sound\(\) \(parselmouth.Spectrogram method\)](#), 125
[to_sound\(\) \(parselmouth.Spectrum method\)](#), 135
[to_sound_hum\(\) \(parselmouth.Pitch method\)](#), 88
[to_sound_pulses\(\) \(parselmouth.Pitch method\)](#), 88
[to_sound_sine\(\) \(parselmouth.Pitch method\)](#), 88
[to_spectrogram\(\) \(parselmouth.Sound method\)](#), 113
[to_spectrogram\(\) \(parselmouth.Spectrum method\)](#), 135
[to_spectrum\(\) \(parselmouth.Sound method\)](#), 113
[to_spectrum_slice\(\) \(parselmouth.Spectrogram method\)](#), 125
[to_tgt\(\) \(parselmouth.TextGrid method\)](#), 139

[total_duration \(parselmouth.CC property\)](#), 37
[total_duration \(parselmouth.Formant property\)](#), 45
[total_duration \(parselmouth.Harmonicity property\)](#), 57
[total_duration \(parselmouth.Intensity property\)](#), 67
[total_duration \(parselmouth.MFCC property\)](#), 74
[total_duration \(parselmouth.Pitch property\)](#), 89
[total_duration \(parselmouth.Sound property\)](#), 115
[total_duration \(parselmouth.Spectrogram property\)](#), 126
[total_duration \(parselmouth.TimeFrameSampled property\)](#), 145
[total_duration \(parselmouth.TimeFunction property\)](#), 149
[trange \(parselmouth.CC property\)](#), 37
[trange \(parselmouth.Formant property\)](#), 45
[trange \(parselmouth.Harmonicity property\)](#), 57
[trange \(parselmouth.Intensity property\)](#), 67
[trange \(parselmouth.MFCC property\)](#), 74
[trange \(parselmouth.Pitch property\)](#), 89
[trange \(parselmouth.Sound property\)](#), 115
[trange \(parselmouth.Spectrogram property\)](#), 126
[trange \(parselmouth.TimeFrameSampled property\)](#), 145
[trange \(parselmouth.TimeFunction property\)](#), 149
[TRIANGULAR \(parselmouth.WindowShape attribute\)](#), 159
[ts\(\) \(parselmouth.CC method\)](#), 36
[ts\(\) \(parselmouth.Formant method\)](#), 44
[ts\(\) \(parselmouth.Harmonicity method\)](#), 56
[ts\(\) \(parselmouth.Intensity method\)](#), 66
[ts\(\) \(parselmouth.MFCC method\)](#), 73
[ts\(\) \(parselmouth.Pitch method\)](#), 88
[ts\(\) \(parselmouth.Sound method\)](#), 113
[ts\(\) \(parselmouth.Spectrogram method\)](#), 125
[ts\(\) \(parselmouth.TimeFrameSampled method\)](#), 144

U

[unvoice\(\) \(parselmouth.Pitch method\)](#), 88
[unvoice\(\) \(parselmouth.Pitch.Frame method\)](#), 85

V

[value \(parselmouth.AmplitudeScaling property\)](#), 31
[value \(parselmouth.CC.FileFormat property\)](#), 34
[value \(parselmouth.Data.FileFormat property\)](#), 38
[value \(parselmouth.Formant.FileFormat property\)](#), 43
[value \(parselmouth.FormantUnit property\)](#), 46
[value \(parselmouth.Function.FileFormat property\)](#), 48
[value \(parselmouth.Harmonicity.FileFormat property\)](#), 53
[value \(parselmouth.Intensity.AveragingMethod property\)](#), 62
[value \(parselmouth.Intensity.FileFormat property\)](#), 63
[value \(parselmouth.Matrix.FileFormat property\)](#), 78
[value \(parselmouth.MFCC.FileFormat property\)](#), 71
[value \(parselmouth.Pitch.FileFormat property\)](#), 85

value (*parselmouth.PitchUnit* property), 91
 value (*parselmouth.Sampled.FileFormat* property), 92
 value (*parselmouth.SampledXY.FileFormat* property), 97
 value (*parselmouth.SignalOutsideTimeDomain* property), 99
 value (*parselmouth.Sound.FileFormat* property), 106
 value (*parselmouth.Sound.ToHarmonicityMethod* property), 107
 value (*parselmouth.Sound.ToPitchMethod* property), 107
 value (*parselmouth.SoundFileFormat* property), 117
 value (*parselmouth.SpectralAnalysisWindowShape* property), 119
 value (*parselmouth.Spectrogram.FileFormat* property), 123
 value (*parselmouth.Spectrum.FileFormat* property), 132
 value (*parselmouth.TextGrid.FileFormat* property), 138
 value (*parselmouth.TimeFrameSampled.FileFormat* property), 143
 value (*parselmouth.TimeFunction.FileFormat* property), 148
 value (*parselmouth.ValueInterpolation* property), 150
 value (*parselmouth.Vector.FileFormat* property), 154
 value (*parselmouth.WindowShape* property), 159
 ValueInterpolation (class in *parselmouth*), 149
 values (*parselmouth.Harmonicity* property), 57
 values (*parselmouth.Intensity* property), 67
 values (*parselmouth.Matrix* property), 80
 values (*parselmouth.Sound* property), 115
 values (*parselmouth.Spectrogram* property), 126
 values (*parselmouth.Spectrum* property), 136
 values (*parselmouth.Vector* property), 157
 Vector (class in *parselmouth*), 151
 Vector.FileFormat (class in *parselmouth*), 153
 VERSION (in module *parselmouth*), 27

W

WAV (*parselmouth.SoundFileFormat* attribute), 117
 WAV_24 (*parselmouth.SoundFileFormat* attribute), 117
 WAV_32 (*parselmouth.SoundFileFormat* attribute), 117
 WELCH (*parselmouth.SpectralAnalysisWindowShape* attribute), 119
 WindowShape (class in *parselmouth*), 157

X

x1 (*parselmouth.CC* property), 37
 x1 (*parselmouth.Formant* property), 45
 x1 (*parselmouth.Harmonicity* property), 57
 x1 (*parselmouth.Intensity* property), 67
 x1 (*parselmouth.Matrix* property), 80
 x1 (*parselmouth.MFCC* property), 74
 x1 (*parselmouth.Pitch* property), 89
 x1 (*parselmouth.Sampled* property), 94

x1 (*parselmouth.SampledXY* property), 98
 x1 (*parselmouth.Sound* property), 115
 x1 (*parselmouth.Spectrogram* property), 126
 x1 (*parselmouth.Spectrum* property), 136
 x1 (*parselmouth.TimeFrameSampled* property), 145
 x1 (*parselmouth.Vector* property), 157
 x_bins() (*parselmouth.CC* method), 36
 x_bins() (*parselmouth.Formant* method), 44
 x_bins() (*parselmouth.Harmonicity* method), 56
 x_bins() (*parselmouth.Intensity* method), 66
 x_bins() (*parselmouth.Matrix* method), 79
 x_bins() (*parselmouth.MFCC* method), 73
 x_bins() (*parselmouth.Pitch* method), 88
 x_bins() (*parselmouth.Sampled* method), 93
 x_bins() (*parselmouth.SampledXY* method), 98
 x_bins() (*parselmouth.Sound* method), 114
 x_bins() (*parselmouth.Spectrogram* method), 125
 x_bins() (*parselmouth.Spectrum* method), 135
 x_bins() (*parselmouth.TimeFrameSampled* method), 144
 x_bins() (*parselmouth.Vector* method), 156
 x_grid() (*parselmouth.CC* method), 36
 x_grid() (*parselmouth.Formant* method), 44
 x_grid() (*parselmouth.Harmonicity* method), 56
 x_grid() (*parselmouth.Intensity* method), 66
 x_grid() (*parselmouth.Matrix* method), 79
 x_grid() (*parselmouth.MFCC* method), 73
 x_grid() (*parselmouth.Pitch* method), 88
 x_grid() (*parselmouth.Sampled* method), 93
 x_grid() (*parselmouth.SampledXY* method), 98
 x_grid() (*parselmouth.Sound* method), 114
 x_grid() (*parselmouth.Spectrogram* method), 125
 x_grid() (*parselmouth.Spectrum* method), 135
 x_grid() (*parselmouth.TimeFrameSampled* method), 144
 x_grid() (*parselmouth.Vector* method), 156
 xmax (*parselmouth.CC* property), 37
 xmax (*parselmouth.Formant* property), 45
 xmax (*parselmouth.Function* property), 49
 xmax (*parselmouth.Harmonicity* property), 57
 xmax (*parselmouth.Intensity* property), 67
 xmax (*parselmouth.Matrix* property), 80
 xmax (*parselmouth.MFCC* property), 74
 xmax (*parselmouth.Pitch* property), 89
 xmax (*parselmouth.Sampled* property), 94
 xmax (*parselmouth.SampledXY* property), 98
 xmax (*parselmouth.Sound* property), 115
 xmax (*parselmouth.Spectrogram* property), 126
 xmax (*parselmouth.Spectrum* property), 136
 xmax (*parselmouth.TextGrid* property), 139
 xmax (*parselmouth.TimeFrameSampled* property), 145
 xmax (*parselmouth.TimeFunction* property), 149
 xmax (*parselmouth.Vector* property), 157
 xmin (*parselmouth.CC* property), 37

[xmin \(parselmouth.Formant property\)](#), 45
[xmin \(parselmouth.Function property\)](#), 49
[xmin \(parselmouth.Harmonicity property\)](#), 57
[xmin \(parselmouth.Intensity property\)](#), 67
[xmin \(parselmouth.Matrix property\)](#), 80
[xmin \(parselmouth.MFCC property\)](#), 74
[xmin \(parselmouth.Pitch property\)](#), 89
[xmin \(parselmouth.Sampled property\)](#), 94
[xmin \(parselmouth.SampledXY property\)](#), 98
[xmin \(parselmouth.Sound property\)](#), 115
[xmin \(parselmouth.Spectrogram property\)](#), 126
[xmin \(parselmouth.Spectrum property\)](#), 136
[xmin \(parselmouth.TextGrid property\)](#), 139
[xmin \(parselmouth.TimeFrameSampled property\)](#), 145
[xmin \(parselmouth.TimeFunction property\)](#), 149
[xmin \(parselmouth.Vector property\)](#), 157
[xrange \(parselmouth.CC property\)](#), 37
[xrange \(parselmouth.Formant property\)](#), 45
[xrange \(parselmouth.Function property\)](#), 49
[xrange \(parselmouth.Harmonicity property\)](#), 57
[xrange \(parselmouth.Intensity property\)](#), 67
[xrange \(parselmouth.Matrix property\)](#), 80
[xrange \(parselmouth.MFCC property\)](#), 74
[xrange \(parselmouth.Pitch property\)](#), 89
[xrange \(parselmouth.Sampled property\)](#), 94
[xrange \(parselmouth.SampledXY property\)](#), 98
[xrange \(parselmouth.Sound property\)](#), 115
[xrange \(parselmouth.Spectrogram property\)](#), 126
[xrange \(parselmouth.Spectrum property\)](#), 136
[xrange \(parselmouth.TextGrid property\)](#), 139
[xrange \(parselmouth.TimeFrameSampled property\)](#), 145
[xrange \(parselmouth.TimeFunction property\)](#), 149
[xrange \(parselmouth.Vector property\)](#), 157
[xs\(\) \(parselmouth.CC method\)](#), 36
[xs\(\) \(parselmouth.Formant method\)](#), 45
[xs\(\) \(parselmouth.Harmonicity method\)](#), 56
[xs\(\) \(parselmouth.Intensity method\)](#), 66
[xs\(\) \(parselmouth.Matrix method\)](#), 80
[xs\(\) \(parselmouth.MFCC method\)](#), 73
[xs\(\) \(parselmouth.Pitch method\)](#), 88
[xs\(\) \(parselmouth.Sampled method\)](#), 93
[xs\(\) \(parselmouth.SampledXY method\)](#), 98
[xs\(\) \(parselmouth.Sound method\)](#), 114
[xs\(\) \(parselmouth.Spectrogram method\)](#), 125
[xs\(\) \(parselmouth.Spectrum method\)](#), 135
[xs\(\) \(parselmouth.TimeFrameSampled method\)](#), 144
[xs\(\) \(parselmouth.Vector method\)](#), 156

Y

[y1 \(parselmouth.Harmonicity property\)](#), 57
[y1 \(parselmouth.Intensity property\)](#), 67
[y1 \(parselmouth.Matrix property\)](#), 80
[y1 \(parselmouth.SampledXY property\)](#), 98
[y1 \(parselmouth.Sound property\)](#), 115

[y1 \(parselmouth.Spectrogram property\)](#), 126
[y1 \(parselmouth.Spectrum property\)](#), 136
[y1 \(parselmouth.Vector property\)](#), 157
[y_bins\(\) \(parselmouth.Harmonicity method\)](#), 56
[y_bins\(\) \(parselmouth.Intensity method\)](#), 66
[y_bins\(\) \(parselmouth.Matrix method\)](#), 80
[y_bins\(\) \(parselmouth.SampledXY method\)](#), 98
[y_bins\(\) \(parselmouth.Sound method\)](#), 114
[y_bins\(\) \(parselmouth.Spectrogram method\)](#), 125
[y_bins\(\) \(parselmouth.Spectrum method\)](#), 135
[y_bins\(\) \(parselmouth.Vector method\)](#), 156
[y_grid\(\) \(parselmouth.Harmonicity method\)](#), 56
[y_grid\(\) \(parselmouth.Intensity method\)](#), 66
[y_grid\(\) \(parselmouth.Matrix method\)](#), 80
[y_grid\(\) \(parselmouth.SampledXY method\)](#), 98
[y_grid\(\) \(parselmouth.Sound method\)](#), 114
[y_grid\(\) \(parselmouth.Spectrogram method\)](#), 125
[y_grid\(\) \(parselmouth.Spectrum method\)](#), 135
[y_grid\(\) \(parselmouth.Vector method\)](#), 156
[ymax \(parselmouth.Harmonicity property\)](#), 57
[ymax \(parselmouth.Intensity property\)](#), 67
[ymax \(parselmouth.Matrix property\)](#), 80
[ymax \(parselmouth.SampledXY property\)](#), 98
[ymax \(parselmouth.Sound property\)](#), 115
[ymax \(parselmouth.Spectrogram property\)](#), 126
[ymax \(parselmouth.Spectrum property\)](#), 136
[ymax \(parselmouth.Vector property\)](#), 157
[ymin \(parselmouth.Harmonicity property\)](#), 57
[ymin \(parselmouth.Intensity property\)](#), 67
[ymin \(parselmouth.Matrix property\)](#), 80
[ymin \(parselmouth.SampledXY property\)](#), 98
[ymin \(parselmouth.Sound property\)](#), 115
[ymin \(parselmouth.Spectrogram property\)](#), 126
[ymin \(parselmouth.Spectrum property\)](#), 136
[ymin \(parselmouth.Vector property\)](#), 157
[yrange \(parselmouth.Harmonicity property\)](#), 58
[yrange \(parselmouth.Intensity property\)](#), 67
[yrange \(parselmouth.Matrix property\)](#), 80
[yrange \(parselmouth.SampledXY property\)](#), 98
[yrange \(parselmouth.Sound property\)](#), 115
[yrange \(parselmouth.Spectrogram property\)](#), 127
[yrange \(parselmouth.Spectrum property\)](#), 136
[yrange \(parselmouth.Vector property\)](#), 157
[ys\(\) \(parselmouth.Harmonicity method\)](#), 56
[ys\(\) \(parselmouth.Intensity method\)](#), 66
[ys\(\) \(parselmouth.Matrix method\)](#), 80
[ys\(\) \(parselmouth.SampledXY method\)](#), 98
[ys\(\) \(parselmouth.Sound method\)](#), 114
[ys\(\) \(parselmouth.Spectrogram method\)](#), 125
[ys\(\) \(parselmouth.Spectrum method\)](#), 135
[ys\(\) \(parselmouth.Vector method\)](#), 156

Z

[ZERO \(parselmouth.SignalOutsideTimeDomain attribute\)](#),

