Python 3 - the basics Cheat Sheet

by sean via cheatography.com/22969/cs/4904/

NUMBERS

max()	returns largest argument of list
min()	returns smallest argument of list
abs()	returns absolute value of argument
type()	return data type of argument

STRINGS

Create Strings

'string'	creates string
"string"	creates string

String Methods

print( )	("hello"), (variable)

```
g = "Golf"
h = "Hotel"
print("%s, %s" % (g, h))
print("My name is {0}".format('Fred'))
```

str(3)	returns "3", not #3
len("string")	returns 5
"string".upper()	returns 'STRING'
"STRING".lower()	returns 'string'
word[0]	returns "w"
word[1:len(word)]	returns "ord"

LOOPING

For loops `for variable in sequence :`

```
for x in range(1, 10):
    print(x)
```

The variable will iterate through each item in the sequence.

If `if condition :`

```
if x == 7:
    print(x)
```

*The following lines are only executed if the **condition** is true. The **condition** can be either a single comparison, eg. 5 > 3, or multiple comparisons using Boolean operators to evaluate ultimate True or False eg. 1<2 **and** 2<3 (returns True, so the following lines are evaluated)*

LOOPING (cont)

elif `elif condition :`

*provides second **if** test, when first **if** test is false*:

```
elif x == 5:
    print(x)
```

else *(runs if the **if**/**elif** test is false)*

```
else:
    print(y)
```

while `while condition :`

Remains in the loop while the condition is true. Use the "break" command to leave the loop. This is useful if waiting on an input from microcontroller. Once the microcontroller input is received, the break command kicks out of the loop and continues with the program.

FUNCTION STRUCTURE

```
def shut_down(s):
    if s == "yes":
        return "shutting down"
    elif s == "no":
        return "Shutdown aborted"
    else:
        return "Sorry"
--------------------------------
def is_numeric(num):
    return type(num) == int or
           type(num) == float:
--------------------------------
def finish_game(score):
    tickets = 10 * score
    if score >= 10:
        tickets += 50
    elif score >= 7:
        tickets += 20
    return tickets
```

IMPORTING MODULES & FUNCTIONS

import math

makes math module available
usage: math.sqrt()

from math **import** *

imports **all** math functions into program
usage: sqrt()

from math **import** sqrt

imports **only** the math function specified
usage: sqrt()

READING & WRITING FILES

`input = open(file)`	variable to hold file
`indata = input.read()`	reads the file
`output = open(to_file, 'w')`	open file to write to, 'w' makes file writable
`output.write(in data)`	writes data to `to_file`
`output.close()`	finish by closing `to_file`
`input.close()`	finish by closing `file`

example: exercise 17 in the book Python The Hard Way (page 43).

LISTS

list_name = [item_1, item_2]

Items can be 'strings', variables, integers, etc.

List Index

use the index to access specific locations
usage: `list_name[0] #first entry`

LISTS (cont)

Add to Lists: *.append*

use .append to add a new entry to the end of the List; note the use of ( )

```
list_name.append('string')
```

adds 'string' to end of the list contents

Add to Lists: *assignment*

replace an item in a List:

```
list_name[4] = 'string'
```

replaces 5th item in list with 'string'

Adding Lists together

```
list_A = [1,2,3]
list_B = [4,5]
list_C = list_A + list_B
print(list_C)    #returns
[1,2,3,4,5]
```

len(*list_name***)**

returns the number of items in the list

print (*list_name***)**

returns `[item_1, item_2, etc.]`

List Slicing

```
letters = ['a', 'b', 'c', 'd']
slice = letters[1:3]
print slice #returns ['b', 'c']
```

`[1:3]` means this will return entries starting at index 1 and continue up to, *but not including* the third index position

List Slicing with Strings

strings are considered to be natural lists, with each letter being an item of the list

```
animal = 'blueWhale'
print = (animal[4:])    #returns
Whale
print = (animal[:4])  #returns blue
```

List Manipulation *.index*

```
letters = ['a', 'b', 'c']
print(letters.index('b'))    #prints
1
```

LISTS (cont)

List Manipulation *.sort*

```
numbers = [5, 3, 7]
print(numbers.sort())    #prints
[3,5,7]
```

List Manipulation *.pop*

```
numbers = [5, 3, 7]
print(numbers.pop())  #no index pops
last item
  #prints 7, and removes it from the
list
print(numbers.pop(1))  #pops second
item
  #prints 3, and removes it from the
list
```

List Manipulation *.insert*

```
numbers = [5, 3, 7]
print(numbers.insert(1,9))   #prints
[5,9,3,7]
```

inserts the number 9 "before" position 1 in the list

Dictionaries

```
ph_numbers = {'Jack':x123,
'Mark':x655}
ph_numbers['Mark']  #returns x655
ph_numbers['Mark'] = x899  #assigns
new number
ph_numbers  #returns {'Jack':x123,
'Mark'=x899}
```

A dictionary is comprised of both a "key" and a "value". You access or reassign a particular value by using the key, NOT the position. The dictionary does not maintain a specific order to any of its entries.