Contents

[Background: 2](#_Toc517857716)

[Packages/Module (as called in python): 2](#_Toc517857717)

[Import **os** 2](#_Toc517857718)

[Import **random** 3](#_Toc517857719)

[Code Basics: 4](#_Toc517857720)

[Control Structures 6](#_Toc517857721)

[If condition 6](#_Toc517857722)

[While Loops 6](#_Toc517857723)

[For Loops: 7](#_Toc517857724)

[Data Structures 8](#_Toc517857725)

[String: 8](#_Toc517857726)

[Lists:: 8](#_Toc517857727)

[Tuples:: 9](#_Toc517857728)

[Dictionaries:: 10](#_Toc517857729)

[Functions: 11](#_Toc517857730)

[Operating with Files: 11](#_Toc517857731)

[Different modes of file operation: 11](#_Toc517857732)

[OOP: 12](#_Toc517857733)

[References: 13](#_Toc517857734)

Python

# Background:

* Python is an **interpreted** (no need of compiling. Each line is executed as it is written) **High level language.**
* Three major versions: 1.x, 2.x, 3.x (2.x and 3.x currently in use)
* An interpreter is a program that runs scripts written in an interpreted language such as Python.
* **CPython** is an implementation of Python
* **Inventor:** Guido Van Rossum
* Python includes **IDLE** (Integrated DeveLopment Environment) which includes tools for writing and debugging entire programs
* <https://www.python.org/shell/>
* Python is case sensitive
* Python source file extension is \*.py

# Packages/Module (as called in python):

**import** modules at the beginning of code just like #include

Modules are pieces of code that others have written to fulfill common tasks. The basic way to use a module is to use **import module\_name** at the top of your code and then using module\_name.var to access functions and values with the name “var” in that module.  
Format --  
**import** math

Another way to import is if you want to import only a certain function of a module.  
Format to do so --   
**from** math **import**  pi,sqrt   
# importing a constant “pi” and function “sqrt” from “math” module.  
print(pi)  
print(sqrt(4))

Aliasing:

**from** math **import**  sqrt as square\_root  
print(square\_root(4)) #after aliasing, you cannot use the original name.

Any import failure raises exception: “ImportError”

### Import **os**

* Executing a shell command  
  **os.system()**
* Get the users environment

**os.environ()**

* Returns the current working directory.

**os.getcwd()**

* Return the real group id of the current process.

**os.getgid()**

* Return the current process’s user id.

**os.getuid()**

* Returns the real process ID of the current process.

**os.getpid()**

* Set the current numeric umask and return the previous umask.

**os.umask(mask)**

* Return information identifying the current operating system.

**os.uname()**

* Change the root directory of the current process to path.

**os.chroot(path)**

* Return a list of the entries in the directory given by path.

**os.listdir(path)**

* Create a directory named path with numeric mode mode.

**os.mkdir(path)**

* Recursive directory creation function.

**os.makedirs(path)**

* Remove (delete) the file path.

**os.remove(path)**

* Remove directories recursively.

**os.removedirs(path)**

* Rename the file or directory src to dst.

**os.rename(src, dst)**

* Remove (delete) the directory path.

**os.rmdir(path)**

### Import **random**

* To generate random numbers  
  random\_num = random.randrange(0,100)

### Import **math**

* square­\_root = math.sqrt(num)

# Code Basics:

* >>> is the prompt for python
* >>>**print**(‘abc’) //output: abc print produces output which **won’t** have quotes.
* >>>”Hello” //output: ‘Hello’ string will return string with single quotes
* >>>print(‘print(“print”)’) //output: print(“print”) . Double quotes inside single quotes don’t need escape character. Same goes vise versa.
* Different ways of using **print** command:
  + print(“hgjhg”+”bbbb”,23+34,4.5)
  + print( “%s %s %s” % (‘first string’, ‘second’, ‘’’ my multiline  
    string ‘’’)) [\\this](file:///\\this) is a multiline print out
  + print(“I don’t like “, end=””)   
    print(“newlines”) //output: I don’t like newlines. end=”” is used to remove newlines at the end of print
  + print(‘\n’ \* 5) [\\this](file:///\\this) prints 5 newlines(or any other string)
* Comment:
  + Single Line: #abcdhjkhkjh
  + Multiline:  
     ‘’’ My  
    multiline  
    comment ‘’’
* >>>2+2\*(3-5) //python can calculate. Spaces can be added for enhanced readability
* >>>10/2 //5.0 **Single slash** division in python produces float result.
* >>>10//2 // 5 **Double slash** produces int
* >>>10//4 //2 (floor division)
* >>>2//2.0 //1.0 Operations with float will always produce float. Integers are implicitly converted to float, against the rule in Python to explicitly convert types.
* >>>-5//2 //-3 not -2 because -5 should be greater than 2\*-3 i.e. -5>-6. That is how floor is taken
* >>>2.3-2 //0.2999999999999998
* >>> 5\*4.56 //22.799999999999997
* Dividing by zero produces “**ZeroDivisionError**”
* >>>2\*\*3 //8 \*\* is the sign for power function(exponentiation)
* >>>1.25%0.5 //0.25 modulo function
* A string is created in python using text between two single or double quotes
* To escape a character such as quotes, use a backslash before it. Use \n to add new line in a print.
* To **clear screen**:
  + >>>import os
  + >>>os.system(‘cls’) //if windows
  + >>>os.system(‘clear’) // if linux
* **MultiLine string**:  
  >>> """jhgjhgjhg

... jhgjgjhgjhg

... hvnbvnbv"""

Output: 'jhgjhgjhg\njhgjgjhgjhg\nhvnbvnbv'

Use three single/double quotes

* To take input, use input function:

>>>**input**(“Enter here: ”)

Enter here: This is what \n user entered

‘This is what \\n user entered  
  
**OR**import sys  
print( ‘Name please: ‘)  
name = sys.stdin.readline()  
print( ‘Hello ‘,name)

* **Concatenation**:
  + >>>”2” + ’2’ \\’22’
  + >>>”2” + 1 + ’3’ \\**TypeError:** unsupported operand type(s) for +:’int’ and ‘str’
* **Explicit Type Conversion:**
  + String to Int**: int** function>>>int(‘2’)+int(‘3’) \\5  
    >>>int(‘3’+’4’) \\34
  + String to float**: float** function  
    >>>float(‘2.3’)+float(‘3.0’) \\5.3
  + To string: **str** function  
    >>> str(2)+str(3.5) \\'23.5'
* >>>float(“210”\*int(input(“Enter a number: “)))  
  Enter a number: 2  
  \\output 210210.0
* **Variables**:
  + >>>x = 3
  + >>>print(x) \\3
  + In python, variables don’t have specific types.
  + Variable names can have alphabets, numbers and underscores. The can’t start with number (Gives **SyntaxError: invalid syntax**)
  + Trying to refer a variable which is not yet created throws error (**NameError: name ‘…’ is not defined**)
  + To delete a variable, use **del** function (>>>del myVar)
* **In-Place Operators:**
  + Python supports +=, \*=, -=, /= and %=  
    eg: >>> x = 4  
    >>> x -=1 \\same as x=x-1  
    output: 3  
    >>> x = ‘a’  
    >>> x \*= 3 \\ ‘aaa’
  + Python doesn’t support incrementor or decrementor i.e. ++ and --

# Control Structures

* **Boolean** data type: Values- **True/False**
* **Comparator ==** , != , >, <, <=, >=
* Comparators can be used to compare string lexicographically.

If condition**:**>>>if expression **:** \\note the colon  
 statement1  
 statement2 \\note the indentation for inner statements

* >>>if expression\_out : \\if block 1  
   statement1  
   if nested\_expression : \\note the indentations at different levels. If block 2  
   statement2  
   else: \\this else belongs to the inner if(block 2)  
   if expression1 : \\block 3  
   statement3  
   else: \\block 3  
   statement4  
   elif expression\_elif : \\block 1  
   statement5  
   else : \\block 1  
   statement6
* Python uses indentation(mandatory) to delimit blocks of code. In C language, it is accomplished using curly braces. If the indentation is not proper, you’ll get “**IndentationError”**
* **Boolean Comparators:** “and”, “or”, “not”
* **Operator Precedence:** (google or add the image)

While Loops**:**   
i = 5  
while i > 0 :  
 if i == 1 :  
 **break**  
 elif i == 3 :  
 **continue**  
 else :   
 print(i)  
 i -= 1

* Leave a newline after the while loop or if block to indicate end of statements.
* Using **break/continue** outside the loop will cause error

### For Loops:

* for num in **range**(0,10) :  
   print(num, ‘ ‘ , end=””)  
  // 0 1 2 3 4 5 6 7 8 9
* for num in **range**(0,10,2) :  
   print(num, ‘ ‘ , end=””)  
  // 0 2 4 6 8
* someList = [1,2,’Mango’,’Hi’]  
  for item in someList:  
   print(item, ‘ ‘ , end=””)  
  // 1 2 Mango Hi
* for item in [1,2,3,4,5]:  
   print(item, ‘ ‘ , end=””)  
  // 1 2 3 4 5
* list\_2D = [[1,2],[3,4],[5,6]]   
  for x in range(0,3):  
   for y in range(0,2):  
   print(list\_2D[x][y], ‘ ‘ , end=””)

# Data Structures

### String:

* **String multiplied by integer**: It produces a repeated version of the original string. String can’t be multiplied by other strings or floats.  
  >>>print(“sp am”\*3) \\ sp amsp am  
  >>>4 \* ’2’ [\\’2222](file:///\\'2222)’
* **Substring**:  
  my\_string = “I’ll catch you if you fall – The Floor”  
  print(my\_string[0:4]) #I’ll  
  print(my\_string[-5:]) #Floor  
  print(my\_string[:-5]) #I’ll catch you if you fall – The  
  print(my\_string[:4] + “ be there”) #I’ll be there
* **Other functions**:  
  my\_string.capitalize() #capitalizes first character in the string  
  my\_string.find(‘Floor’) #33  
  my\_string.isalpha() #False - checks if all are alphabets  
  my\_string.isalnum() #False - checks if string is alphanumeric  
  len(my\_string) #38  
  my\_string.replace(“Floor”,”Ground”) # I’ll catch you if you fall – The Ground  
  my\_string.strip() #removes extra whitespaces  
  word\_list = my\_string.split(“ “) #It prints a list of words in your string which are separated by “ “

## Lists::

* **List:** Lists are another type of objects in Python. List is indexed place holders.
* **Defining a list:** heterogeneousList = [“Hi” , 123 ,”yo!”] \\List can store items of different types together  
  print(heterogeneousList[0]) \\Hi Indexing starts from zero.
* Empty List:  
  emptyList = []
* The last item in a list can be followed by a comma. It’s totally valid.
* **List of Lists**:  
  number=3  
  things = [“string”,0,[1,2,number],”X”]  
  print(things[1]) \\0  
  print(things[2]) \\[1,2,3]  
  print(things[2][2]) \\3
* A string can behave like a list and its characters can be retrieved using index:  
  str = “Hello world!”  
  print(str[6]) \\w
* Indexing out of bound will cause **IndexError (list index out of range)** and using indexing with anything other than list or string will cause **TypeError ('int' object is not subscriptable).**
* **Reassigning** an item in list:  
  things[1]=26  
  print(things) \\[“string”,26,[1,2,number],”X”]
* Reassigning a character in string throws error:  
  >>> string[2]='y'

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

**TypeError**: 'str' object **does not support item assignment**

* Lists can be added and multiplied just like strings:  
  nums=[1,2,3]  
  strings=[‘a’,’b’,’c’]  
  combined=[nums,strings]  
  print(nums+[4,5,6,7]) \\[1,2,3,4,5,6,7]  
  print(combined) \\[[1,2,3],[‘a’,’b’,’c’]]  
  print(combined[1][0]) \\a  
  print(nums\*3) \\[1,2,3,1,2,3,1,2,3]
* Check if an item is in the List: **“in”** operator:  
  nums=[1,2,3]  
  print(2 in nums) \\true
* Check if an item is not in the List: **“not in”** operator:  
  nums=[1,2,3]  
  print(4 not in nums) \\true  
  print(not 4 in nums) \\true both ways are allowed
* **Append** an item to a list:  
  nums.append(6) [\\nums](file:///\\nums) is now [1,2,3,6]  
  Equivalent to nums[len(nums):] = [6].
* **Insert** an item at a particular index:  
  nums.insert(2,5) [\\[1,2,5,3,6](file:///\\[1,2,5,3,6)]
* **Remove** an item:  
  nums.remove(2) [\\[1,5,3,6](file:///\\[1,5,3,6)]
* **Reverse:** nums.reverse \\[6,3,5,1]
* **Sort:** nums.sort() [\\[1,3,5,6](file:///\\[1,3,5,6)]
* **Delete:** del nums[0] [\\[3,5,6](file:///\\[3,5,6)]
* **Get len,max or min:** print(len(nums))

## Tuples::

* Tuples are almost same as Lists except that Tuples are **immutable**.
* my\_tuple = (1,3,5,2,4)
* tuple\_to\_list = list(my\_tuple)
* list\_to\_tuple = tuple(tuple\_to\_list)
* min,max,len functions are applicable here.

## Dictionaries::

* They are like lists but they cannot be joined using “+”
* They contain key value pairs.
* agents = {‘438847’ : ‘Anulika Banerjee’ , ’438846’ : ‘James Bond’}
* print(agents[‘438847’]) [\\Anulika](file:///\\Anulika) Banerjee  
  or print(agent.get(‘438847’))
* del,replace…such activities are performed using only keys.
* To get the list of keys: print(agents.keys()) \\dict\_keys([‘438847’,’438846’])
* To get the list of values: print(agents.values()) [\\dict\_values([‘Anulika](file:///\\dict_values(['Anulika) Banerjee’,’James Bond’])

# Functions:

* **def** function\_Name(arg1,arg2) :   
   #logic goes here  
   sum = arg1 + arg2  
   return sum #return is not compulsory

print(function\_Name(2,5)) # 7

* Functions are just like any other variable. It can be assigned to another variable.

def add(x+y):  
 return x+y

a=2  
b=4  
add\_nums=add  
print(add\_nums(a,b)) #6

* Functions can also be used as arguments to other function

def add(x,y):  
 return x+y  
def do\_twice(**func**,x,y):  
 return **func**(**func**(x,y),**func**(x,y))  
a=5  
b=10  
print(do\_twice(**add**,a,b)) #30

# Operating with Files:

import os

my\_file = open(“test.txt”, “wb”)

my\_file.write(bytes(“write the file content here”, ‘UTF-8’))

my\_file.close()

my\_file = open(“test.txt”, “r+”)

print(my\_file.mode()) #r+

print(my\_file.name()) #test.txt

text\_in\_file = my\_file.read()

print(text\_in\_file) # write the file content here

os.remove(“test.txt”)

### Different modes of file operation:

* wb
* ab+
* r+
* etc…..

# OOP:

* to define attributes in a class as private, it should start with double underscores  
  eg: \_\_name

### Encapsulation

# References:

* <https://docs.python.org/3.3/tutorial/datastructures.html>
* <https://www.youtube.com/watch?v=N4mEzFDjqtA>
* <http://www.pythonforbeginners.com/os/pythons-os-module>
* <https://www.sololearn.com/Course/Python/>