python-utils Documentation

Release

Jack Evans

Sep 27, 2017

Contents

CHAPTER 1

Core

basic_utils.core.**slurp**(*fname*)
Reads a file and all its contents, returns a single string

Return type str

basic_utils.core.**clear**()
Clears the terminal screen from python, operating system agnostic

Return type None

basic_utils.core.**to_string**(*objects*, *sep=', '*)
Converts a list of objects into a single string

```
>>> to_string([1, 2, 3])
'1, 2, 3'
```

Return type str

basic_utils.core.**getattrs**(*obj*, *keys*)
Supports getting multiple attributes from a model at once

Return type Tuple[Any, ...]

basic_utils.core.**map_getattr**(*attr*, *object_seq*)
Returns a map to retrieve a single attribute from a sequence of objects

Return type Tuple[Any, ...]

basic_utils.core.**rgetattr**(*obj*, *attrs*)
Get a nested attribute within an object

Return type Any

basic_utils.core.**rsetattr**(*obj*, *attr*, *val*)
Sets a nested attribute within an object

Return type None

CHAPTER 2

Date Helpers

basic_utils.date_helpers.**dates_between**(*start*, *end*)
Returns lazy sequence of dates between a start/end point

Return type Iterator[datetime]

CHAPTER 3

Dict Helpers

basic_utils.dict_helpers.**get_keys**(*d*, *keys*, *default=None*)

Returns multiple values for keys in a dictionary

Empty key values will be None by default

```
>>> d = {'x': 24, 'y': 25}
>>> get_keys(d, ('x', 'y', 'z'))
(24, 25, None)
```

Return type Tuple

basic_utils.dict_helpers.**get_in_dict**(*d*, *keys*)

Retrieve nested key from dictionary

```
>>> d = {'a': {'b': {'c': 3}}}
>>> get_in_dict(d, ('a', 'b', 'c'))
3
```

Return type Any

basic_utils.dict_helpers.**set_in_dict**(*d*, *keys*, *value*)

Sets a value inside a nested dictionary

```
>>> d = {'a': {'b': {'c': 3}}}
>>> set_in_dict(d, ('a', 'b', 'c'), 10)
>>> d
{'a': {'b': {'c': 10}}}
```

Return type None

basic_utils.dict_helpers.**prune_dict**(*d*)

Returns new dictionary with falesly values removed.

```
>>> prune_dict({'a': [], 'b': 2, 'c': False})
{'b': 2}
```

Return type dict

CHAPTER 4

Seq Helpers

basic_utils.seq_helpers.**first**(*seq*)

Returns first element in a sequence.

```
>>> first([1, 2, 3])
1
```

Return type Any

basic_utils.seq_helpers.**last**(*seq*)

Returns the last item in a Sequence

```
>>> last([1, 2, 3])
3
```

Return type Any

basic_utils.seq_helpers.**butlast**(*seq*)

Returns all but the last item in sequence

```
>>> butlast([1, 2, 3])
[1, 2]
```

Return type Sequence[+T_co]

basic_utils.seq_helpers.**rest**(*seq*)

Returns remaining elements in a sequence

```
>>> rest([1, 2, 3])
[2, 3]
```

Return type Any

basic_utils.seq_helpers.**reverse**(*seq*)
Returns sequence in reverse order

```
>>> reverse([1, 2, 3])
[3, 2, 1]
```

Return type Sequence[+T_co]

basic_utils.seq_helpers.**cons**(*item*, *seq*)
Adds item to beginning of sequence.

```
>>> list(cons(1, [2, 3]))
[1, 2, 3]
```

Return type chain

basic_utils.seq_helpers.**flatten**(*seq*)
Returns a flatten version of sequence.

```
>>> flatten([1, [2, [3, [4, 5], 6], 7]])
[1, 2, 3, 4, 5, 6, 7]
```

Return type Iterable[+T_co]

basic_utils.seq_helpers.**partial_flatten**(*seq*)
Returns partially flattened version of sequence.

```
>>> partial_flatten(((1,), [2, 3], (4, [5, 6])))
(1, 2, 3, 4, [5, 6])
```

Return type Iterable[+T_co]

basic_utils.seq_helpers.**sorted_index**(*seq*, *item*, *key=None*)

```
>>> sorted_index([10, 20, 30, 40, 50], 35)
3
```

Return type int

basic_utils.seq_helpers.**dedupe**(*seq*, *key=None*)
Removes duplicates from a sequence while maintaining order

```
>>> dedupe([1, 5, 2, 1, 9, 1, 5, 10])
[1, 5, 2, 9, 10]
```

Return type Iterable[+T_co]

basic_utils.seq_helpers.**concat**(*seqX*, *seqY*)
Joins two sequences together, returning a single combined sequence. Preserves the type of passed arguments.

```
>>> concat((1, 2, 3), (4, 5, 6))
(1, 2, 3, 4, 5, 6)
```

Return type Sequence[+T_co]

basic_utils.seq_helpers.**take**(*n*, *iterable*)
Return first n items of the iterable as a list.

```
>>> take(2, range(1, 10))
[1, 2]
```

Return type Iterable[+T_co]

basic_utils.seq_helpers.**nth**(*iterable*, *n*, *default=None*)
Returns the nth item or a default value.

```
>>> nth([1, 2, 3], 1)
2
```

Return type Any

basic_utils.seq_helpers.**all_equal**(*iterable*)
Returns True if all the elements are equal to each other.

```
>>> all_equal([True, True])
True
```

Return type bool

basic_utils.seq_helpers.**quantify**(*iterable*, *pred=<class 'bool'>*)
Returns count of how many times the predicate is true.

```
>>> quantify([True, False, True])
2
```

Return type int

basic_utils.seq_helpers.**head**(*seq*)
Returns first element in a sequence.

```
>>> first([1, 2, 3])
1
```

Return type Any

basic_utils.seq_helpers.**tail**(*seq*)
Returns remaining elements in a sequence

```
>>> rest([1, 2, 3])
[2, 3]
```

Return type Any

basic_utils.seq_helpers.**init**(*seq*)
Returns all but the last item in sequence

```
>>> butlast([1, 2, 3])
[1, 2]
```

Return type Sequence[+T_co]

CHAPTER 5

Primitives

basic_utils.primitives.**natural_nums**(*start=0*, *end=None*)

Yields a lazy sequence of natural numbers

```
>>> from itertools import islice
>>> list(islice(natural_nums(5), 3))
[5, 6, 7]
```

Return type Iterator[int]

basic_utils.primitives.**identity**(*x*)

Returns the same values passed as arguments

```
>>> x = (10, 20)
>>> identity(x)
(10, 20)
```

Return type Any

basic_utils.primitives.**comp**(**funcs*)

Takes a set of functions and returns a fn that is the composition of those functions

Return type Callable

basic_utils.primitives.**complement**(*fn*)

Takes a function fn and returns a function that takes the same arguments as fn with the opposite truth value.

```
>>> not_five = complement(lambda x: x == 5)
>>> not_five(6)
True
```

Return type Callable

basic_utils.primitives.**inc**(*n*)

Increments n by 1

```
>>> inc(10)
11
```

Return type int

basic_utils.primitives.**dec**(*n*)

Decrements n by 1

```
>>> dec(5)
4
```

Return type int

basic_utils.primitives.**even**(*n*)

Returns true if n is even

```
>>> even(2)
True
```

Return type bool

basic_utils.primitives.**odd**(*n*)

Returns true if n is odd

```
>>> even(3)
False
```

Return type bool

basic_utils.primitives.**compose**(**funcs*)

Takes a set of functions and returns a fn that is the composition of those functions

Return type Callable

Python Module Index

b

Index