Python Basics

1. Write a program to print your name.

print ("SHASHWAT KHAITAN")

2. Write a program for a Single line comment and multi-line comments.

#This is a multiline comment in python

#And expands to more than one line

print("Pythonforbeginners.com")

3. Define variables for different Data Types int, Boolean, char, float, double and print on the Console.

a = -5

print ("Type of a: ", type(a))

b = False

print ("Type of b: ", type(b))

c = 5.0

print ("Type of c: ", type(c))

String = 'Hello'

print ("Type of String: ", type (String))

4. Define the local and Global variables with the same name and print both variables and understand the scope of the variables.

a=8

def f ():

print ('Inside f (): ', a)

def g ():

a=5

print ('Inside g (): ', a)

def h ():

global a

a=6

print ('Inside h (): ', a)

print ('global: ', a)

f ()

print ('global: ', a)

g ()

print ('global: ', a)

h ()

print ('global: ', a)

Operators

1. Write a function for arithmetic operators (+, -,\*,/)

def add (n1, n2):

return n1+n2

def minus (n1, n2):

return n1-n2

def multiply (n1, n2):

return n1\*n2

def divide (n1, n2):

return n1/n2

num1= int (input ("Enter First number:"))

num2 = int (input ("Enter Second number:"))

print (num1,"+”, num2,"=", add (num1, num2))

print (num1,"-”, num2,"=", minus (num1, num2))

print (num1,"\*”, num2,"=",multiply(num1, num2))

print (num1,"/", num2,"= “, divide (num1, num2))

2. Write a method for increment and decrement operators (++, --)

a=0

a+=1

a=a+1

print ('The value of a is ‘, a)

print ("INCREMENTED FOR LOOP")

for i in range (0,5):

print(i)

print ("DECREMENTED FOR LOOP")

for i in range (4, -1, -1):

print(i)

3. Write a program to find the two numbers equal or not.

a=int (input ("Enter first number:"))

b=int (input ("Enter second number:"))

if a==b:

print ("Both numbers are equal")

else:

print ("Both numbers are not equal")

4. Program for relational operators (<, <==, >, >==)

a=10

b=15

print(a<b)

print(a<=b)

print(a>b)

print(a>=b)

print(a==b)

print (a! =b)

5. Print the smaller and larger number.

a=int (input('Enter first number: '))

b=int (input ('Enter second number: '))

if a>b:

print ("a is greater")

else:

print ("b is greater")

if a<b:

print ("a is smaller")

else:

print ("b is smaller")

Loops

1. Write a program to print “Bright IT Career” ten times using for loop.

a="Bright IT Career"

for i in range (10):

print(a)

2. Write a java program to print 1 to 20 numbers using the while loop.

i=1

while(i<=20):

print(i)

i+=1

3.Program to equal operator and not equal operators.

a=15

b=20

print(a==b)

print (a! =b)

4. Write a program to print the odd and even numbers.

Numbers= [1,2,3,4,5,6,7,8,9,10]

print ("Even Numbers: ")

for i in Numbers:

if i%2==0:

print (i, end =" ")

print ("Odd Numbers: ")

for i in Numbers:

if i%2! =0:

print (i, end =" ")

print ()

5. Write a program to print largest number among three numbers.

k=40

a=50

s=90

if k>=a and k>=s:

largest=k

elif a>=k and a>=s:

largest=a

else:

largest=s

print ("Largest number is: “, largest)

6. Write a program to print even number between 10 and 20 using while.

a=10

b=20

print ("Even Numbers Between 10 to 20: “, end=" ")

while a<=b:

if(a%2==0):

print ("{0}”. format(a), end=" ")

a=a+1

print ()

7. Write a program to print 1 to 10 using the do-while loop statement.

a=1

print ("Print 1 to 10 using the do-while loop statement:”, end=" ")

while True:

print (a, end=" ")

a=a+1

if(a>10):

break

print ()

8. Write a program to find Armstrong number or not.

a=int (input ("Enter a number: "))

sum=0

temp=0

temp=a

while temp>0:

r=temp%10

sum+=r\*\*3

temp//=10

if a==sum:

print ("It is an Armstrong number")

else:

print ("It is not an Armstrong number")

9. Write a program to find the prime or not.

number=int (input ("Enter a number: "))

if number>1:

for i in range (2, number):

if (number% I) ==0:

print ("It is not a prime number")

break

else:

print ("It is a prime number")

else:

print ("It is not a prime number")

10. Write a program to palindrome or not.

n=int (input ("Enter a number:"))

temp=n

rev=0

while(n>0):

dig=n%10

rev=rev\*10+dig

n=n//10

if(temp==rev):

print ("The number is a palindrome")

else:

print ("The number is not a palindrome")

11. Program to check whether a number is EVEN or ODD using switch.

a=int (input ("Enter a Number: "))

if a%2==0:

print ("{0} is Even “. format(a))

else:

print ("{0} is Odd “. format(a))

Arrays

1. Write a function to add integer values of an array.

arr= [10,20,30,40]

sum=0

for i in range (0, len(arr)):

sum=sum+arr[i]

print ("Sum of all integer values in array: “, sum)

1. Write a function to calculate the average value of an array of integers.

def cal\_average(num):

sum\_num=0

for i in num:

sum\_num=sum\_num+i

avg=sum\_num/len(num)

return avg

print ("The average is", cal\_average([10,21,32,43,54]))

1. Write a program to find the index of an array element.

arr= [1,3,5,3,1,2,6,5,3,8,6,9]

index=arr.index(3)

print ("Index of 3:”, index)

index=arr.index(9)

print ("Index of 9:”, index)

index=arr.index(8)

print ("Index of 8:”, index)

1. Write a function to test if array contains a specific value.

arr= [4,5,45,40,50]

for num in arr:

if num==5:

print ("Element exist")

1. Write a function to remove a specific element from an array.

arr = [44,55,0,15,19,5,4]

arr.remove(0)

print(arr)

1. Write a function to copy an array to another array.

arr=['k','a','s','h','i']

arr1= []

arr1=arr.copy()

print(arr1)

1. Write a function to insert an element at a specific position in the array.

arr= [101,303,404,505,606,707,808,909]

arr.insert(1,202)

print(arr)

1. Write a function to find the minimum and maximum value of an array.

arr= [100,3,3000,20,500,9999,10000,10003]

minposition=arr.index(min(arr))

print ("The minimum is at position:", minposition)

maxposition=arr.index(max(arr))

print ("The maximum is at position:", maxposition)

1. Write a function to reverse an array of integer values.

arr= [9,8,7,6,5]

arr. reverse ()

print(arr)

1. Write a function to find the duplicate values of an array.

arr= [21,11,31,13,11]

for i in range (0, len(arr)):

for k in range (i+1, len(arr)):

if arr[i]==arr[k]:

print ("Duplicate element in given array:”, arr[k])

1. Write a program to find the common values between two arrays.

arr=['k','a','s','h','i']

arr1=['s','h','g']

print ("Common values in given arrays:",set(arr).intersection(arr1))

1. Write a method to remove duplicate elements from an array.

arr = [11,22,33,11,44,55]

finalarr = []

for i in arr:

if i not in finalarr:

finalarr.append(i)

print(finalarr)

1. Write a method to find the second largest number in an array.

arr= [101,404,202,909,606,505,808,303,707]

arr.sort()

print ("Second largest number:”, arr[-2])

14. Write a method to find number of even number and odd numbers in an array.

arr=[1,2,3,4,5,6,7,8,9]

evennumbers=0

oddnumbers=0

for i in arr:

if i%2==0:

evennumbers+=1

else:

oddnumbers+=1

print ("Even Numbers in given array:”, evennumbers)

print ("Odd Numbers in given array:",oddnumbers)

15. Write a function to get the difference of largest and smallest value.

arr= [10,6,11,13,14]

arr.sort()

print ("Diffrence of largest and smallest value:",arr[4]-arr[0])

16. Write a method to verify if the array contains two specified elements (12,23)

arr= [1,12,19,23,33,54]

for i in arr:

if i==12:

print ("Exist in array")

if i==23:

print ("Exist in array")

Static

1. Define a static variable and access that through a class.

class Python:

staticVariable = 9

print (Python.staticVariable)

1. Define a static variable and access that through an instance.

Python.staticVariable=12

print (Python.staticVariable)

1. Define a static variable and change within the instance.

instance = Python ()

print(instance.staticVariable)

4. Define a static variable and change within the class.

instance.staticVariable=15

print(instance.staticVariable)

print (Python.staticVariable)

Strings

1. Different ways creating a string.

string='Hello'

print(string)

string="Hello"

print(string)

string1='''World'''

print(string1)

string2="""Welcome to the world of Python"""

print(string2)

print ()

1. Concatenating two strings using + operator.

str1=string+string1

print ("Concatenated two different strings:",str1)

print ()

1. Finding the length of the string.

print ("length of the string:”, len(str1))

print ()

1. Searching in strings using index ().

str3='kashish'

str1='ish'

str2='h'

print ("Position of ish:”, str3.index(str1))

print ("Position of h:”, str3.index(str2))

print ()

1. Matching a String Against a Regular Expression With matches ().

from ast import Str

import re

Substr='Madara'

str6='Madara once said- Wake up to reality nothing goes as planned in this cursed world'

print(re.match(Substr,str6))

print ()

1. Comparing strings.

str8='Itachi uchiha'

str1='Madara uchiha'

str2=str8

print(str8==str1)

print(str8==str2)

print(str1==str2)

print (str8! =str1)

print ()

1. startsWith(), endsWith() and compareTo().

string='Minato Namikaze'

print(string.startswith("Minato"))

print(string.endswith("kaze"))

print ()

1. Trimming strings with strip ().

str7='Hello World hi'

print (str7.strip("hi"))

print ()

1. Replacing characters in strings with replace ().

string='Hi World'

print(string.replace("Hi","Hello"))

print ()

1. Splitting strings with split ().

str9='hi-hello-bye'

print (str9.split("-"))

print ()

1. Converting integer objects to Strings.

numb=10

numb1=str(numb)

print(numb1)

print(type(numb1))

1. Converting to uppercase and lowercase.

string='hello'

string1='WORLD'

print (string. Upper ())

print (string1.lower())

Inheritance

A, B and C are classes

A is a super class. B is a sub class of A. C is a sub class of B.

Create three methods in each class, 2 methods are specific to each class and third method (override method) should be in all three Classes A, B and C

Create a class with main method.

Create an object for each class A, B and C in main method and call every method of each class using its own object/instance.

Call an overridden method with super class reference to B and C class’s objects

Runtime Polymorphism with Data Members/Instance variables, Repeat the above process only for data members.

class A():

def display(dp):

print ("Display Inside class A ")

def show(self):

print ("Show Inside class A")

class B (A):

def print(pt):

print ("Print Inside class B")

def show(self):

print ("Show Inside class B")

class C (B):

def show(self):

print ("Show Inside class C ")

s = A ()

s.display()

k= B ()

k.print()

g = C ()

g.show()

Access Modifiers

1. Create a class with PRIVATE fields, private method, and a main method. Print the fields in main method. Call the private method in main method. Create a sub class and try to access the private fields and methods from sub class.

2. Create a class with PROTECTED fields and methods. Access these fields and methods from any other class in the same package. Also, Access the PROTECTED fields and methods from child class located in a different package Access the PROTECTED fields and methods from any class in different package.

3. Create a class with PUBLIC fields and methods. Access the public methods and fields from any class in the same package or different package.

class Super:

var1 = None

var2 = None

var3 = None

def \_\_init\_\_(self, var1, var2, var3):

self.var1 = var1

self.\_var2 = var2

self.\_\_var3 = var3

def displayPublicMembers(self):

print("Public Data Member: ", self.var1)

def \_displayProtectedMembers(self):

print("Protected Data Member: ", self.\_var2)

def \_\_displayPrivateMembers(self):

print("Private Data Member: ", self.\_\_var3)

def accessPrivateMembers(self):

self.\_\_displayPrivateMembers()

class Sub(Super):

def \_\_init\_\_(self, var1, var2, var3):

Super.\_\_init\_\_(self, var1, var2, var3)

def accessProtectedMembers(self):

self.\_displayProtectedMembers()

obj = Sub("KG", 5 , "KG !")

obj.displayPublicMembers()

obj.accessProtectedMembers()

obj.accessPrivateMembers()

print("Object is accessing protected member:", obj.\_var2)

Abstract Class

1. Create an abstract class with abstract and non-abstract methods.

2. Create a sub class for an abstract class. Create an object in the child class for the abstract class and access the non-abstract methods

3. Create an instance for the child class in child class and call abstract methods

4. Create an instance for the child class in child class and call non-abstract methods.

from abc import ABC, abstractmethod

class Polygon(ABC):

@abstractmethod

def noofsides(self):

pass

class Triangle(Polygon):

def noofsides(self):

print("Triangle: I have 3 sides")

class Pentagon(Polygon):

def noofsides(self):

print("Pentagon: I have 5 sides")

class Hexagon(Polygon):

def noofsides(self):

print("Hexagon: I have 6 sides")

class Quadrilateral(Polygon):

def noofsides(self):

print("Quadrilateral: I have 4 sides")

R = Triangle()

R.noofsides()

K = Quadrilateral()

K.noofsides()

R = Pentagon()

R.noofsides()

K = Hexagon()

K.noofsides()

Files

1. Write a program to read text file.

file1 = open("HW.txt","r")

data = file1.read()

print(data)

print()

file2 = open("Blank.txt","w")

str1 = 'Python'

file2.write(str1)

print()

file3 = open("HW.txt","r+")

print(file3.readline(11))

print()

1. Write a program to write text to .txt file using InputStream.

f=open('C:\myfile.txt')

while True:

try:

line=next(f)

print(line)

except StopIteration:

break

f.close()

1. Write a program to read a file stream.

import sys, io

zen = io.StringIO()

old\_stdout = sys.stdout

sys.stdout = zen

import this

sys.stdout = old\_stdout

f=open('zen.txt','w')

f.write(zen.getvalue())

f.close()

1. Write a program to read a file a just to a particular index using seek().

fi = open("text.txt", "r")

fi.seek(30)

print(fi.tell())

print(fi.readline())

fi.close()

1. Write a program to check whether a file is having read access and write access permissions.

import os.path

def check\_file\_writable(fnm):

if os.path.exists(fnm):

if os.path.isfile(fnm):

return os.access(fnm, os.W\_OK)

else:

return False

pdir = os.path.dirname(fnm)

if not pdir: pdir = '.'

return os.access(pdir, os.W\_OK)

if \_\_name\_\_ == '\_\_main\_\_':

from sys import argv

print argv[1], '=>', check\_file\_writable(argv[1])

Constructors

1. Write a class with a default constructor, one argument constructor and two argument constructors. Instantiate the class to call all the constructors of that class from a main class.

2. Call the constructors (both default and argument constructors) of super class from a child class.

3. Apply private, public, protected and default access modifiers to the constructor.

4. Write a program which illustrates the concept of attributes of a constructor.

from unicodedata import name

class A:

def \_\_init\_\_(self):

self.name = "Kashish"

def print\_A(self):

print(self.name)

obj = A()

obj.print\_A()

class B(A):

def \_\_init\_\_(self):

self.name = "KG"

def print\_B(self):

print(self.name)

obj1 = B()

obj1.print\_B()

class C:

name = None

\_roll = None

\_\_branch = None

def \_\_init\_\_(self,name,roll,branch):

self.name = name

self.\_roll = roll

self.\_\_branch = branch

def dsiplayName(self):

print("Name:",self.name)

def \_displayRoll(self):

print("Roll:",self.\_roll)

def \_\_displayBranch(self):

print("Branch:",self.\_\_branch)

def access\_\_displayBranch(self):

self.\_\_displayBranch()

class D(C):

def \_\_init\_\_(self,name, roll, branch):

super().\_\_init\_\_(name,roll, branch)

def access\_displayRoll(self):

self.\_displayRoll()

obj = D("Kashish", 5 , "CSE")

obj.dsiplayName()

obj.access\_displayRoll()

obj.access\_\_displayBranch()

Exceptions

1. Write a program to generate Arithmetic Exception without exception handling

2. Handle the Arithmetic exception using try-catch block

3. Write a method which throws exception, call that method in main class without try block

4. Write a program with multiple catch blocks

5. Write a program to throw exception with your own message

6. Write a program to create your own exception

7. Write a program with finally block

8. Write a program to generate Arithmetic Exception

9. Write a program to generate FileNotFoundException

10. Write a program to generate ClassNotFoundException

11. Write a program to generate IOException

12. Write a program to generate NoSuchFieldException.

a = [1, 2, 3]

try:

print ("Second element = ",a[1])

print ("Fourth element = ",a[3])

except:

print ("An error occurred")

print()

b = [3,2,1]

try:

a == b

except:

print("They are not equal")

else:

print("Both Equal")

print()

try:

k = 5/0

print(k)

except ZeroDivisionError:

print("Can't divide by zero")

finally:

print('This is always executed')

Dictionary

1. Create a Dictionary with at least 5 key value pairs of the Student ID and Name

1.1. Adding the values in dictionary

1.2. Updating the values in dictionary

1.3. Accessing the value in dictionary

1.4. Create a nested loop dictionary

1.5. Access the values of nested loop dictionary

1.6. Print the keys present in a particular dictionary

1.7. Delete a value from a dictionary.

Dict = dict([(1,'Kashish'), (2,'Kritika'), (3,'Aastha'), (4,'Vaishali'), (5,'Muskan')])

print("Dictionary with each item as a pair:",Dict) #printing dictionary

Dict[6] = 'Nitya'

print("Dictionary with new item added:",Dict)

Dict[3] = 'Navdisha'

print("Dictionary with updated values:",Dict)

print("Accessing one value in Dictionary:",Dict[1])

del Dict[6]

print("Delete a value from a Dictionary:",Dict)

Dict1 = {1: 'Kashish', 2: 'Kritika',

3:{'Age' : 18, 'Branch' : 'CSE', 'Year' : 'Third Year'}}

print("Nested loop Dictionary:",Dict1)

print("Accessing an element of a Nested Dictionary:",Dict1[2])