# Introduction to Python/Python Refresher

---

Here are some helpful tips for python coding usage in this guide:

***Every python3 code/script should start with:***

`#!/bin/python3`

***TIP: When you want to work on a code file and also have shell access, use this command:***

***`gedit sample.py&`***

## Strings

When printing strings in python we use the follwoing formats:

```
print("Hello, world! ")
print('Hello, world!')
print("""This string runs
multiple lines""") #triple quote is used for multiple lines
print("This string is "+"awesome") #we can also concatenate strings
```

## Math

These are some basic math calculations:

```
print(50 + 50) #addition
print(50 - 50) #subtraction
print(50 * 50) #multiplication
print(50 / 50) #division
print(50 ** 2) #exponent
print(50 + 50 - 50 / 50) #PEMDAS
print(50 / 6) #remainders in decimal format
print(50 // 6) #no remainders
```

## Variables and Methods

```
quote = "All is fair in love and war" #storing a string in a variable
print(quote) #will print the quote
print(quote.upper()) #will print the quote in all UPPERCASE
print(quote.lower()) #will print the quote in all lowercase
print(quote.title()) #will print the quote in title case
print(len(quote)) #will print the length of the quote
```

```
name = "John" #string
age = 30 #integer (int)
gpa = 3.7 #float
print(int(age)) #printing an integer
print(float(gpa)) #printing a float
print("My name is " + name + " and I am " + str(age) + " years old.") #we use str(age) to convert the integer value of the age variable to a string value
age += 1 #increment the value of age by 1
age -= 1 #decrement the value of age by 1
birthday = 1
age += birthday #incrementing the variable age by the value stored in the variable brithday
```

## Functions

```
#Function definition syntax:
def function_name():
    <funtion code> #indentation is required

#Function call syntax:
function_name()

#Function definition
def who_am_i():
    name = "John"
    age = "30"
    print("My name is " + name + " and I am " + str(age) + " years old.")

who_am_i() #Function call

#Adding parameters
def add_one_hundred(num):
    print(num + 100)

add_one_hundred(100) #Function call with parameter in the parentheses

#Multiple parameters
def add(x,y):
    print(x + y)

add(7,7) #Function call with the parameters in the parentheses
```

```
def multiply(x,y):
    return x * y #Computes the math problem, stores it in memory but does not print
the result

print(multiply(7,7)) #Printing the results of the function call with the parameters in
the parentheses

def square_root(x):
    print(x ** .5)

square_root(7) #Function call with the parameters in the parentheses

#Function to print new line
def new_line():
    print('\n')

new_line() #Function call to print new line
```

## Boolean Expressions (True or False)

```
print("Boolean expressions:")

bool1 = 3 * 3 == 9 #this boolean expression checks if the relational operation is true
or false
bool2 = 3 * 3 != 9 #...
print(bool1,bool2) #prints the results of the boolean expression (True/False)
print(type(bool1)) #prints the datatype of the variable
print(type(bool2)) #...
```

## Relational and Boolean Operators

```
greater_than = 7 > 5
less_than = 5 < 7
greater_than_equal_to = 7 >= 7
less_than_equal_to = 7 <= 7

test_and = (7 > 5) and (5 < 7) #True
test_and2 = (7 > 5) and (5 > 7) #False
test_or = (7 > 5) or (5 < 7) #True
test_or2 = (7 > 5) or (5 > 7) #True

test_not = not True #False
```

## Conditional Statements

```
def drink(money): #we don't need a function of this. It's just for fun
    if money >= 2:
        return "You've got yourself a drink"
    else:
        return "NO drink for you!"

print(drink(3)) #will yeild a result of "You've got yourself a drink"
print(drink(1)) #will yeild a result of "NO drink for you!"

def alcohol(age,money):
    if (age >= 21) an (money >= 5):
        return "We're getting a drink"
    elif (age >= 21) and (money < 5): #else-if (elif) statement is used for
alternatives
        return "Come back with more money"
    elif (age < 21) and (money >= 5): #...
        return "Nice try, kid!"
    else:
        return "You're too poor and too young"

print(alcohol(21,5))
print(alcohol(21,4))
print(alcohol(20,4))
```

## Lists

Lists are mutable data structures (They can be changed). The contents of a list are called "items".

```
#Lists - Have brackets []

movies = ["a", "b", "c"]
print(movies[0]) #will return the first item "a"
print(movies[1]) #will return the secon item "b"
print(movies[1:3]) #will return "b" and "c"
print(movies[1:2]) #will return "b"
print(movies[1:]) #will return the all values to the right starting at postion "1"
print(movies[:1]) #will return all values to the left of position 1 except the value
at position "1"
```

```
print(movies[-1]) #will return the last value of the list
print(len(movies)) #will count the number of items in the list
movies.append("d") #will append "d" to the end of the list
movies.pop() #will delete the last item on the list
movies.pop(0) #will delete the first item on the list
```

### Tuples

Tuples are immutable data structures (They cannot be changed).

```
#Tuples - Do not change and use parentheses ()

grades = ("a", "b", "c", "d", "f")
print(grades[1])
```

### Looping

```
#For Loops - Start to finish of an iterate

letters = ["a", "b", "c"]
for x in letters:
    print(x)

#While Loops - Execute as long as true

i = 1
while i < 10:
    print(i)
    i += 1 #incrementing the value of i by 1 until it is 9
```

### Importing Modules

```
#!/bin/bash
import sys #sys - system functions and parameters module
from datetime import datetime #importing a specific part of the datetime module
from datetime import datetime as dt #importing a specific part of the datetime module
as an alias "dt"
print(sys.version) #will print the python version
print(datetime.now()) #will print the current date and time
print(dt.now()) #will print the current date and time using the alias "dt"
```

### Advanced Strings

```
my_name = "Joe"
print(my_name[0]) #will print first letter from word
print(my_name[-1]) #will print the last letter from the word
sentence = "This is a sentence"
print(sentence[:4]) #will print the first word of the sentence
print(sentence.split()) #will split the sentence whenever there is a space
sentence_split = sentence.split() #...
sentence_join = ' '.join(sentence_split) #will join the sentence whenever there is a
space

quote = "He said 'a'"
print(quote) #will print He said 'a'

quote2 = "He said \"a\""
print(quote2) #will print He said "a"
too_much_space = "       Hello   "
print(too_much_space.strip()) #will not print bthe space around "Hello"
print("a" in "Apple") #will yeild false because the letter "a" in apple is uppercase
(Python is case-sensitive)

letter = "A"
word = "Apple"
print(letter.lower() in word.lower()) #will convert the letter and word to lowercase
and check for the letter "a"

movie = "It"
print("My favorite movie is {}.".format(movie)) #uses the {} placeholder to emulate
concatenation
```

### Dictionaries

```
#Dictionaries - key/value pairs {}
drink = {"Water":1, "Juice":5, "Lemonade":3} #drink is the key and the price is the
value
print(drinks)

employees = {"Finance": ["Bob", "Linda", "Tina"], "IT": ["Gene", "Louise", "Teddy"]}
print(employees)
employees['Legal'] = ["Mr. Frond"] #add new key:value pair
print(employees)
employees.update({"Sales": ["Andie", "Ollie"]}) #add new key:value pair
```

```
print(employees)

drinks['Water'] = 2
print(drinks)

print(drinks.get("Water"))
```

### Sockets

Sockets are used to connect two nodes together (make a file that's not called socket.py because the machine will think that it is a socket)

```
#!/bin/python3
import socket #importing the socket module

HOST = '127.0.0.1'
PORT = 7777
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #AF_INET is IPv4 and SOCK_STREAM
is a port
s.connect((HOST,PORT))
```

Next, open a new Terminal window and use the following netcat command `nc -nvlp 7777`. Next, open a new terminal window and run the python program using `python3 s.py`

### Building a Port Scanner

The following is a python program that scans target IP addresses for open ports.

Make a file called `scanner.py`.

```
#!/bin/python3
import sys
import socket
from datetime import datetime

#Define our target
if len(sys.argv) == 2:
    target = socket.gethostbyname(sys.argv[1]) #Translating hostname to IPv4 just in
case the person enters the hostname instead of ipaddress
else:
    print("Invalid amount of arguments.")
    print("Syntax: python3 scanner.py <ip>")

#Add a banner
```

```
print("-" * 50) #Print 50 dashes
print("Scanning target "+target)
print("Time started: "+str(datetime.now()))
print("-" * 50)

try:
    for port in range(50,85):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #AF_INET is IPv4 and
SOCK_STREAM is a port
        socket.setdefaulttimeout(1) #if the connection times out after a second, the
script moves to the next port
        result = s.connect_ex((targert,port)) #returns an error indicator if a port is
closed
        if result == 0:
            print("Port {} is open".format(port))
        s.close() #close the connection

except KeyboardInterrupt: #if we enter an interrupt such as Ctrl+C
    print("\nExiting program.")
    sys.exit()

except socket.gaierror: #For failed DNS hostname resolution
    print("Hostname could not be resolved.")
    sys.exit()

except socket.error: #If we can't connect to the server
    print("Couldn't connect to server.")
    sys.exit()
```

Source: [https://github.com/hmaverickadams](https://github.com/hmaverickadams)