# Wonderwords

Sep 09, 2020

# Contents

Wonderwords is a python package with a command line interface with the purpose of generating random words and sentences. Wonderwords is **free** and **open source**, meaning anyone can contribute to its repository.

Here is a simple example:

```
>>> from wonderwords import RandomWord
>>>
>>> generator = RandomWord()
>>> generator.word()
'stomach'
```

The random words can also be further customized:

```
>>> generator.word(starts_with="ap", include_parts_of_speech=["adjectives"])
'apathetic'
```

With Wonderwords you can also generate lists of words and sentences, though the sentence feature is in its early development. To install, head over to the *install* page. The *quickstart* is a great place to get started with Wonderwords.

A full reference of all modules (the API documentation) can be found *here*.

CHAPTER 1

# Table of Contents

## 1.1 Install

To install Wonderwords, use your favorite PyPI package manager, such as pip to install the package `wonderwords`:

```
$ pip install wonderwords
```

Once installed, verify that it was correctly installed by going to a python console and typing:

```
>>> import wonderwords
```

If you get a `ModuleNotFoundError`, python cannot find Wonderwords. If you have trouble with installation, create a new issue at the Wonderwords GitHub page.

The command line interface can be accessed with the `wonderwords` command. Verify the command line interface by typing:

```
$ wonderwords -v
```

You should get an output similar to the following:

### 1.1.1 Upgrade

To upgrade Wonderwords to the latest stable version, use:

```
$ pip install --upgrade wonderwords
```

### 1.1.2 Uninstall

To uninstall with pip, use:

```
$ pip uninstall wonderwords
```

## 1.2 Quickstart

Wonderwords is a lightweight, pure python tool that can be used to generate random words and sentences. In this tutorial you will learn the basics of Wonderwords and the command line interface. This tutorial is meant for those who have never used Wonderwords or those who want to learn more about it. For a full reference of all commands, visit the API documentation.

---

**Note:** This tutorial assumes you have already installed Wonderwords. If this is not the case, head over to the *Install* page before proceeding.

---

### 1.2.1 The `RandomWord` class

One of the core Wonderwords classes is the `RandomWord` class. This class encapsulates operations dealing with individual words. One method of this class is the `word` method, which can be used to generate individual random words:

```
>>> from wonderwords import RandomWord
>>>
>>> w = RandomWord()
>>> w.word()
>>> 'sordid'
```

Calling the word class returned a string containing a word. When using Wonderwords, it is helpful to create an instance of the `RandomWord` class in the top-level module of your project and import it when necessary.

#### The `word` Method

What if we want to generate a word that starts with a certain string, say `n`? Here is where the `starts_with` and `ends_with` arguments come into play. For example, to retrieve a word that starts with `"n"` and ends with `"ies"`, we can do the following:

```
>>> w.word(starts_with="n", ends_with="es")
'noodles'
```

You don't have to use both arguments. You can specify either one individually like so:

```
>>> w.word(starts_with="can")
'cannon'
```

Sometimes, however, we may try to look for a pattern that doesn't exist. In that case a `NoWordsToChoseFrom` exception is raised:

```
>>> w.word(starts_with="ja", ends_with="ka")
NoWordsToChoseFrom: There aren't enough words to choose from. Cannot generate 1␣
↪word(s)
```

We can also narrow down a word by part of speech. By default, nouns, verbs and adjectives are all enabled. If you want to generate a word by only a certain part of speech, you can use the `include_parts_of_speech` parameter:

```
>>> w.word(include_parts_of_speech=["adjectives"])
'tough'
>>> w.word(include_parts_of_speech=["nouns", "verbs"])
'cinder'
```

Finally, we can also filter words by length using the `word_min_length` and `word_max_length` parameters:

```
>>> w.word(word_min_length=5)
'documentary'
>>> w.word(word_max_length=3)
'dog'
>>> w.word(word_min_length=9, word_max_length=10)
'velodrome'
```

Remember that we can combine multiple filters together, like so:

```
>>> w.word(
...     word_min_length=4,
...     starts_with="k",
...     include_parts_of_speech=["verbs"]
... )
'keep'
```

### The `filter` method

As you saw above, the word class allows the filtering of many words. What if we want to get a list of all words that match a certain filter? The `filter` method allows us to get all words matching a certain criteria:

```
>>> w.filter(word_min_length=4, starts_with="loc")
['locality',
 'local',
 'locket',
 'location',
 'locomotive',
 'locust',
 'locker',
 'lock',
 'locate']
```

The `filter` method has the same arguments as the `word` method, except it returns **all** matching words, while the `word` method matches a random word fitting the criteria.

### The `random_words` method

The `random_words` methods acts just like the `filter` method, except with two differences:

- You can limit the amount of words fitting the criteria
- If there aren't enough words to reach the limit, a `NoWordsToChoseFrom` exception is raised **unless** `return_less_if_necessary` is set to `True`.

This method is useful if we want to get a list of words:

```
>>> w.random_words(3)
['prince', 'handover', 'cell']
```

(continues on next page)

(continued from previous page)

```
>>> w.random_words(4, word_min_length=5, starts_with="a")
['abrogation', 'animal', 'appropriation', 'angry']
>>> w.random_words(3, word_min_length=5, starts_with="alg") # The exception is
...                                                         # raised as 3 words␣
↪cannot be generated
NoWordsToChoseFrom: There aren't enough words to choose from. Cannot generate 3␣
↪word(s)
>>> w.random_words(3, word_min_length=5, starts_with="alg", return_less_if_
↪necessary=True)
['algebra', 'algorithm']
```

### 1.2.2 The `RandomSentence` class

Wonderwords makes generation of structured sentences made of random words easy. The `RandomSentence` class houses many of these features. You should keep an instance of this class at the top-level of your project just like the `RandomWord` class:

```
>>> from wonderwords import RandomSentence
>>>
>>> s = RandomSentence()
```

#### Creating sentences with the RandomSentence class

The RandomSentence class provides multiple methods to generate random sentences, for example:

```
>>> s.bare_bone_sentence() # generate a bare-bone sentence (The [subject] [predicate])
'The hut frames.'
>>> s.simple_sentence() # generate a simple sentence
'The reprocessing formulates enrollment.'
>>> s.sentence() # a sentence with a subject, predicate, adjective and direct object
'The strong mean shears movement.'
```

As you can see, these sentences have almost no meaning, and are very simple and structured. These sentences are good for creating memorable phrases for your programs.

### 1.2.3 The Wonderwords CLI

Wonderwords also provides a CLI, or *command line interface* which is installed along with the python modules. To use the CLI, open your terminal and type the command `wonderwords`:

```
$ wonderwords
```

When typing the `wonderwords` command, you are greeted with a main page with basic information, such as basic commands and the `wonderwords` version. To get a full list of commands, type `wonderwords -h` or `wonderwords --help`.

#### Generating random words

To generate a random word, use the `-w` or `--word` flag. A random word will be printed to the console:

```
$ wonderwords -w
```

All of the filters that you have learned above have their own commands, too:

```
$ wonderwords -w -sw a -ew e # -sw: starts with, -ew ends with; word that starts with␣
↪a and ends with e
$ wonderwords -p nouns verbs # -p: parts of speech; select only nouns and verbs
$ wonderwords -min 3 -max 5 # -min: minimum length, -max maximum length; minimum␣
↪length 3 and maximum length 5
```

### Generating filters and lists

You can also generate filters with the `-f` flag and lists with the `-l` flag. All modifiers such as `-sw` and `-min` can also be used. Additionally, the `-d` flag can set a delimiter between words:

```
$ wonderwords -f -min 3 # get all words with a minimum length of 3
$ wonderwords -l 5 -sw ap # get 5 words that start with "ap"
$ wonderwords -l 3 -d " | " # get 3 random words separated with " | "
```

### Generating random sentences

The `-s` flag followed by a sentence type can generate a random sentence. The options of type are:

- `bb`: bare-bone sentence
- `ss`: simple sentence
- `bba`: bare-bone sentence with adjective
- `s`: a simple sentence plus and adjective

For example:

```
$ wonderwords -s bb # generate a bare-bone sentence
$ wonderwords -s ss # generate a simple sentence
```

## 1.2.4 And that's it!

The quickstart tutorial has come to an end. In this tutorial, you learned the basics of Wonderwords. More specifically, you learned about:

- **The `RandomWord` class**
  - The `word` method
  - The `filter` method
  - The `random_words` method
- The `RandomSentence` class and some of its methods
- How to use the Wonderwords command line interface

### What's next?

After you have gotten comfortable using wonderwords, you can use the API reference for help on specific classes, and functions. If you want to contribute, please read the contribution guidelines. If you have any problems, bugs, or feature requests, please open up an issue on the Wonderwords GitHub page.

## 1.3 API Documentation

Here you can find a list of Wonderwords modules and their relevant documentations. If you are new to Wonderwords, it is highly suggested that you start with *the quickstart*.

### 1.3.1 wonderwords.random_word

The `random_word` module houses all classes and functions relating to the generation of single random words.

**exception** wonderwords.random_word.**NoWordsToChoseFrom**

Bases: `Exception`

NoWordsToChoseFrom is raised when there is an attempt to access more words than exist. This exception may be raised if the amount of random words to generate is larger than the amount of words that exist.

**class** wonderwords.random_word.**RandomWord**(*nouns: Optional[List[str]] = None*, *verbs: Optional[List[str]] = None*, *adjectives: Optional[List[str]] = None*)

Bases: `object`

The RandomWord class encapsulates multiple methods dealing with the generation of random words and lists of random words.

Example:

```
>>> r = RandomWord(nouns=["apple", "orange"])
>>> r2 = RandomWord()
```

**Parameters**

- **nouns** (*list, optional*) – a list of nouns that will be used to generate random nouns. Defaults to None.
- **verbs** (*list, optional*) – a list of verbs that will be used to generate random verbs. Defaults to None.
- **adjectives** (*list, optional*) – a list of adjectives that will be used to generate random adjectives. Defaults to None.

**filter**(*starts_with: str = ''*, *ends_with: str = ''*, *include_parts_of_speech: Optional[List[str]] = None*, *word_min_length: Optional[int] = None*, *word_max_length: Optional[int] = None*)

Return all existing words that match the criteria specified by the arguments.

Example:

```
>>> # Filter all nouns that start with a:
>>> r.filter(starts_with="a", include_parts_of_speech="noun")
```

**Parameters**

- **starts_with** (*str, optional*) – the string each word should start with. Defaults to "".
- **ends_with** (*str, optional*) – the string each word should end with. Defaults to "".
- **include_parts_of_speech** (*list of strings, optional*) – a list of strings denoting a part of speech. Each word returned will be in the category of at least one part of speech. By default, all parts of speech are enabled. Defaults to None.
- **word_min_length** (*int, optional*) – the minimum length of each word. Defaults to None.
- **word_max_length** (*int, optional*) – the maximum length of each word. Defaults to None.

**Returns** a list of unique words that match each of the criteria specified

**Return type** list of strings

**random_words**(*amount: int = 1*, *starts_with: str = ''*, *ends_with: str = ''*, *include_parts_of_speech: Optional[List[str]] = None*, *word_min_length: Optional[int] = None*, *word_max_length: Optional[int] = None*, *return_less_if_necessary: bool = False*)

Generate a list of n random words specified by the `amount` parameter and fit the criteria specified.

Example:

```
>>> # Generate a list of 3 adjectives or nouns which start with "at"
>>> # and are at least 2 letters long
>>> r.random_words(
...     3,
...     starts_with="at",
...     include_parts_of_speech=["adjectives", "nouns"],
...     word_min_length=2
... )
```

**Parameters**

- **amount** (*int, optional*) – the amount of words to generate. Defaults to 1.
- **starts_with** (*str, optional*) – the string each word should start with. Defaults to "".
- **ends_with** (*str, optional*) – the string each word should end with. Defaults to "".
- **include_parts_of_speech** (*list of strings, optional*) – a list of strings denoting a part of speech. Each word returned will be in the category of at least one part of speech. By default, all parts of speech are enabled. Defaults to None.
- **word_min_length** (*int, optional*) – the minimum length of each word. Defaults to None.
- **word_max_length** (*int, optional*) – the maximum length of each word. Defaults to None.
- **return_less_if_necessary** (*bool, optional*) – if set to True, if there aren't enough words to statisfy the amount, instead of raising a NoWordsToChoseFrom exception, return all words that did statisfy the original query.

**Raises** ***NoWordsToChoseFrom*** – if there are less words to choose from than the amount that was requested, a NoWordsToChoseFrom exception is raised, **unless** return_less_if_necessary is set to True.

**Returns** a list of the words

**Return type** list of strings

**word**(*starts_with: str = ''*, *ends_with: str = ''*, *include_parts_of_speech: Optional[List[str]] = None*, *word_min_length: Optional[int] = None*, *word_max_length: Optional[int] = None*)

Returns a random word that fits the criteria specified by the arguments.

Example:

```
>>> # Select a random noun that starts with y
>>> r.word(ends_with="y", include_parts_of_speech=["nouns"])
```

**Parameters**

- **starts_with** (*str, optional*) – the string each word should start with. Defaults to "".
- **ends_with** (*str, optional*) – the string the word should end with. Defaults to "".
- **include_parts_of_speech** (*list of strings, optional*) – a list of strings denoting a part of speech. The returned will be in the category of at least one part of speech. By default, all parts of speech are enabled. Defaults to None.
- **word_min_length** (*int, optional*) – the minimum length of the word. Defaults to None.
- **word_max_length** (*int, optional*) – the maximum length of the word. Defaults to None.

**Raises** ***NoWordsToChoseFrom*** – if a word fitting the criteria doesn't exist

**Returns** a word

**Return type** str

### 1.3.2 wonderwords.random_sentence

Generate structured sentences in which every word is random.

**class** wonderwords.random_sentence.**RandomSentence**(*nouns: Optional[List[str]] = None*, *verbs: Optional[List[str]] = None*, *adjectives: Optional[List[str]] = None*)

Bases: object

The RandomSentence provides an easy interface to generate structured sentences where each word is randomly generated.

Example:

```
>>> s = RandomSentence(nouns=["car", "cat", "mouse"], verbs=["eat"])
>>> s2 = RandomSentence()
```

**Parameters**

- **nouns** (*list, optional*) – a list of nouns that will be used to generate random nouns. Defaults to None.
- **verbs** (*list, optional*) – a list of verbs that will be used to generate random verbs. Defaults to None.
- **adjectives** (*list, optional*) – a list of adjectives that will be used to generate random adjectives. Defaults to None.

**bare_bone_sentence**()

Generate a bare-bone sentence in the form of `The [subject (noun)] [predicate (verb)].`. For example: `The cat runs.`.

Example:

```
>>> s.bare_bone_sentence()
```

**Returns** string in the form of a bare bone sentence where each word is randomly generated

**Return type** str

**bare_bone_with_adjective**()

Generate a bare-bone sentence with an adjective in the form of: `The [(adjective)] [subject (noun)] [predicate (verb)].`. For example: `The skinny cat reads.`

Example:

```
>>> s.bare_bone_with_adjective()
```

**Returns** a string in the form of a bare-bone sentence with an adjective where each word is randomly generated

**Return type** str

**sentence**()

Generate a simple sentence with an adjective in the form of: `The [(adjective)] [subject (noun)] [predicate (verb)] [direct object (noun)].`. For example: `The green orange likes food.`

Example:

```
>>> s.sentence()
```

**Returns** a string in the form of a simple sentence with an adjective where each word is randomly generated

**Return type** str

**simple_sentence**()

Generate a simple sentence in the form of `The [subject (noun)] [predicate (verb)] [direct object (noun)].`. For example: `The cake plays golf.`

Example:

```
>>> s.simple_sentence()
```

**Returns** a string in the form of a simple sentence where each word is randomly generated

**Return type** str

### 1.3.3 Command Line Interface

The Wonderwords command line interface can be accessed with the `wonderwords` command. Usage:

```
usage: wonderwords [-h] [-w] [-f] [-l LIST] [-s {bb,ss,bba,s}] [-v]
                   [-sw STARTS_WITH] [-ew ENDS_WITH]
                   [-p {nouns,verbs,adjectives} [{nouns,verbs,adjectives} ...]]
                   [-min WORD_MIN_LENGTH] [-max WORD_MAX_LENGTH]
                   [-d DELIMITER]

optional arguments:
  -h, --help            show this help message and exit
  -w, --word, --random-word
                        generate a random word
  -f, --filter          filter a list of words matching the criteria specified
  -l LIST, --list LIST  return a list of words of a certain length
  -s {bb,ss,bba,s}, --sentence {bb,ss,bba,s}
                        return a sentence based on the structure chosen
  -v, --version         Print the version number and exit
  -sw STARTS_WITH, --starts-with STARTS_WITH
                        specify what string the random word(s) should start
                        with
  -ew ENDS_WITH, --ends-with ENDS_WITH
                        specify what string the random word(s) should end with
  -p {nouns,verbs,adjectives} [{nouns,verbs,adjectives} ...], --parts-of-speech
↪{nouns,verbs,adjectives} [{nouns,verbs,adjectives} ...]
                        specify to only include certain parts of speech (by
                        default all parts of speech are included)
  -min WORD_MIN_LENGTH, --word-min-length WORD_MIN_LENGTH
                        specify the minimum length of the word(s)
  -max WORD_MAX_LENGTH, --word-max-length WORD_MAX_LENGTH
                        specify the maximum length of the word(s)
  -d DELIMITER, --delimiter DELIMITER
                        Specify the delimiter to put between a list of words,
                        default is ', '
```

#### Core commands

There are a number of core commands that provide basic functionality:

- `-w` or `--random-word`: generate a random word
- `-f` or `--filter`: return a list of words based on specified criteria
- `-l LIST` or `--list LIST`: much like filter, except you need to specify an integer of the amount of words you want to get
- `-s {bb,ss,bba,s}` or `--sentence {bb,ss,bba,s}` generate a sentence. You must specify the sentence type from the following types:
  - `bb`: bare bone sentence
  - `ss`: simple sentence (bare bone sentence with a direct object)
  - `bba`: bare bone sentence with an adjective
  - `s`: simple sentence with an adjective

- `-v` or `--version`: return the version and exit
- `-h` or `--help`: show a list of commands

### Other commands

The following commands apply only to `random-word`, `filter` and `list`:

- `-sw` or `--starts-with`: the string the word(s) must start with
- `-ew` or `--ends-with`: the string the word(s) must end with
- `-p` or `--parts-of-speech`: only include certain parts of speech, choose one or more from nouns, verbs and adjectives
- `-min` or `--word-min-length`: the minimum length of each word
- `-max` or `--word-max-length`: the maximum length of each word

The following commands apply only to `filter` and `list`:

- `-d DELIMITER` or `--delimiter DELIMITER`: the delimiter used to separate words. Is ", " by default.

### Examples

```
$ wonderwords -w -sw ma -max 5 # choose a random word which starts with "ma" and is a␣
↪max length of 5
$ wonderwords -f -sw a -ew k -p nouns # choose all nouns that start with a, and ends␣
↪with k
$ wonderwords -l 5 -sw n -d " & " # random a list of 5 nouns separated by " & " that␣
↪start with "n"
$ wonderwords -s bb # generate a random bare-bone sentence
$ wonderwords -v # print the version
```

CHAPTER 2

# Indices and tables

- genindex
- *API Documentation*
- modindex
- search

# Python Module Index

## W

# Index