Python basics

Credits to Derek Banas @ http://www.newthinktank.com/

```
# Hello world is just one line of code
# print() outputs data to the screen
print("Hello World")

# A variable is a place to store values
# Its name is like a label for that value
name = "Derek"
print(name)


# A variable name can contain letters, numbers, or _
# but can't start with a number

# There are 5 data types Numbers, Strings, List, Tuple, Dictionary
# You can store any of them in the same variable

name = 15
print(name)

# The arithmetic operators +, -, *, /, %, **, //
# ** Exponential calculation
# // Floor Division
print("5 + 2 =", 5+2)
print("5 - 2 =", 5-2)
print("5 * 2 =", 5*2)
print("5 / 2 =", 5/2)
print("5 % 2 =", 5%2)
print("5 ** 2 =", 5**2)
print("5 // 2 =", 5//2)

# Order of Operation states * and / is performed before + and -

print("1 + 2 - 3 * 2 =", 1 + 2 - 3 * 2)
print("(1 + 2 - 3) * 2 =", (1 + 2 - 3) * 2)

# A string is a string of characters surrounded by " or '
# If you must use a " or ' between the same quote escape it with \
quote = "\"Always remember your unique,"

# A multi-line quote
multi_line_quote = ''' just
like everyone else" '''
```

```
print(quote + multi_line_quote)

# To embed a string in output use %s
print("%s %s %s" % ('I like the quote', quote, multi_line_quote))

# To keep from printing newlines use end=""
print("I don't like ",end="")
print("newlines")

# You can print a string multiple times with *
print('\n' * 5)
```

Lists

```
# LISTS -------------
# A list allows you to create a list of values and manipulate them
# Each value has an index with the first one starting at 0

grocery_list = ['Juice', 'Tomatoes', 'Potatoes', 'Bananas']
print('The first item is', grocery_list[1])

# You can change the value stored in a list box
grocery_list[0] = "Green Juice"
print(grocery_list)

# You can get a subset of the list with [min:up to but not including max]

print(grocery_list[1:3])

# You can put any data type in a a list including a list
other_events = ['Wash Car', 'Pick up Kids', 'Cash Check']
to_do_list = [other_events, grocery_list]

print(to_do_list)

# Get the second item in the second list (Boxes inside of boxes)
print(to_do_list[1][1])

# You add values using append
grocery_list.append('onions')
print(to_do_list)

# Insert item at given index
grocery_list.insert(1, "Pickle")
```

```
# Remove item from list
grocery_list.remove("Pickle")

# Sorts items in list
grocery_list.sort()

# Reverse sort items in list
grocery_list.reverse()

# del deletes an item at specified index
del grocery_list[4]
print(to_do_list)

# We can combine lists with a +
to_do_list = other_events + grocery_list
print(to_do_list)

# Get length of list
print(len(to_do_list))

# Get the max item in list
print(max(to_do_list))

# Get the minimum item in list
print(min(to_do_list))
```

Tuples

```
# TUPLES -------------
# Values in a tuple can't change like lists

pi_tuple = (3, 1, 4, 1, 5, 9)

# Convert tuple into a list
new_tuple = list(pi_tuple)

# Convert a list into a tuple
# new_list = tuple(grocery_list)

# tuples also have len(tuple), min(tuple) and max(tuple)
```

Dictionaries

```
# DICTIONARY or MAP --------------
# Made up of values with a unique key for each value
# Similar to lists, but you can't join dicts with a +
```

```
super_villains = {'Fiddler' : 'Isaac Bowin',
                  'Captain Cold' : 'Leonard Snart',
                  'Weather Wizard' : 'Mark Mardon',
                  'Mirror Master' : 'Sam Scudder',
                  'Pied Piper' : 'Thomas Peterson'}

print(super_villains['Captain Cold'])

# Delete an entry
del super_villains['Fiddler']
print(super_villains)

# Replace a value
super_villains['Pied Piper'] = 'Hartley Rathaway'

# Print the number of items in the dictionary
print(len(super_villains))

# Get the value for the passed key
print(super_villains.get("Pied Piper"))

# Get a list of dictionary keys
print(super_villains.keys())

# Get a list of dictionary values
print(super_villains.values())
```

Conditionals

```
# CONDITIONALS -------------
# The if, else and elif statements are used to perform different
# actions based off of conditions
# Comparison Operators : ==, !=, >, <, >=, <=

# The if statement will execute code if a condition is met
# White space is used to group blocks of code in Python
# Use the same number of proceeding spaces for blocks of code

age = 30
if age > 16 :
    print('You are old enough to drive')

# Use an if statement if you want to execute different code regardless
# of whether the condition ws met or not
```

```
if age > 16 :
    print('You are old enough to drive')
else :
    print('You are not old enough to drive')

# If you want to check for multiple conditions use elif
# If the first matches it won't check other conditions that follow

if age >= 21 :
    print('You are old enough to drive a tractor trailer')
elif age >= 16:
    print('You are old enough to drive a car')
else :
    print('You are not old enough to drive')

# You can combine conditions with logical operators
# Logical Operators : and, or, not

if ((age >= 1) and (age <= 18)):
    print("You get a birthday party")
elif (age == 21) or (age >= 65):
    print("You get a birthday party")
elif not(age == 30):
    print("You don't get a birthday party")
else:
    print("You get a birthday party yeah")
```

For loops

```
# FOR LOOPS -------------
# Allows you to perform an action a set number of times
# Range performs the action 10 times 0 - 9
for x in range(0, 10):
    print(x , ' ', end="")

print('\n')

# You can use for loops to cycle through a list
grocery_list = ['Juice', 'Tomatoes', 'Potatoes', 'Bananas']

for y in grocery_list:
    print(y)

# You can also define a list of numbers to cycle through
for x in [2,4,6,8,10]:
    print(x)
```

```
# You can double up for loops to cycle through lists
num_list =[[1,2,3],[10,20,30],[100,200,300]]

for x in range(0,3):
    for y in range(0,3):
        print(num_list[x][y])
```

While loops

```
# WHILE LOOPS -------------
# While loops are used when you don't know ahead of time how many
# times you'll have to loop
import random
random_num = random.randrange(0,100)

while (random_num != 15):
    print(random_num)
    random_num = random.randrange(0,100)

# An iterator for a while loop is defined before the loop
i = 0
while (i <= 20):
    if(i%2 == 0):
        print(i)
    elif(i == 9):
        # Forces the loop to end all together
        break
    else:
        # Shorthand for i = i + 1
        i += 1
        # Skips to the next iteration of the loop
        continue

    i += 1
```

Functions

```
# FUNCTIONS -------------
# Functions allow you to reuse and write readable code
# Type def (define), function name and parameters it receives
# return is used to return something to the caller of the function
def addNumbers(fNum, sNum):
    sumNum = fNum + sNum
    return sumNum
```

```
print(addNumbers(1, 4))

# Can't get the value of rNum because it was created in a function
# It is said to be out of scope
# print(sumNum)

# If you define a variable outside of the function it works every place
newNum = 0
def subNumbers(fNum, sNum):
    newNum = fNum - sNum
    return newNum

print(subNumbers(1, 4))
```

User input

```
# USER INPUT -------------
print('What is your name?')

# Stores everything typed up until ENTER
name = input()

print('Hello', name)
```

Strings

```
# STRINGS -------------
# A string is a series of characters surrounded by ' or "
long_string = "I'll catch you if you fall - The Floor"

# Retrieve the first 4 characters
print(long_string[0:4])

# Get the last 5 characters
print(long_string[-5:])

# Everything up to the last 5 characters
print(long_string[:-5])

# Concatenate part of a string to another
print(long_string[:4] + " be there")

# String formatting
print("%c is my %s letter and my number %d number is %.5f" % ('X', 'favorite', 1, .14))

# Capitalizes the first letter
```

```
print(long_string.capitalize())

# Returns the index of the start of the string
# case sensitive
print(long_string.find("Floor"))

# Returns true if all characters are letters ' isn't a letter
print(long_string.isalpha())

# Returns true if all characters are numbers
print(long_string.isalnum())

# Returns the string length
print(len(long_string))

# Replace the first word with the second (Add a number to replace more)
print(long_string.replace("Floor", "Ground"))

# Remove white space from front and end
print(long_string.strip())

# Split a string into a list based on the delimiter you provide
quote_list = long_string.split(" ")
print(quote_list)
```

File IO

```
# FILE I/O -------------

# Overwrite or create a file for writing
import os
test_file = open("test.txt", "wb")

# Get the file mode used
print(test_file.mode)

# Get the files name
print(test_file.name)

# Write text to a file with a newline
test_file.write(bytes("Write me to the file\n", 'UTF-8'))

# Close the file
test_file.close()

# Opens a file for reading and writing
```

```
test_file = open("test.txt", "r+")

# Read text from the file
text_in_file = test_file.read()

print(text_in_file)

# Delete the file
os.remove("test.txt")
```

Clases

```
# CLASSES AND OBJECTS -------------
# The concept of OOP allows us to model real world things using code
# Every object has attributes (color, height, weight) which are object variables
# Every object has abilities (walk, talk, eat) which are object functions

class Animal:
    # None signifies the lack of a value
    # You can make a variable private by starting it with __
    __name = None
    __height = None
    __weight = None
    __sound = None

    # The constructor is called to set up or initialize an object
    # self allows an object to refer to itself inside of the class
    def __init__(self, name, height, weight, sound):
        self.__name = name
        self.__height = height
        self.__weight = weight
        self.__sound = sound

    def set_name(self, name):
        self.__name = name

    def set_height(self, height):
        self.__height = height

    def set_weight(self, height):
        self.__height = height

    def set_sound(self, sound):
        self.__sound = sound

    def get_name(self):
```

```
        return self.__name

    def get_height(self):
        return str(self.__height)

    def get_weight(self):
        return str(self.__weight)

    def get_sound(self):
        return self.__sound

    def get_type(self):
        print("Animal")

    def toString(self):
        return "{} is {} cm tall and {} kilograms and says {}".format(self.__name, self.__he

# How to create a Animal object
cat = Animal('Whiskers', 33, 10, 'Meow')

print(cat.toString())

# You can't access this value directly because it is private
#print(cat.__name)
```