cardinality

Release 0.1.1

February 07, 2015

Contents

`cardinality` is a small Python library to determine and check the size of any iterable (lists, iterators, generators, and so on). It is intended as a useful companion to the built-in `itertools` module.

CHAPTER 1

Installation

```
pip install cardinality
```

CHAPTER 2

Usage

The `cardinality` module provides these functions:

- `cardinality.count()`
- `cardinality.at_least()`
- `cardinality.at_most()`
- `cardinality.between()`

The API docs below contain usage examples for each function.

Note: Each function creates an iterator for the iterable that is passed to it, and consumes that iterator, but not necessarily until exhaustion. If an argument can be iterated over only once (e.g. generators), do not use the object afterwards.

CHAPTER 3

API

cardinality.**count**(*iterable*)

Count the number of items that *iterable* yields.

Equivalent to the expression

```
len(iterable)
```

... but it also works for iterables that do not support `len()`.

```
>>> import cardinality
>>> cardinality.count([1, 2, 3])
3
>>> cardinality.count(i for i in range(500))
500
>>> def gen():
...     yield 'hello'
...     yield 'world'
>>> cardinality.count(gen())
2
```

cardinality.**at_least**(*size*, *iterable*)

Check whether *iterable* yields at least *size* items.

Equivalent to the expression

```
cardinality.count(iterable) >= size
```

... but more efficient.

```
>>> import cardinality
>>> cardinality.at_least(3, range(2))
False
>>> cardinality.at_least(3, range(5))
True
>>> def gen():
...     yield 'hello'
...     yield 'world'
>>> cardinality.at_least(2, gen())
True
```

cardinality.**at_most**(*size*, *iterable*)

Check whether *iterable* yields no more than *size* items.

Equivalent to the expression

```
cardinality.count(iterable) <= size
```

... but more efficient.

```
>>> import cardinality
>>> cardinality.at_most(3, range(2))
True
>>> cardinality.at_most(3, range(5))
False
>>> def gen():
...     yield 'hello'
...     yield 'world'
>>> cardinality.at_most(1, gen())
False
```

cardinality.**between**(*min*, *max*, *iterable*)

Check whether *iterable* yields at least *min* and at most *max* items.

Equivalent to the expression

```
min <= cardinality.count(iterable) <= max
```

... but more efficient.

```
>>> import cardinality
>>> cardinality.between(4, 6, range(5))
True
>>> cardinality.between(4, 6, range(20))
False
>>> def gen():
...     yield 'hello'
...     yield 'world'
>>> cardinality.between(0, 3, gen())
True
```

CHAPTER 4

Contributing

The source code and issue tracker for this package can be found on Github:

https://github.com/wbolster/cardinality

CHAPTER 5

License

A

B

C