2021
EDITION

AUTHOR
SYED TAIMOOR NAWAZ

Ultimate Python

May 2021

“You primarily write your code to communicate with other coders, and, to a lesser expert, to impose your will on the computer.”

S.TAIMOOR

In [ ]:

```
# NOTE : ALL CODE RUN YOUR EDITOR USING 'PRINT FUNTION' AND PRACTICE IT
```

In [ ]:

```
# COMMENTS
# This is single line comments
'''This is
multi - line comments
'''
# Python Ignores Comments
```

In [5]:

```
# how can data store
# data is a variable: data/values can be stored in temporary storage is called variable
data = "ali"
data = "Taimoor"
data = "Mohmmad Ali"
data # so variable can be changed consistently --> mustakil toor per
```

...

In [10]:

```
# Data Types: Every data/values is associated with data-type
# Types of data-types : integer, float, boolean, string, complex
a1 = 10 # int
type(a1)
a2 = 10.23 # Float
type(a2)
a3 = "Taimoor"
type(a3)
a4 = True # similarly False
type(a4)
a5 = 10j # Complex
type(a5)
```

...

In [15]:

```
# Operators in python
# Airthmatic Operators, Relational Operators, Logical Operators
'''
Airthmatic Operators: +,-,*,**,/,%,//
(// this is floor divider and ** this is power symbol in python)'''

x = 12
y = 15

x+y # ---> + This is addition Airthimatic operator
x-y # ----> - This is subtraction Airthimatic operators
x*y # ----> * This is multiplication Airthimatic operator
x//y # ---> This is Floor Divider Airthimatic Operator (This is only integer result)
x/y # --> This is divider Airthimatic Operator
x%y # ---> This is remainder / Modulo Airthimatic Operator

# Relational Operators: ==, !=, >, <, <=, >=
x == y
''' ---> Run This Code Your Editor and it shown result is 'false'
because x is not equals to y so (== is eqyal Relational Operator)'''

x != y # != This is not-equal Relational operator
x > y  #> This is greater then Relational operator
x < y #< This is Less then Relational operator
x>=y #>= This is Less then equal-to Relational operator
x<=y #<= This is greater then equal-to Relational operator

# logical Operators: and / &, or / |
''' Note:please check all logical truth table your enternet browser
(and operator truth table similarly you can search or and not)'''
a = False
b = True
# This is and/ logic operator
a and b
b & a
a and a
b and b
# This is or/ | logic operator
a or b
b or b
a or a
b | a

# Not Restriction You can use this 'and' ya phir ya & similarly or / |
```

Out[15]:

```
True
```

In [16]:

```
'''
#Strings: strings are sequence of charecters enclosed within single qoutes(''),
double qoutes(""), or triple qoutes("""""")/ ('''''')
'''
#E.g. 'Hello World', "This is taimoor", '''Hi My name is
#                                          taimoor nawaz'''
str1 = 'This is my first string'
str1

str2 = "This is my second string"
str2

str3 = '''
This String is a lot
of lines
so this is a
triple qoute string
'''
str3

# Extractng Indivitual Charecter
My_string = 'My Name is Syed taimoor nawaz'
My_string[0] # Its Shows 1st charecter of your given string
My_string[-1]# Its Shows Last Charecter of your given string

#Extracting Sequence Of Charecter
My_string[8:19]

# String Functions
# Finding Length Of the String
len(My_string)

# Converting String to lower case
My_string.lower()

# Converting String to upper case
My_string.upper()

# Replacing a substring
My_string.replace('is','are')
My_string.replace('My','Hi')

# Number of occurences of substring
My_string.count('a')

# Finding a index of substring
My_string.find("taimoor")

# Spiliting a string
fruit = 'I like apple, mangoes, bananas'
fruit.split(',')

My_string.split('a')
```

...

In [ ]:

```
# Python Token:
# Python Token is a smallest meaningfull in a program
# Python Tokens: Keywords, Identifiers, Literals, Operators
# Operators:
# Its already learn in above program you can see 'In [15]'

# Keyword: Keywords are special reserved words
# e.g. False, is return,in, for etc etc (Please serach your browser python keywords)
True = 12 ''' ----> Its Shows Error because its a
resvered word it can not use variables its a special '''

# Litrals: Litrals are constant in python
# e.g.
a = 12
b = 'Syed Taimoor'
# so '12' and 'syed taimoor' is a litral it can not change because it is a constant

# Identifiers: Identifiers are names used for variables, functions or objects
# Rules:
# No Special Charecter expect _(uderscore)
# E.g.
&alpha = 22 # ----> its throw error
# Identifier are case-sensitive
# e.g.

Ali = "Good Man"
ali = "This is another good man"
# Ali and ali both are diffrent variables names its a case - sensitive

# First Letter cannot be a digit
# e.g.
22Ali = "Good" #---> Its shows error massage
```

In [105]:

```
# Basic Data Structure in python
# Tuples, list, Dictionary, set

# TUPLE
# Tuples: Tuples is an ordered collection of elements enclosed within()
'''Tuples are imputable : Matlab ak baar ap nay ak cheez
bna li hai phir iss mai app change nahi ker skhty '''

tup1 = (1, " " ,1.2,1+5j,True,'Taimoor')
tup1
type(tup1)

# How to Extracting individual items in python # Strings and tuples basicaly work in same
tup1[1]
tup1[-1]
tup1[2]
tup1[1:3]  '''Sequence of elememnts Extract # 3 iss liya q kai
yai excliusive hai yai ak -1 ker kai deta hai '''
tup1[1:-1]

# You can not modify a tuple beacuse it is immputable
# tup1[2] = "Hello"
# tup1 #! ERROR

# Basic Tuple Operators
# Finding Length Of tuple
len(tup1)

# Concatinating Tuples

tup2 = (1,2,3,8,5,6)
tup1 + tup2
tup1,tup2

# Repeating Tuple Elements
tup2*3

# Repeating and concatinate
tup2*2 + tup1

# Tuple Functions
min(tup2) # Minimum Value

max(tup2) # Maximum Value

# LIST
# List is an ordered collection of elements enclosed within []
'''List : List are mutable ( Matlab ak baar ap nay ak cheez(list)
bna li hai phir iss mai app changing ker skhty)'''

l1 = [1, " " ,1.2,1+5j,True,'Taimoor']
l1
type(l1)

# Extracting Indivitual Elemnets
l1[3]
l1[2:5]
l1[-1]
```

```
# Modifying a list
# Changing a element at 0th index
l1[0] = 100
l1

#Appending a new element
l1.append("Fahma")
l1

# popping the last element

l1.pop()
l1

# Reversing Element of a list
l1.reverse()
l1

# Inserting a element as a specific index
l1.insert(2,"Syed Taimoor")
l1

# Sorting a list

l2 = ['Banana', 'Apple', 'Graphs', 'Water maloon', 'Guava']
l2.sort()
l2

# List Basic Operation
# Concatenating List

l1 + l2

# Repeating Elements
l2*3
l2*3 + l1

# Dictionary
# Dictionary is an unordered collection of {key:value} pairs enclosed with {}
# Dictionary is mutable

fruit = {'Apple':10, 'Banana':20, 'Mango':200, 'Guava':100}
fruit
type(fruit)

# Dictionry Functions
# Extracting Keys And Values
# Extracting Keys
fruit.keys()

# Extracting Values
fruit.values()

# Modifying a dictionary

#Adding a new element
fruit["Mango"] = 250
fruit

#Changing a existing element
```

```
fruit["Apple"] = 100
fruit

# Update one dictionary elements with another
fruit1 = {'Apple':10, 'Banana':20, 'Mango':200}
fruit2 = {'Guava':100, 'Grapes':233, 'Water mallon':500}

fruit1.update(fruit2)
fruit1

# Poping an element
fruit = {'Apple':10, 'Banana':20, 'Mango':200, 'Guava':100}
fruit.pop('Mango')
fruit

# SET
# Set is an unordered or unindexed collection of elemnets enclosed with {}
# Duplicates are not allowed in set and its are mutable

s1 = {1,"a",True, "Taimoor"}
s1
type(s1)

# Update one set elements with another
s1.add("Hello")
s1

# Updating Multiple Elements

s1.update([23,11,21])
s1

# Removing an element
s1.remove('Hello')
s1

# Set Functions
# Union Of Two Sets
s2 = {2,4,3}
s3 = {"d","a","c"}
s2.union(s3)

# Intersection of two sets
s4 = {2,6,4}
s2.intersection(s4)
```

...

In [36]:

```
# Flow Control Statements
# If Statement

a = 10
b = 20

if b>a:
    print("B greater then A")

print("\n\n")

if a>b:
    print("A greater then B")
else:
    print("B Greater Then A")

print("\n\n")

a = 10
b = 20
c = 30

if (a>b) & (a>c):
    print("A is the greatest")
elif (b>a) & (b>c):
    print("B is the greatest")
else:
    print("C is the greatest")

# IF with tuple, list or dictionary
# if with tuple:

tup1 = ('a','b','c',1,2,3)
if 'z' in tup1:
    print('Value a is present in tup1')
else:
    print('Value z is not presents in tup1')

print("\n\n")
# if with list:
li1 = ['Hello','Syed Taimoor','2',2,5,7]
if 'Syed Taimoor' in li1:
    print("Value of Syed Taimoor is presents")
print("\n\n")

if li1[1] == '2':
    print('value 2 is present in 2 index number')
else:
    print('Value of 2 is not present in index 1')

if li1[2] == '2':
    li1[2] = 'a','b'
li1
print("\n\n")
# if with dictionary
d1 = {'k1':10, 'k2':20, 'k3':30}
if d1['k3'] == 30:
    d1['k3'] = d1['k3']+100
    d1['k2'] = 40
    d1['k1'] = d1['k1']*20
d1
print("\n\n")
# Looping Statements
# Looping statement are used to reapeat a task multiple times

# While Loop
i = 1
while(i<=10):
    print(i)
    i = i+1
print("\n\n")
# Table
i=1
n=2
while(i<=10):
    print(n ,"*" ,i, "=", n*i)
    i=i+1

print("\n\n")
# while with list
li = [1,2,3,4,5]
i = 0
while i<len(li):
    li[i]=li[i]+100
    i = i+1
li
print("\n\n")
# For loop is used to iterate over a sequance(tuple, list, dictionary, strings...)
# for variable in sequence:
#               body of for
l1 = ['Mango','Apple','Grapes','Guava']
for i in l1:
    print(i)
print("\n\n")
l2 = ['Orange','Red','Green','Blue','Purple']
for i in l1:
    for j in l2:
        print(i,j)
```

...

In [87]:

```
# Functions
# You Can Achive multiple task in one time
# Function is a block of code which performs a specific task
def Hello(): #Function create withowt parameter
    print("Hello World")
# Invoke
Hello()

def add_20(x):
    return x+20
add_20(15)
add_20(20)

def odd_even(x):
    if x%2==0:
        print(x," is Even")
    else:
        print(x," is Odd")

odd_even(13)

# Special Type of function in python
''' Lamda / anonymous function(anonymous function iss liya
becoause iss ka koi name nahi rhata)'''
# lambda parameter: Expresion ---> syntax
g = lambda x: x*x*x
g(7)
g(5)

# Normaly lambda funcion kuxh or functions kai saat mai use hota hai
# or un functions ko filter, map or raduce bolty hai

# Lambda with filter
'''
What is filter: Maaan li jiyan ap kai pass koi sequence hai jasy tuple, list ya dict so
koi bhi sequence hoo or uss sequence main sy koi value filter kerni hoo on
the basis of condition so  we can use filter in conjuction in the lembda function
'''
# filter(lambda,list) --> syntax
l1 = [23,44,76,89,12,11,0,9,67,32,22]

final_list=list(filter(lambda x: (x%2!=0),l1))
final_list

# Lambda with map
l2 = [1,2,3,4,5,6,7,8,9] # All values *by 2
# map(lambda,list) ---> syntax
final_list=list(map(lambda x: x*2,l2))
final_list

'''Lambda with reduce (hamy ak final result chiyan kisi ak sequence
kai upper so u can use reduce)'''
from functools import reduce

# reduce(lambda, list) --> syntax

sum = reduce(lambda x,y : x+y,l2)
sum
```

. . .

In [125]:

```python
# OOP (OBJECT ORIENTED PROGRAMMING)
#There are two important concepts in oop
# 1) Object 2) Classes
# Classes: Class is a template/blue-print for real-world(OBJECT) entities
# E.g. Phone ---> 1) Properties/attributes 2) Bahiours/methods
# Class is user define data type
# Object are specific instances of a class

class Phone: #Creating The Phone Class
    def making_call(self):
        print("Making a Call")
    def play_game(self):
        print("Playing game")

p1 = Phone() # instantiating the 'p1' object

p1.making_call()  #Invoke Through Objects
p1.play_game()

print("")
# Adding Parameter to a class

class Phone2:
    def set_color(self, color):
        self.color = color
    def set_cost(self, cost):
        self.cost = cost
    def show_color(self):
        return self.color
    def show_cost(self):
        return self.cost
    def making_call(self):
        print("Making a call")
    def playing_game(self):
        print("Playing Game")

p2 = Phone2()

p2.set_color("Blue")
p2.set_cost(2000)
p2.show_color()
p2.show_cost()
p2.making_call()
p2.playing_game()

# Creating a class with constructor:
'''Constructor: Constructor is a special type of method (jasy hi object create
kerty hoo usi wqt sabhi koo assign ker skhty ho)'''
# constructor is defined __init__ method
class Employee:
    def __init__(self,name, age, salary, gender):

        self.name = name
        self.age = age
        self.salary = salary
        self.gender = gender

    def employee_details(self):
        print("Name of Employee", self.name)
        print("Age of Employee", self.age)
        print("Salary of Employee", self.salary)
        print("Gender of Employee", self.gender)

e1 = Employee("Taimoor", 18, 200000000, "Male") # Instantiating the Employee Object
e1.employee_details() # Invoke the employee_details method

# Inheritance : with inheritance one class can drive the properties of another class
# E.g. Man Inheriting features from his father
# Single Inheritance
class Vehical:  # Creating The base class/supper class
    def __init__(self, mileage, cost):
        self.mileage = mileage
        self.cost = cost

    def show_Vehical_details(self):
        print("I'm a Vehical")
        print("Vehical Mileage", self.mileage)
        print("Vehical Cost", self.cost)

class Car(Vehical): # Creating the child class
    def Show_car(self):
        print("I'm a car")

print("\n")
v1 = Vehical(23090,30000000) # Instansiating the object of base class/supper class
c1 = Car(2309,230000000)  # Instansiating the object of Child class

v1.show_Vehical_details()
print("\n")
c1.show_Vehical_details()
print("\n")
c1.Show_car() # Invoking the child class method

# Over-riding init method

class car(Vehical):
    def __init__(self, mileage, cost, tyres, hp):
        super().__init__(mileage, cost)
'''over-ride init method
so hamy parent class mai koi cheez add..
kerni hoo or use keri hoo child mai too haam super()
method ka use karyingy child kai under'''

        self.tyres = tyres
        self.hp = hp

    def show_car_details(self):
        print("Number of tyres are", self.tyres)
        print("Horse Power of the car is ", self.hp)
print("\n")
c2 = car(1200,2389999,4,1.9)
c2.show_Vehical_details()
print("\n")
c2.show_car_details()

# Types of inheritance
# Single Inheritance, multi inheritanve, multi-level inheritance, Hybrid inheritance

'''Multiple inheritance (In Multiple Inheritance, the
child inherits from more then 1 parent class)'''
```

```
# Parent1 ------------> child <------------ Parent2

# Parent Class One
class Parent1:
    def assign_string_one(self,str1):
        self.str1 = str1

    def show_string1(self):
        return self.str1

class Parent2:
    def assign_string_two(self, str2):
        self.str2 = str2

    def show_string2(self):
        return self.str2

class Derived(Parent1, Parent2):
    def assign_string_three(self, str3):
        self.str3 = str3

    def show_string3(self):
        return self.str3

der = Derived()
der.assign_string_one("I'm string of parent one")
der.assign_string_two("I'm string of parent two")
der.assign_string_three("i'm string of child")

der.show_string1()
der.show_string2()
der.show_string3()

# Multilevel Inheritance
# In Multi-level inheritance, we have Parent, child, grand-child relationship
# Parent ----> Child(inherited parent) ----> Grandchild(inherited child)

class Parent:
    def get_name(self,name):
        self.name = name
    def show_name(self):
        return self.name

class Child(Parent):
    def get_age(self,age):
        self.age = age
    def show_age(self):
        return self.age

class Grand_child(Child):
    def get_gender(self,gender):
        self.gender = gender
    def show_gender(self):
        return self.gender

final = Grand_child()

final.get_name("Taimoor")
final.get_age(18)
final.get_gender("male")
final.show_name()
final.show_age()
final.show_gender()
```

...

In [ ]:

```
'''
COMPILED BY
SYED TAIMOOR NAWAZ
'''
#   --------------------------------  THANK YOU -----------------------------------------
```