

Basic Python cheat sheet

LISTS

`l.pop(3)` - Returns the fourth item from `l` and deletes it from the list

`l.remove(x)` - Removes the first item in `l` that is equal to `x`

`l.reverse()` - Reverses the order of the items in `l` `l[1::2]` - Returns every second item from `l`, commencing from the 1st item

`l[-5:]` - Returns the last 5 items from `l` specific axis

STRINGS

`s.lower()` - Returns a lowercase version of `s`

`s.title()` - Returns `s` with the first letter of every word capitalized

`"23".zfill(4)` - Returns `"0023"` by left-filling the string with 0's to make it's length 4.

`s.splitlines()` - Returns a list by splitting the string on any newline characters.

Python strings share some common methods with lists

`s[:5]` - Returns the first 5 characters of `s` `"fri" + "end"` - Returns `"friend"`

`"end" in s` - Returns `True` if the substring `"end"` is found in `s`.

RANGE:

Range objects are useful for creating sequences of integers for looping.

`range(5)` - Returns a sequence from 0 to 4

`range(2000,2018)` - Returns a sequence from 2000 to 2017

`range(0,11,2)` - Returns a sequence from 0 to 10, with each item incrementing by 2

`range(0,-10,-1)` - Returns a sequence from 0 to -9 `list(range(5))` - Returns a list from 0 to 4

List Comprehension

A one-line expression of a for loop

`[i ** 2 for i in range(10)]` - Returns a list of the squares of values from 0 to 9

`[s.lower() for s in l_strings]` - Returns the list `l_strings`, with each item having had the `.lower()` method applied

`[i for i in l_floats if i < 0.5]` - Returns the items from `l_floats` that are less than 0.5

Random:

`import random` - Import the random module

`random.random()` - Returns a random float between 0.0 and 1.0

`random.randint(0,10)` - Returns a random integer between 0 and 10

`random.choice(l)` - Returns a random item from the list l

itertools Module

The itertools module is a collection of tools intended to be fast and use memory efficiently when handling iterators (like lists or dictionaries).

`accumulate()`

Makes an iterator that returns the results of a function.

```
>>> data = [1, 2, 3, 4, 5]
```

```
>>> result = itertools.accumulate(data, operator.mul)
```

```
>>> for each in result:
```

```
>>>     print(each)
```

```
1
```

```
2
```

```
6
```

```
24
```

```
120
```

The `operator.mul` takes two numbers and multiplies them

```
operator.mul(1, 2)
```

```
2
```

```
operator.mul(2, 3)
```

```
6
```

```
operator.mul(6, 4)
```

```
24
```

```
operator.mul(24, 5)
```

```
120
```

Passing a function is optional:

```
data = [5, 2, 6, 4, 5, 9, 1]
```

```
>>> result = itertools.accumulate(data)
```

```
>>> for each in result:
```

```
>>>     print(each)
```

```
5
```

```
7
```

```
13
```

```
17
```

```
22
```

```
31
```

```
32
```

If no function is designated the items will be summed:

```
5
```

```
5 + 2 = 7
```

```
7 + 6 = 13
```

```
13 + 4 = 17
```

```
17 + 5 = 22
```

```
22 + 9 = 31
```

```
31 + 1 = 32
```

```
combinations()
```

Takes an iterable and a integer. This will create all the unique combination that have r members.

```
itertools.combinations(iterable, r)
```

```
>>> shapes = ['circle', 'triangle', 'square',]
```

```
>>> result = itertools.combinations(shapes, 2)
```

```
>>> for each in result:
```

```
>>>     print(each)
```

```
('circle', 'triangle')
```

```
('circle', 'square')
```

```
('triangle', 'square')
```