Cheatsheet

Casting

Casting is the native Python way of transforming objects to other types of objects. Example:

```
My_number = 3

my_float = float(My_number)
```

Casting can happen between many different types of objects

Common whitespace characters

Character	Meaning
\t	tab
\s	whitespace character. Different languages and editors may have different protocol for what is included in this designation
\n	newline, often used on Mac and Linux systems, may not be read properly by some PC programs
\r	newline, often used on PCs. May not be read properly by Mac or Linux

open()

Opens a file as an object in memory.

Example:

```
f = open('filename', 'mode')
f.open()
```

File Modes

Mode	Meaning
w	write mode - can write content to this file
r	Read - can read from (but not write to) this file
a	Append - content added goes to the end of the file, as opposed to overwriting the content in the file
r+	Mac and Linux - open the file for reading + writing
rw	Windows - open for reading + writing

Dictionary

Dictionaries are key : value pairs. Keys must be uniqe

Example:

```
my_dictionary = {}
#This initializes an empty dictionary.

my_dictionary['April'] = "Wright"
#This adds my first name as a key and my last name as a value.

my_dictionary['April']
#This calls the vlaue of my dictionary from my key, April.
```

Comprehensions

These are compact ways to iterate through items and populate lists.

Example:

```
list1  = [7, 8,9,10]
#This is a regular list declaration, as we have done before.

list2 = [x+1 for x in list1]
#This is a comprehension. list2 is declared and populated in one line.
```

with

With statements can be used to execute a process, and do any necessary clean-up at the end.

Example:

```
with open("filename") as f:
#This opens filename and tags it to the variable f
        print [line for line in f]
#this uses a list comprehension to print the file to the screen
```