lorem

Release 1.1.1

Jarry Shaw

Nov 21, 2020

CONTENTS:

CHAPTER

ONE

LOREM IPSUM GENERATOR

1.1 Module contents

1.1.1 Get Random Words

lorem.**word**(*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

- **count** (*int*) – Number of non-repeated random words.
- **func** (Optional[Union[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**(*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

- **count** (`Union[int, Tuple[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*) – Seperator between each word.
- **func** (`Optional[Union[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`

1.1.2 Get Random Sentences

`lorem.`**sentence**(*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 duis dolor.']
```

Parameters

- **count** (*int*) – Number of non-repeated random sentences.
- **comma** (`Tuple[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]`) – 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**(*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

- **count** (`Union[int, Tuple[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*) – Seperator between each sentence.
- **comma** (`Tuple[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]) – 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

1.1.3 Get Random Paragraphs

lorem.**paragraph**(*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

- **count** (*int*) – Number of non-repeated random paragraphs.
- **comma** (Tuple[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]) – 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]) – 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**(*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 duis. Est '
'ipsum qui deserunt, deserunt nostrud reprehenderit esse. Do velit '
'est in velit sed. Sunt officia officia lorem. Commodo 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 incididunt ipsum sed.'
```

Parameters

- **count** (Union[int, Tuple[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*) – Seperator between each paragraph. The default value is OS-dependent as `os.linsep` (`\r\n` on Windows, `\n` on POSIX).
- **comma** (`Tuple[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]`) – 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]`) – 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`

1.1.4 Customise Word Pool

`lorem.`**`set_pool`**(*pool*)

Customise random word pool.

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

1.2 Internal utilities

`lorem.`**`_TEXT = ('ad', 'adipiscing', 'aliqua', 'aliquip', 'amet', 'anim', 'aute', 'cillum',`**

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.
```

`lorem.`**`_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]`

`lorem.`**`_gen_word`**(*pool*, *func=None*, *args=()*, *kwargs={}*)

Generate random word.

Parameters

- **pool** (`Iterator[str]`) – Word pool, returned by *`_gen_pool()`*.
- **func** (`Optional[Union[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`

`lorem._gen_sentence`(*pool*, *comma*, *word_range*)
Generate random sentence.

Parameters

- **pool** (`Iterator[str]`) – Word pool, returned by *_gen_pool()*.
- **comma** (`Tuple[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]`) – 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`

`lorem._gen_paragraph`(*pool*, *comma*, *word_range*, *sentence_range*)
Generate random paragraph.

Parameters

- **pool** (`Iterator[str]`) – Word pool, returned by *_gen_pool()*.
- **comma** (`Tuple[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]`) – 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]`) – 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`

CHAPTER

TWO

MODULE UNITTESTS

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*.

CHAPTER

THREE

GET RANDOM WORDS

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

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

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

2. *`get_word()`* – return random words

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

CHAPTER

FOUR

GET RANDOM SENTENCES

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

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

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

2. *`get_sentence()`* – return random sentences

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

CHAPTER

FIVE

GET RANDOM PARAGRAPHS

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

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

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

2. *`get_paragraph()`* – return random paragraphs

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

CHAPTER

SIX

CUSTOMISE WORD POOL

If wanted, the `lorem` module also provides an interface to customise the word pool as you wish.

1. *`set_pool()`* – customise random word pool

```
set_pool(pool)
```

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

G

P

S

W