
lorem

Release 1.2.0

Jarry Shaw

Nov 01, 2022

CONTENTS

1	 Lorem Ipsum Generator	1
1.1	Get Random Words	1
1.2	Get Random Sentences	2
1.3	Get Random Paragraphs	3
1.4	Internal utilities	5
2	 Module Unittests	9
3	 Get Random Words	13
4	 Get Random Sentences	15
5	 Get Random Paragraphs	17
6	 Indices and tables	19
	Python Module Index	21
	Index	23

LOREM IPSUM GENERATOR

1.1 Get Random Words

```

lorem.word(pool=('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum', 'commodo', 'consectetur',
               'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor', 'dolore', 'duis', 'ea', 'eiusmod', 'elit', 'enim',
               'esse', 'est', 'et', 'eu', 'ex', 'excepteur', 'exercitation', 'fugiat', 'id', 'in', 'incidunt', 'ipsum', 'irure',
               'labore', 'laboris', 'laborum', 'lorem', 'magna', 'minim', 'mollit', 'nisi', 'non', 'nostrud', 'nulla', 'occaecat',
               'officia', 'pariatur', 'proident', 'qui', 'quis', 'reprehenderit', 'sed', 'sint', 'sit', 'sunt', 'tempor', 'ullamco',
               'ut', 'velit', 'veniam', 'voluptate'), count=1, func=None, args=(), kwargs={})

```

Generate a list of random words.

```

>>> list(itertools.cycle(word(count=3), 3))
['labore', 'tempor', 'commodo']
>>> list(itertools.cycle(word(count=3, func='capitalize'), 3))
['Ea', 'Lorem', 'Do']
>>> list(itertools.cycle(word(count=3, func=lambda s: s.upper()), 3))
['UT', 'AMET', 'EXCEPTEUR']

```

Parameters

- **pool** (*Iterable*[*str*]) – List of words to be used as random word pool.
- **count** (*int*) – Number of non-repeated random words.
- **func** (*Optional*[*str* | *Callable*[[*str*], *str*]]) – Filter function. It can be an attribute name of *str*, or a customised function that takes the original *str* and returns the modified *str*.
- **args** (*tuple*[*str*, ...]) – Additional positional arguments for *func*.
- **kwargs** (*dict*[*str*, *Any*]) – Additional keyword arguments for *func*.

Returns

Indefinite random words generator.

Return type

Iterator[*str*]

```

lorem.get_word(pool=('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum', 'commodo', 'consectetur',
                    'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor', 'dolore', 'duis', 'ea', 'eiusmod', 'elit',
                    'enim', 'esse', 'est', 'et', 'eu', 'ex', 'excepteur', 'exercitation', 'fugiat', 'id', 'in', 'incidunt', 'ipsum',
                    'irure', 'labore', 'laboris', 'laborum', 'lorem', 'magna', 'minim', 'mollit', 'nisi', 'non', 'nostrud',
                    'nulla', 'occaecat', 'officia', 'pariatur', 'proident', 'qui', 'quis', 'reprehenderit', 'sed', 'sint', 'sit',
                    'sunt', 'tempor', 'ullamco', 'ut', 'velit', 'veniam', 'voluptate'), count=1, sep=' ', func=None, args=(),
                    kwargs={})

```

Return random words.

```
>>> get_word(count=3)
'anim voluptate non'
>>> get_word(count=3, func='capitalize')
'Non Labore Ut'
>>> get_word(count=3, func=lambda s: s.upper())
'NISI TEMPOR CILLUM'
```

Parameters

- **pool** (*Iterable*[*str*]) – List of words to be used as random word pool.
- **count** (*int* | *tuple*[*int*, *int*]) – Number of random words. To generate random number of words, supply a 2-element tuple of *int*, the function will use `random.randint()` to choose a random integer as the number of random words.
- **sep** (*str*) – Separator between each word.
- **func** (*Optional*[*str* | *Callable*[[*str*], *str*]]) – Filter function. It can be a function name of *str*, or a customised function that takes the original *str* and returns the modified *str*.
- **args** (*tuple*[*str*, ...]) – Additional positional arguments for *func*.
- **kwargs** (*dict*[*str*, *Any*]) – Additional keyword arguments for *func*.

Returns

Random words.

Return type

str

1.2 Get Random Sentences

`lorem.sentence(pool=('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum', 'commodo', 'consectetur', 'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor', 'dolore', 'duis', 'ea', 'eiusmod', 'elit', 'enim', 'esse', 'est', 'et', 'eu', 'ex', 'excepteur', 'exercitation', 'fugiat', 'id', 'in', 'incidunt', 'ipsum', 'irure', 'labore', 'laboris', 'laborum', 'lorem', 'magna', 'minim', 'mollit', 'nisi', 'non', 'nostrud', 'nulla', 'occaecat', 'officia', 'pariatur', 'proident', 'qui', 'quis', 'reprehenderit', 'sed', 'sint', 'sit', 'sunt', 'tempor', 'ullamco', 'ut', 'velit', 'veniam', 'voluptate'), count=1, comma=(0, 2), word_range=(4, 8))`

Generate a list of random sentences.

```
>>> list(itertools.islice(sentence(), 1))
['Aute irure et commodo sunt do dui dolor.']
```

Parameters

- **pool** (*Iterable*[*str*]) – List of words to be used as random word pool.
- **count** (*int*) – Number of non-repeated random sentences.
- **comma** (*tuple*[*int*, *int*]) – Random range for number of commas. The function will use `random.randint()` to choose a random integer as the number of commas.

- **word_range** (*tuple[int, int]*) – Random range for number of words in each sentence. The function will use `random.randint()` to choose a random integer as the number of words.

Returns

Indefinite random sentence generator.

Return type

Iterator[*str*]

```
lorem.get_sentence(pool=('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum', 'commodo',
                        'consectetur', 'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor', 'dolore', 'duis', 'ea',
                        'eiusmod', 'elit', 'enim', 'esse', 'est', 'et', 'eu', 'ex', 'excepteur', 'exercitation', 'fugiat', 'id', 'in',
                        'incidunt', 'ipsum', 'irure', 'labore', 'laboris', 'laborum', 'lorem', 'magna', 'minim', 'mollit',
                        'nisi', 'non', 'nostrud', 'nulla', 'occaecat', 'officia', 'pariatur', 'proident', 'qui', 'quis',
                        'reprehenderit', 'sed', 'sint', 'sit', 'sunt', 'tempor', 'ullamco', 'ut', 'velit', 'veniam', 'voluptate'),
                    count=1, sep=' ', comma=(0, 2), word_range=(4, 8))
```

Return random sentences.

```
>>> get_sentence()
'Nostrud laboris lorem minim sit culpa, aliqua nostrud in amet, sint pariatur.
↪eiusmod esse.'
```

Parameters

- **pool** (*Iterable[str]*) – List of words to be used as random word pool.
- **count** (*int | tuple[int, int]*) – Number of random sentences. To generate random number of sentences, supply a 2-element tuple of *int*, the function will use `random.randint()` to choose a random integer as the number of random sentences.
- **sep** (*str*) – Separator between each sentence.
- **comma** (*tuple[int, int]*) – Random range for number of commas. The function will use `random.randint()` to choose a random integer as the number of commas.
- **word_range** (*tuple[int, int]*) – Random range for number of words in each sentence. The function will use `random.randint()` to choose a random integer as the number of words.

Returns

Random sentences.

Return type

str

1.3 Get Random Paragraphs

```
lorem.paragraph(pool=('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum', 'commodo',
                      'consectetur', 'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor', 'dolore', 'duis', 'ea',
                      'eiusmod', 'elit', 'enim', 'esse', 'est', 'et', 'eu', 'ex', 'excepteur', 'exercitation', 'fugiat', 'id', 'in',
                      'incidunt', 'ipsum', 'irure', 'labore', 'laboris', 'laborum', 'lorem', 'magna', 'minim', 'mollit', 'nisi',
                      'non', 'nostrud', 'nulla', 'occaecat', 'officia', 'pariatur', 'proident', 'qui', 'quis', 'reprehenderit',
                      'sed', 'sint', 'sit', 'sunt', 'tempor', 'ullamco', 'ut', 'velit', 'veniam', 'voluptate'), count=1, comma=(0,
2), word_range=(4, 8), sentence_range=(5, 10))
```

Generate a list of random paragraphs.

```
>>> list(itertools.islice(paragraph(), 1))
['Aute sint et cupidatat aliquip. Non exercitation est aliquip voluptate '
'fugiat, reprehenderit ad occaecat laboris velit consequat. Magna enim '
'deserunt aute laborum fugiat exercitation. Aliqua ex sunt fugiat in '
'magna voluptate. Elit nisi exercitation nostrud. Magna proident et '
'fugiat eiusmod cupidatat fugiat, sit culpa fugiat non ea eu '
'reprehenderit elit. Proident mollit mollit ut cillum. Nostrud voluptate '
'aliquip cupidatat anim.']
```

Parameters

- **pool** (*Iterable*[*str*]) – List of words to be used as random word pool.
- **count** (*int*) – Number of non-repeated random paragraphs.
- **comma** (*tuple*[*int*, *int*]) – Random range for number of commas. The function will use `random.randint()` to choose a random integer as the number of commas.
- **word_range** (*tuple*[*int*, *int*]) – Random range for number of words in each sentence. The function will use `random.randint()` to choose a random integer as the number of words.
- **sentence_range** (*tuple*[*int*, *int*]) – Random range for number of sentences in each paragraph. The function will use `random.randint()` to choose a random integer as the number of sentences.

Returns

Random paragraph generator.

Return type

Iterator[*str*]

```
lorem.get_paragraph(pool=('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum', 'commodo',
'consectetur', 'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor', 'dolore', 'duis', 'ea',
'eiusmod', 'elit', 'enim', 'esse', 'est', 'et', 'eu', 'ex', 'excepteur', 'exercitation', 'fugiat', 'id', 'in',
'incidunt', 'ipsum', 'irure', 'labore', 'laboris', 'laborum', 'lorem', 'magna', 'minim', 'mollit',
'nisi', 'non', 'nostrud', 'nulla', 'occaecat', 'officia', 'pariatur', 'proident', 'qui', 'quis',
'reprehenderit', 'sed', 'sint', 'sit', 'sunt', 'tempor', 'ullamco', 'ut', 'velit', 'veniam', 'voluptate'),
count=1, sep='\n', comma=(0, 2), word_range=(4, 8), sentence_range=(5, 10))
```

Return random paragraphs.

```
>>> get_paragraph()
'Exercitation magna sunt excepteur irure adipiscing commodo dui. Est '
'ipsum qui deserunt, deserunt nostrud reprehenderit esse. Do velit '
'est in velit sed. Sunt officia officia lorem. Commodore lorem '
'exercitation veniam officia pariatur velit. Deserunt deserunt sed '
'consequat laborum consequat dolor. Et consectetur irure sint elit tempor,'
'est minim nisi eiusmod id excepteur. Minim cillum veniam sed aliquip '
'anim sit, pariatur nostrud ex cillum laboris laborum. Laborum ullamco '
'mollit elit. Amet id incidunt ipsum sed.'
```

Parameters

- **pool** (*Iterable*[*str*]) – List of words to be used as random word pool.

- **count** (*int* | *tuple[int, int]*) – Number of random paragraphs. To generate random number of paragraphs, supply a 2-element tuple of *int*, the function will use *random.randint()* to choose a random integer as the number of random paragraphs.
- **sep** (*str*) – Separator between each paragraph. The default value is OS-dependent as *os.linesep* (*\r\n* on Windows, *\n* on POSIX).
- **comma** (*tuple[int, int]*) – Random range for number of commas. The function will use *random.randint()* to choose a random integer as the number of commas.
- **word_range** (*tuple[int, int]*) – Random range for number of words in each sentence. The function will use *random.randint()* to choose a random integer as the number of words.
- **sentence_range** (*tuple[int, int]*) – Random range for number of sentences in each paragraph. The function will use *random.randint()* to choose a random integer as the number of sentences.

Returns

Random paragraphs.

Return type

str

1.4 Internal utilities

```
lorem._TEXT = ('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum',
'commodo', 'consectetur', 'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor',
'dolore', 'duis', 'ea', 'eiusmod', 'elit', 'enim', 'esse', 'est', 'et', 'eu', 'ex',
'excepteur', 'exercitation', 'fugiat', 'id', 'in', 'incidunt', 'ipsum', 'irure',
'labore', 'laboris', 'laborum', 'lorem', 'magna', 'minim', 'mollit', 'nisi', 'non',
'nostrud', 'nulla', 'occaecat', 'officia', 'pariatur', 'proident', 'qui', 'quis',
'reprehenderit', 'sed', 'sint', 'sit', 'sunt', 'tempor', 'ullamco', 'ut', 'velit',
'veniam', 'voluptate')
```

The original lorem ipsum text pool.

The text pool is generated directly from the original lorem ipsum paragraph:

```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id
est laborum.
```

```
class lorem.LoremGenerator(pool=('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum',
'commodo', 'consectetur', 'consequat', 'culpa', 'cupidatat', 'deserunt', 'do', 'dolor',
'dolore', 'duis', 'ea', 'eiusmod', 'elit', 'enim', 'esse', 'est', 'et', 'eu', 'ex', 'excepteur',
'exercitation', 'fugiat', 'id', 'in', 'incidunt', 'ipsum', 'irure', 'labore', 'laboris',
'laborum', 'lorem', 'magna', 'minim', 'mollit', 'nisi', 'non', 'nostrud', 'nulla',
'occaecat', 'officia', 'pariatur', 'proident', 'qui', 'quis', 'reprehenderit', 'sed', 'sint',
'sit', 'sunt', 'tempor', 'ullamco', 'ut', 'velit', 'veniam', 'voluptate'), dupe=1)
```

Bases: *object*

Generate random words.

Parameters

- **pool** (*Iterable*[*str*]) – List of words to be used as random word pool.
- **dupe** (*int*) – Duplication to generate the word pool.

property pool: *Iterator*[*str*]

Return the current random word pool.

_gen_pool(*dupe=1*)

Generate word pool.

Parameters

- **dupe** (*int*) – Duplication to generate the word pool.

Returns

An infinite loop word pool.

Return type

Iterator[*str*]

gen_word(*func=None, args=(), kwargs={}*)

Generate random word.

Parameters

- **func** (*Optional*[*str* | *Callable*[[*str*], *str*]]) – Filter function. It can be an attribute name of *str*, or a customised function that takes the original *str* and returns the modified *str*.
- **args** (*tuple*[*str*, ...]) – Additional positional arguments for *func*.
- **kwargs** (*dict*[*str*, *Any*]) – Additional keyword arguments for *func*.

Returns

Random word.

Return type

str

gen_sentence(*comma, word_range*)

Generate random sentence.

Parameters

- **comma** (*tuple*[*int*, *int*]) – Random range for number of commas. The function will use *random.randint()* to choose a random integer as the number of commas.
- **word_range** (*tuple*[*int*, *int*]) – Random range for number of words in each sentence. The function will use *random.randint()* to choose a random integer as the number of words.

Returns

Random sentence.

Return type

str

gen_paragraph(*comma, word_range, sentence_range*)

Generate random paragraph.

Parameters

- **comma** (*tuple*[*int*, *int*]) – Random range for number of commas. The function will use `random.randint()` to choose a random integer as the number of commas.
- **word_range** (*tuple*[*int*, *int*]) – Random range for number of words in each sentence. The function will use `random.randint()` to choose a random integer as the number of words.
- **sentence_range** (*tuple*[*int*, *int*]) – Random range for number of sentences in each paragraph. The function will use `random.randint()` to choose a random integer as the number of sentences.

Returns

Random paragraph.

Return type

`str`

MODULE UNITTESTS

Test suite for *lorem* module.

`test_lorem.islice(iterable, stop)`

Wrapper function for `itertools.islice()`.

Parameters

- **iterable** (`Iterator[_T]`) –
- **stop** (`int`) –

Return type

`list[_T]`

`test_lorem.shuffle(x, random=None)`

Mock `random.shuffle()`, but actually do nothing.

Parameters

- **x** (`list[Any]`) –
- **random** (`Optional[Callable[[], float]]`) –

Return type

`None`

`test_lorem.randint(a, b)`

Mock `random.randint()`, but return the lower boundary.

Parameters

- **a** (`int`) –
- **b** (`int`) –

Return type

`int`

`test_lorem.choice_first(seq)`

Mock `random.choice()`, but return the first element.

Parameters

seq (`Sequence[_T]`) –

Return type

`_T`

`test_lorem.choice_last(seq)`

Mock `random.choice()`, but return the last element.

Parameters

`seq` (`Sequence[_T]`) –

Return type

`_T`

`test_lorem.pool(self, dupe=1)`

Mock `lorem.LoremGenerator._gen_pool()`, but return a minimised pool.

Parameters

- `self` (`lorem.LoremGenerator`) –
- `dupe` (`int`) –

Return type

`Iterator[str]`

`class test_lorem.TestLorem(methodName='runTest')`

Bases: `TestCase`

Unittest case for lorem module.

`test_lorem_init()`

Test `lorem.__init__()`.

Return type

`None`

`test_gen_word()`

Test `lorem.LoremGenerator.gen_word()`.

Return type

`None`

`test_gen_sentence()`

Test `lorem.LoremGenerator.gen_sentence()`.

Return type

`None`

`test_gen_paragraph()`

Test `lorem.LoremGenerator.gen_paragraph()`.

Return type

`None`

`test_word()`

Test `lorem.word()`.

Return type

`None`

`test_sentence()`

Test `lorem.sentence()`.

Return type

`None`

test_paragraph()

Test *lorem.paragraph()*.

Return type

None

test_get_word()

Test *lorem.get_word()*.

Return type

None

test_get_sentence()

Test *lorem.get_sentence()*.

Return type

None

test_get_paragraph()

Test *lorem.get_paragraph()*.

Return type

None

In publishing and graphic design, lorem ipsum is a placeholder text commonly used to demonstrate the visual form of a document or a typeface without relying on meaningful content.

The lorem module provides a generic access to generating the lorem ipsum text from its very original text:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Usage of the lorem module is rather simple. Depending on your needs, the lorem module provides generation of *words*, *sentences*, and *paragraphs*.

GET RANDOM WORDS

The `lorem` module provides two different ways for getting random words.

1. `word()` – generate a list of random words

```
def word(count=1, func=None, args=(), kwargs={}) -> Iterator[str]: ...
```

2. `get_word()` – return random words

```
def get_word(count=1, sep=' ', func=None, args=(), kwargs={}) -> str: ...
```


GET RANDOM SENTENCES

The `lorem` module provides two different ways for getting random sentences.

1. `sentence()` – generate a list of random sentences

```
def sentence(count=1, comma=(0, 2), word_range=(4, 8)) -> Iterator[str]: ...
```

2. `get_sentence()` – return random sentences

```
def get_sentence(count=1, sep=' ', comma=(0, 2), word_range=(4, 8)) -> Union[str]: .  
↪ ...
```


GET RANDOM PARAGRAPHS

The `lorem` module provides two different ways for getting random paragraphs.

1. `paragraph()` – generate a list of random paragraphs

```
def paragraph(count=1, comma=(0, 2), word_range=(4, 8), sentence_range=(5, 10)) -> ↪
↪Iterator[str]: ...
```

2. `get_paragraph()` – return random paragraphs

```
def get_paragraph(count=1, sep=os.linesep, comma=(0, 2), word_range=(4, 8), ↪
↪sentence_range=(5, 10)) -> Union[str]: ...
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

t

test_lorem, 9

Symbols

`_TEXT` (in module *lorem*), 5
`_gen_pool()` (*lorem.LoremGenerator* method), 6

C

`choice_first()` (in module *test_lorem*), 9
`choice_last()` (in module *test_lorem*), 9

G

`gen_paragraph()` (*lorem.LoremGenerator* method), 6
`gen_sentence()` (*lorem.LoremGenerator* method), 6
`gen_word()` (*lorem.LoremGenerator* method), 6
`get_paragraph()` (in module *lorem*), 4
`get_sentence()` (in module *lorem*), 3
`get_word()` (in module *lorem*), 1

I

`islice()` (in module *test_lorem*), 9

L

LoremGenerator (class in *lorem*), 5

M

module
 test_lorem, 9

P

`paragraph()` (in module *lorem*), 3
`pool` (*lorem.LoremGenerator* property), 6
`pool()` (in module *test_lorem*), 10

R

`randint()` (in module *test_lorem*), 9

S

`sentence()` (in module *lorem*), 2
`shuffle()` (in module *test_lorem*), 9

T

`test_gen_paragraph()` (*test_lorem.TestLorem* method), 10

`test_gen_sentence()` (*test_lorem.TestLorem* method), 10

`test_gen_word()` (*test_lorem.TestLorem* method), 10
`test_get_paragraph()` (*test_lorem.TestLorem* method), 11

`test_get_sentence()` (*test_lorem.TestLorem* method), 11

`test_get_word()` (*test_lorem.TestLorem* method), 11

test_lorem
 module, 9

`test_lorem_init()` (*test_lorem.TestLorem* method), 10

`test_paragraph()` (*test_lorem.TestLorem* method), 10

`test_sentence()` (*test_lorem.TestLorem* method), 10

`test_word()` (*test_lorem.TestLorem* method), 10

TestLorem (class in *test_lorem*), 10

W

`word()` (in module *lorem*), 1