

**K.J. Somaiya Polytechnic, Mumbai-77**

**Batch No:C1**

**Enrollment No.: FCOG19117**

**Experiment No:1**

**Experiment Name: Introduction to Python**

**K.J. Somaiya Polytechnic, Mumbai-77**

**Experiment No.1**

**Experiment Name: Introduction to Python**

**Theory:**

Sometimes when working with some kind of financial or scientific projects it becomes necessary to implement mathematical calculations in the project. Python provides the **math module** to deal with such calculations. Math module provides functions to deal with both basic operations such as addition(+), subtraction(-), multiplication(\*), division(/) and advance operations like trigonometric, logarithmic, exponential functions.

Write python programs to understand Expressions, Variables, basic Math operations, Strings, Basic String Operation & String Methods. ( Minimum Three Programs based on math operations, Strings)

**Implementation & Output:**

String1 = 'Welcome to the Geeks World' print("String with the use of Single Quotes: ") print(String1)

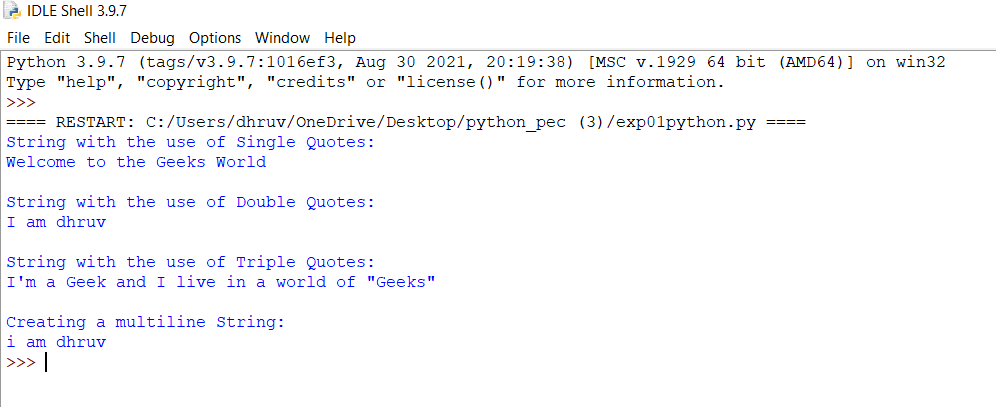
String1 = "I am dhruv"

print("\nString with the use of Double Quotes: ") print(String1)

String1 = '''I'm a Geek and I live in a world of "Geeks"''' print("\nString with the use of Triple Quotes: ") print(String1)

String1 = "i am dhruv" print("\nCreating a multiline String: ")

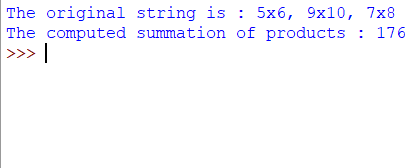
print(String1)



**from operator import mul test\_str = '5x6, 9x10, 7x8'**

**print("The original string is : " + str(test\_str))**

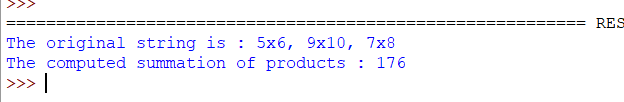
**res = sum(mul(\*map(int, ele.split('x'))) for ele in test\_str.split(', ')) print("The computed summation of products : " + str(res))**



**test\_str = '5x6, 9x10, 7x8'**

**print("The original string is : " + str(test\_str)) temp = test\_str.replace(',', '+').replace('x', '\*') res = eval(temp)**

**print("The computed summation of products : " + str(res))**

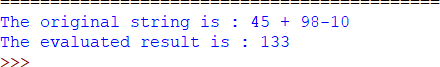


**import re**

**test\_str = "45 + 98-10"**

**print("The original string is : " + test\_str)**

**res = sum(map(int, re.findall(r'[+-]?\d+', test\_str))) print("The evaluated result is : " + str(res))**



**n=input("Enter a number ")**

**n=int(n)**

**flag=0**

**for i in range(2,n):**

**if n%i==0:**

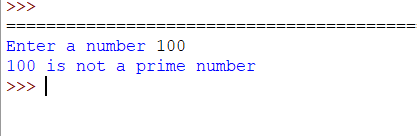
**flag=1**

**print("%d is not a prime number"%n)**

**break**

**if flag==0:**

**print("%d is a prime number"%n)**



**import math**

**a = 2.3**

**print ("The ceil of 2.3 is : ", end="") print (math.ceil(a))**

**print ("The floor of 2.3 is : ", end="") print (math.floor(a))**

**a = 5**

**print("The factorial of 5 is : ", end="") print(math.factorial(a))**

**print ("The value of 3\*\*4 is : ",end="")**

**print (pow(3,4)) num1 = 1.5**

**num2 = 6.3**

**sum = num1 + num2**

**print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))**

**num1 = int(input("Enter First Number: ")) num2 = int(input("Enter Second Number: "))**

**print("Enter which operation would you like to perform?")**

**ch = input("Enter any of these char for specific operation +,-,\*,/: ")**

**result = 0 if ch == '+':**

**result = num1 + num2 elif ch == '-':**

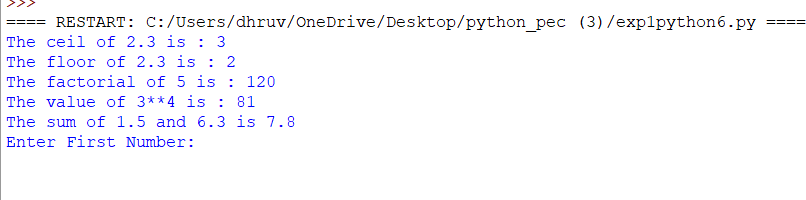
**result = num1 - num2 elif ch == '\*':**

**result = num1 \* num2 elif ch == '/':**

**result = num1 / num2 else:**

**print("Input character is not recognized!")**

**print(num1, ch , num2, ":", result)**



**Conclusion:** n Python, **Strings** are arrays of bytes representing Unicode characters. However, Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string



**K.J. Somaiya Polytechnic, Mumbai-77**

**Batch No:C1**

**Enrollment No.: FCOG19117**

**Experiment No:2**

**Experiment Name: Introduction to Python**

**K.J. Somaiya Polytechnic, Mumbai-77**

**Experiment No.2**

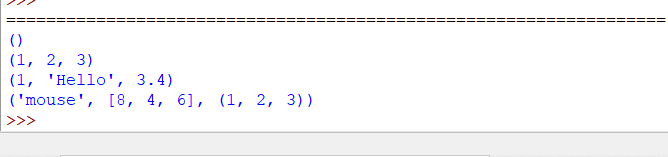
**Experiment Name: Introduction to Python**

**Theory:**

A Structure is one of the 5 data types in programming. A structure is **used to represent information about something more complicated than a single number, character, or boolean can do** (and more complicated than an array of the above data types can do).

Develop programs to learn different types of structures (list, dictionary, tuples, arrays) in python

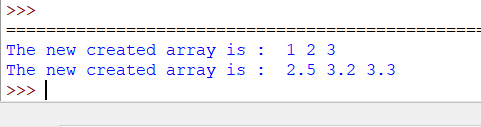
**Implementation & Output:**



**my\_tuple = () print(my\_tuple) my\_tuple = (1, 2, 3) print(my\_tuple)**

**my\_tuple = (1, "Hello", 3.4) print(my\_tuple)**

**my\_tuple = ("mouse", [8, 4, 6], (1, 2, 3)) print(my\_tuple)**



**import** array as arr

a **=** arr.array('i', [1, 2, 3])

**print** ("The new created array is : ", end **=**" ")

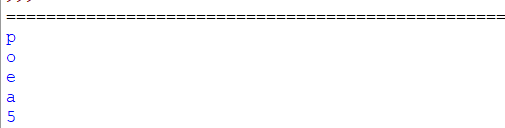
**for** i **in** range (0, 3): print (a[i], end **=**" ")

**print**()

b **=** arr.array('d', [2.5, 3.2, 3.3])

print ("The new created array is : ", end **=**" ")

**for** i **in** range (0, 3): print (b[i], end **=**" ")



**my\_list = ['p', 'r', 'o', 'b', 'e']**

**# first item print(my\_list[0]) # p**

**# third item print(my\_list[2]) # o**

**# fifth item print(my\_list[4]) # e**

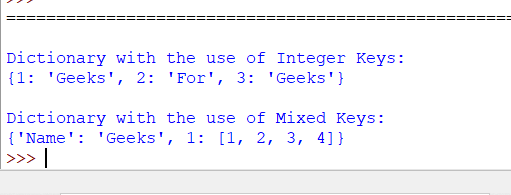
**# Nested List**

**n\_list = ["Happy", [2, 0, 1, 5]]**

**# Nested indexing print(n\_list[0][1])**

**print(n\_list[1][3])**

**# Error! Only integer can be used for indexing print(my\_list[4.0])**



**Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'} print("\nDictionary with the use of Integer Keys: ") print(Dict)**

**# Creating a Dictionary # with Mixed keys**

**Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]} print("\nDictionary with the use of Mixed Keys: ") print(Dict)**

**Conclusion:**

When a struct type is declared, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables. Another way of creating a struct variable



**K.J. Somaiya Polytechnic, Mumbai-77**

**Batch No:C1**

**Enrollment No.: FCOG18128**

**Experiment No:.3**

**Experiment Name: Introduction to Python**

**K.J. Somaiya Polytechnic, Mumbai-77**

**Experiment No.3**

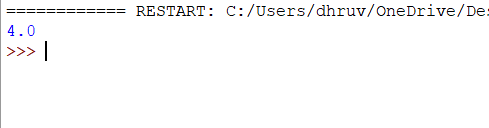
**Experiment Name: Introduction to Python**

**Theory:**

Normally, a library is a collection of books or is a room or place where many books are stored to be used later. Similarly, in the programming world, a library is a collection of precompiled codes that can be used later on in a program for some specific well-defined operations. Other than pre-compiled codes, a library may contain documentation, configuration data, message templates, classes, and values, etc.A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer. As we don’t need to write the same code again and again for different programs. Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, et

Develop a program using built in libraries.

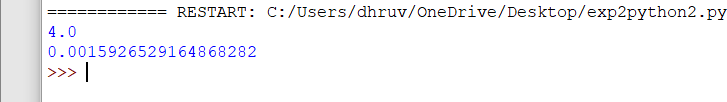
**Implementation & Output:**



**import math**

**A = 16**

**print(math.sqrt(A)**



**from math import sqrt, sin**

**A = 16**

**B = 3.14**

**print(sqrt(A)) print(sin(B))**



**import sys, math**

**def hash\_fraction(m, n):**

**"Compute the hash of a rational number m / n.Assumes m and n are integers, with n positive.Equivalent to hash(fractions.Fraction(m, n))"**

**P = sys.hash\_info.modulus**

**while m % P == n % P == 0: m, n = m // P, n // P**

**if n % P == 0:**

**hash\_value = sys.hash\_info.inf else:**

**hash\_value = (abs(m) % P) \* pow(n, P - 2, P) % P if m < 0:**

**hash\_value = -hash\_value if hash\_value == -1:**

**hash\_value = -2 return hash\_value**

**def hash\_float(x):**

**"Compute the hash of a float x."**

**if math.isnan(x):**

**return object.\_\_hash (x**

**return sys.hash\_info.inf if x > 0 else -sys.hash\_info.inf else:**

**return hash\_fraction(\*x.as\_integer\_ratio())**

**def hash\_complex(z):**

**"Compute the hash of a complex number z."**

**hash\_value = hash\_float(z.real) + sys.hash\_info.imag \* hash\_float(z.imag) M = 2\*\*(sys.hash\_info.width - 1)**

**hash\_value = (hash\_value & (M - 1)) - (hash\_value & M) if hash\_value == -1:**

**hash\_value = -2 return hash\_value**

**Conclusion:** The Python Standard Library contains the exact syntax, semantics, and tokens of Python. It contains built-in modules that provide access to basic system functionality like I/O and some other core modules



**K.J. Somaiya Polytechnic, Mumbai-77**

**Batch No:C1**

**Enrollment No.: FCOG19117**

**Experiment No:.4**

**Experiment Name: Decision Making and Functions**

K.J. Somaiya Polytechnic, Mumbai-77

**Experiment No.4**

**Experiment Name: Decision Making and Functions**

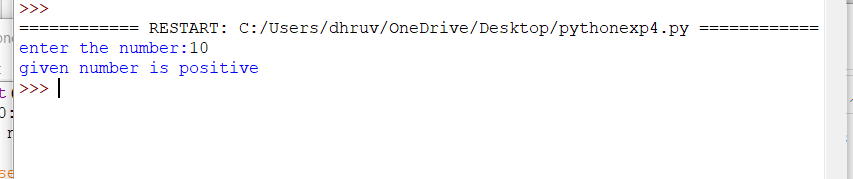
**Theory:**

Decision making is globally thought to be selection from alternatives. It is deeply related with all the traditional functions of a manager, such as **planning, organizing, staffing, directing and controlling**. When he performs these functions, he makes decisions.

. A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called user-defined functions*.*

**Implementation & Output:**



**n=float(input('enter the number:')) if n>=0:**

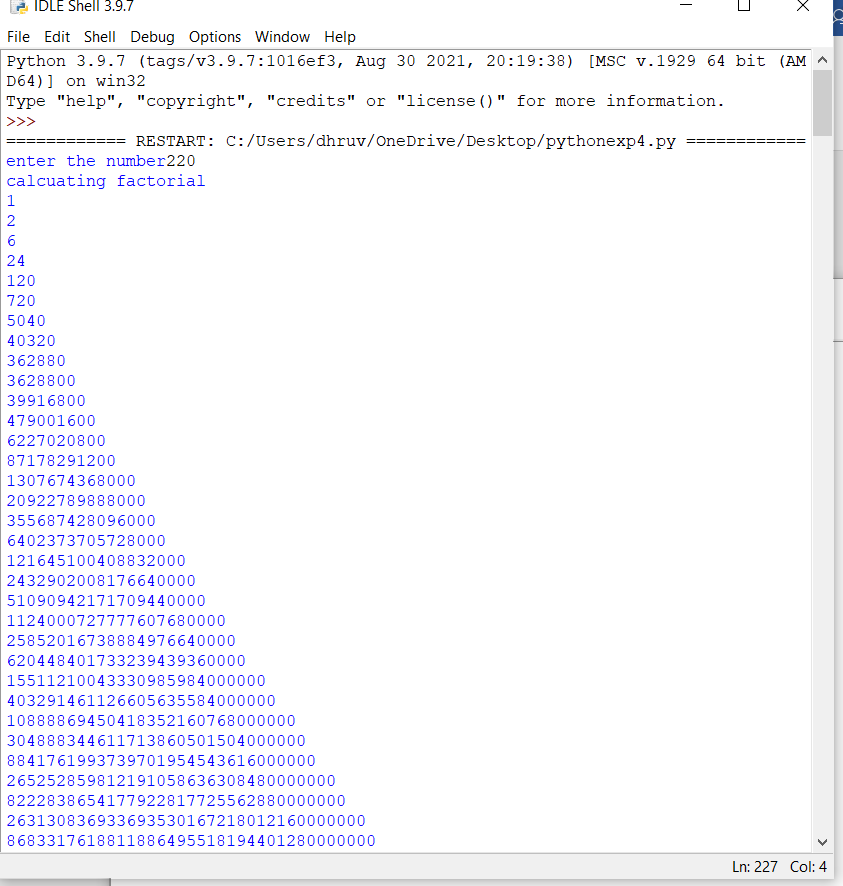
**if n==0:**

**print('given number is zero') else:**

**print('given number is positive')**

**else:**

**print('given number is negative')**



**n=int(input('enter the number')) factorial=1**

**i=1**

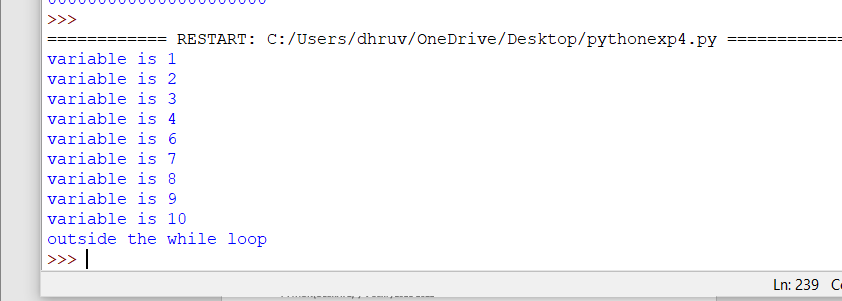
**if n<=0:**

**print('number is negative and zero') else:**

**print('calcuating factorial') while i<=n:**

**factorial=factorial\*i i=i+1**

**print(factorial)**

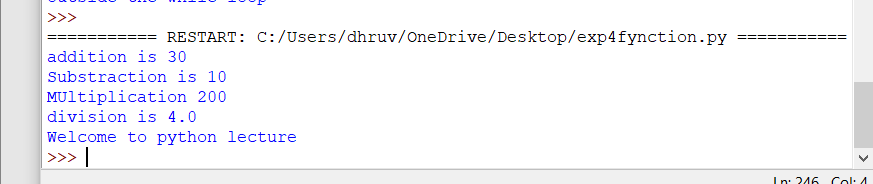


**var=0**

**while var<10: var=var+1 if var==5:**

**continue print('variable is',var)**

**print('outside the while loop')**



**def cal(): y=10+20 a=30-20 c=20\*10**

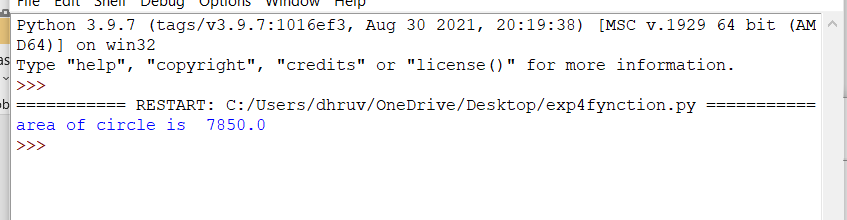
**d=20/5**

**print("addition is",y) print("Substraction is",a) print("MUltiplication",c) print("division is",d)**

**def display():**

**print("Welcome to python lecture") cal()**

**display()**



**def area(r): if r<=0:**

**print("warning : radius cannot be less than zero") else:**

**a=3.14\*r\*\*2 return a**

**print("area of circle is ",area(50))**

**Conclusion:** Itis the simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not

i.e if a certain condition is true then a block of statement is executed otherwise



K.J. Somaiya Polytechnic, Mumbai-77

Batch No:C1

Enrollment No.: FCOG19117

Experiment No:.5

**Experiment Name: Object Oriented Concepts in Python programming**

K.J. Somaiya Polytechnic, Mumbai-77

**Experiment No.5**

**Experiment Name: Object Oriented Concepts in Python programming**

Theory:

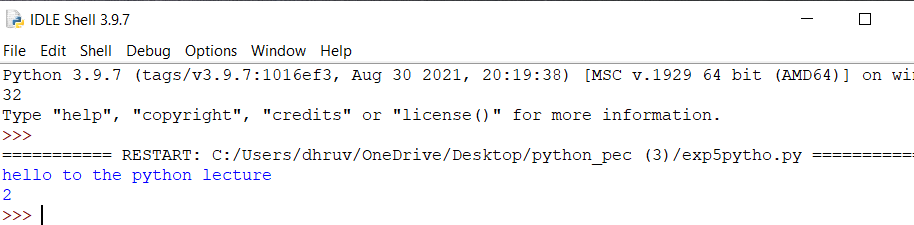
Objected oriented programming as a discipline has gained a universal following among developers.

Python, an in-demand [programming](https://www.edureka.co/blog/python-programming-language) [language](https://www.edureka.co/blog/python-programming-language) also follows an object-oriented

programming paradigm. It deals with declaring Python classes and objects which lays the foundation of OOPs concepts. This article on “object oriented programming python” will walk you through declaring [python classes,](https://www.edureka.co/blog/python-class/) instantiating objects from them along with the four methodologies of OOPs

Implementation & Output:

Write python programs to understand classes and objects, (minimum 2 programs to create classes and objects



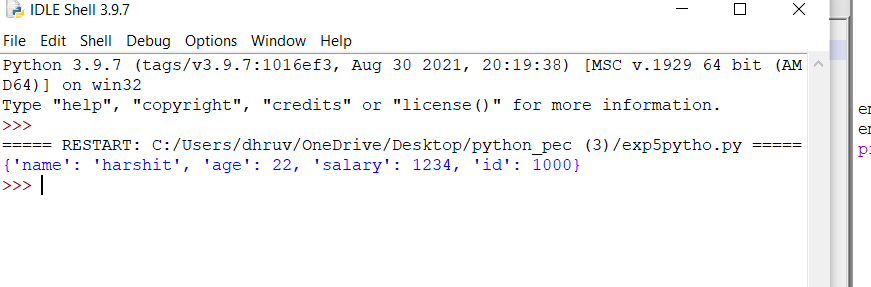
class ty(): y=2

v=4

def show(self):

print("hello to the python lecture") t=ty()

t.show() print(t.y)



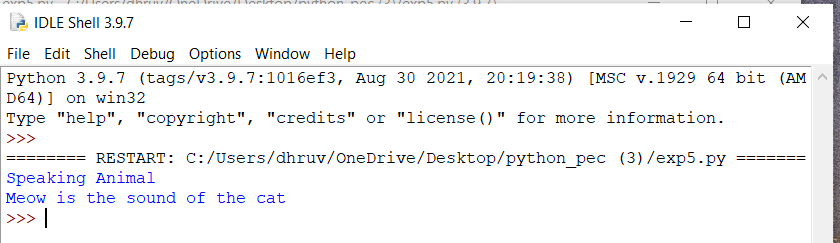
class employee():

def init (self,name,age,id,salary): self.name = name

self.age = age self.salary = salary self.id = id

emp1 = employee("harshit",22,1000,1234)

emp2 = employee("arjun",23,2000,2234) print(emp1. dict )



class Animal:

def speak(self): print("Speaking Animal")

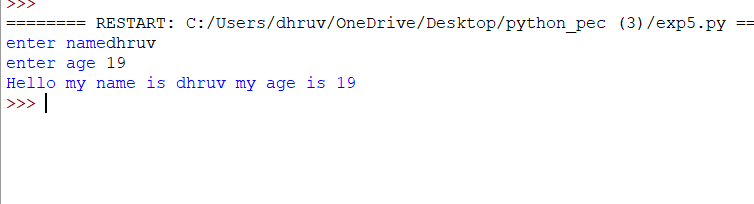
class cat(Animal):

def meow(self):

print("Meow is the sound of the cat") c=cat()

c.speak()

c.meow()



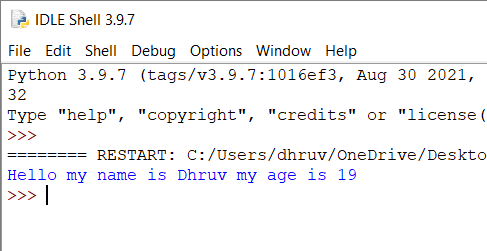
class Person:

def init\_\_(self,name,age): self.name=name self.age=age

def display\_data(self):

print("Hello my name is "+self.name+" my age is "+str(self.age))

p=Person("Dhruv",19) p.display\_data()



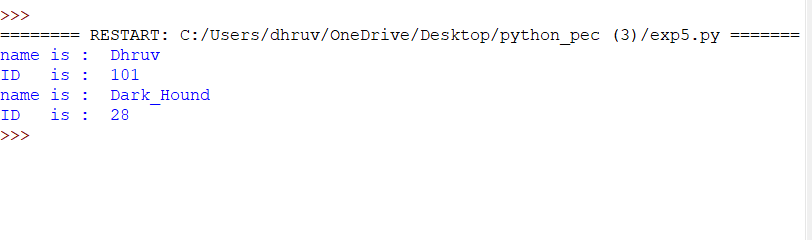
class Person:

def init\_\_(self,name,age): self.name=name self.age=age

def display\_data(self):

print("Hello my name is "+self.name+" my age is "+str(self.age))

p=Person("Dhruv",19) p.display\_data()



class student:

def get\_data(self,name,id): self.name=name self.id=id

def display(self):

print("name is : ",self.name) print("ID is : ",self.id)

s1=student() s1.get\_data("Dhruv","101") s1.display()

s2=student() s2.get\_data("Dark\_Hound","28") s2.display()

**Conclusion:** What is class and object in OOP? A Class in object oriented programming is a blueprint or prototype that defines the variables and the methods (functions) common to all Java Objects of a certain kind. An object in OOPS is **a specimen of a class**. Software objects are often used to model real-world objects you find in everyday life.



K.J. Somaiya Polytechnic, Mumbai-77

Batch No:C1

Enrollment No.: FCOG19117

Experiment No:.6

**Experiment Name: Object Oriented Concepts in Python programming**

K.J. Somaiya Polytechnic, Mumbai-77

**Experiment No.8**

**Experiment Name: Object Oriented Concepts in Python programming**

Theory:

Objected oriented programming as a discipline has gained a universal following among developers.

Python, an in-demand [programming](https://www.edureka.co/blog/python-programming-language) [language](https://www.edureka.co/blog/python-programming-language) also follows an object-oriented

programming paradigm. It deals with declaring Python classes and objects which lays the foundation of OOPs concepts. This article on “object oriented programming python” will walk you through declaring [python classes,](https://www.edureka.co/blog/python-class/) instantiating objects from them along with the four methodologies of OOPs

Implementation & Output:

Write a python program to implement multiple inheritances.

class Animal:

def speak(self): print("Speaking Animal")

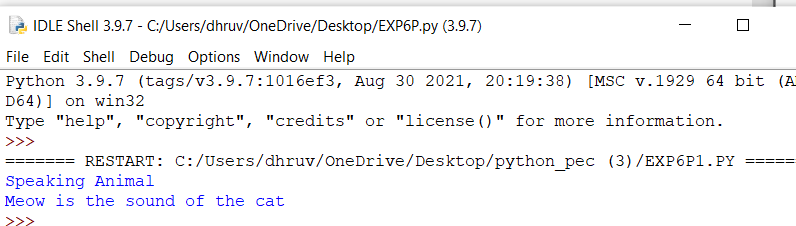
class cat(Animal):

def meow(self):

print("Meow is the sound of the cat") c=cat()

c.speak()

c.meow()



-

class base1:

def num1(self):

self.n1=10

print("value of no in first base class:",self.n1) class base2:

def num2(self): self.n2=20

print("value of no in second base class :",self.n2) class der(base1,base2):

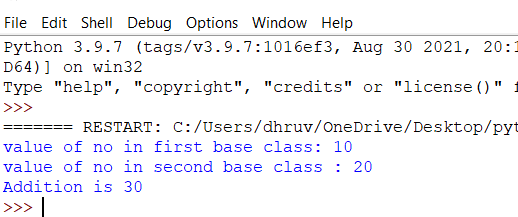
def add(self):

return self.n1+self.n2 d=der()

d.num1()

d.num2()

print("Addition is",d.add())



Conclusion:

A Class in object oriented programming is a blueprint or prototype that defines the variables and the methods (functions) common to all Java Objects of a certain kind. An object in OOPS is **a specimen of a class**. Software objects are often used to model real-world objects you find in everyday life.



K.J. Somaiya Polytechnic, Mumbai-77

Batch No:C1

Enrollment No.: FCOG19117

Experiment No:.7

**Experiment Name: Object Oriented Concepts in Python programming**

K.J. Somaiya Polytechnic, Mumbai-77

**Experiment No.7**

**Experiment Name: Object Oriented Concepts in Python programming**

Theory:

Objected oriented programming as a discipline has gained a universal following among developers.

Python, an in-demand [programming](https://www.edureka.co/blog/python-programming-language) [language](https://www.edureka.co/blog/python-programming-language) also follows an object-oriented

programming paradigm. It deals with declaring Python classes and objects which lays the foundation of OOPs concepts. This article on “object oriented programming python” will walk you through declaring [python classes,](https://www.edureka.co/blog/python-class/) instantiating objects from them along with the four methodologies of OOPs

Implementation & Output:

Develop programs for method overloading and overriding.

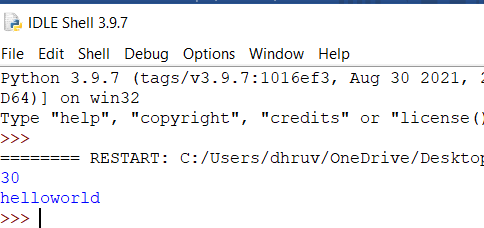
class demo:

def init (self,a): self.a=a

def add (self,o): return self.a+o.a

obj1=demo(10) obj2=demo(20) print(obj1+obj2) obj3=demo("hello")

obj4=demo("world") print(obj3+obj4)



def add(datatype, \*args):

# if datatype is int

# initialize answer as 0 if datatype =='int':

answer = 0

# if datatype is str

# initialize answer as '' if datatype =='str':

answer =''

# Traverse through the arguments for x in args:

# This will do addition if the

# arguments are int. Or concatenation # if the arguments are str

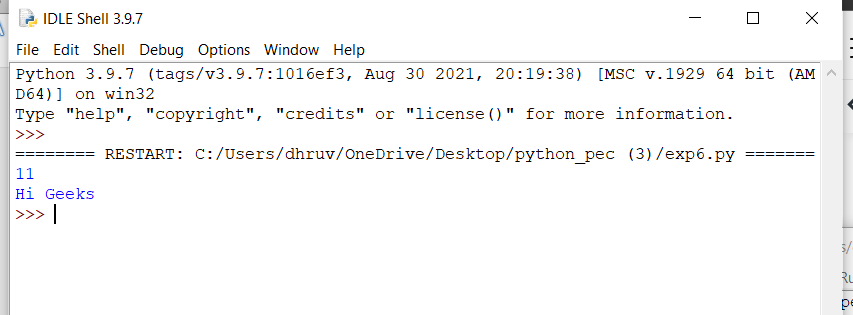
answer = answer + x

print(answer)

# Integer add('int', 5, 6)

# String

add('str', 'Hi ', 'Geeks')



class base: def f1(self):

print("function 1 of base class") def f2(self):

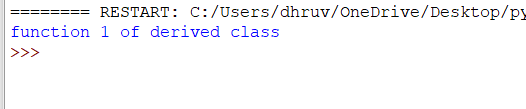
print("function 2 of base class") class derived(base):

def f1(self):

print("function 1 of derived class") def f3(self):

print("function 3 of derived class") d=derived()

d.f1()



Conclusion:

A Class in object oriented programming is a blueprint or prototype that defines the variables and the methods (functions) common to all Java Objects of a certain kind. An object in OOPS is **a specimen of a class**. Software objects are often used to model real-world objects you find in everyday life.



K.J. Somaiya Polytechnic, Mumbai-77

Batch No:C1

Enrollment No.: FCOG19117

Experiment No:.8

**Experiment Name: Object Oriented Concepts in Python programming**

K.J. Somaiya Polytechnic, Mumbai-77

**Experiment No.8**

**Experiment Name: Object Oriented Concepts in Python programming**

Theory:

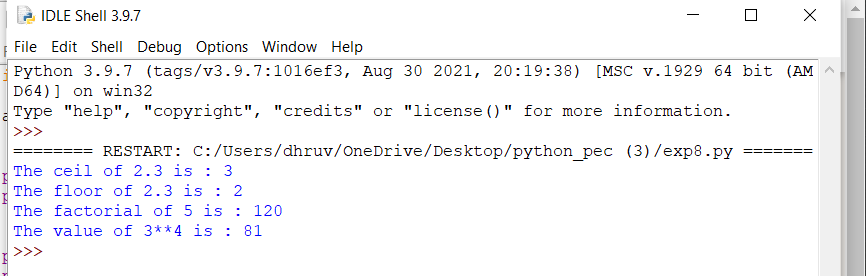
Objected oriented programming as a discipline has gained a universal following among developers.

Python, an in-demand [programming](https://www.edureka.co/blog/python-programming-language) [language](https://www.edureka.co/blog/python-programming-language) also follows an object-oriented

programming paradigm. It deals with declaring Python classes and objects which lays the foundation of OOPs concepts. This article on “object oriented programming python” will walk you through declaring [python classes,](https://www.edureka.co/blog/python-class/) instantiating objects from them along with the four methodologies of OOPs

Implementation & Output:

Develop programs to understand working of exception handing.



import math a = 2.3

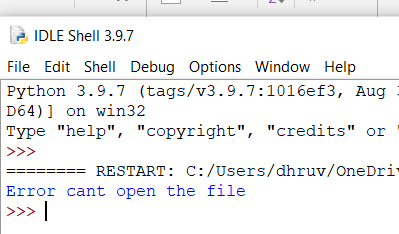
print ("The ceil of 2.3 is : ", end="") print (math.ceil(a))

print ("The floor of 2.3 is : ", end="") print (math.floor(a))7ujm

a = 5

print("The factorial of 5 is : ", end="") print(math.factorial(a))

print ("The value of 3\*\*4 is : ",end="") print (pow(3,4))



try:

f=open("test","r") f.write("Hello World") except IOError:

print("Error cant open the file") else:

print("Data written in the file successfully")

Conclusion:

A Class in object oriented programming is a blueprint or prototype that defines the variables and the methods (functions) common to all Java Objects of a certain kind. An object in OOPS is **a specimen of a class**. Software objects are often used to model real-world objects you find in everyday life.



K.J. Somaiya Polytechnic, Mumbai-77

Batch No:C1

Enrollment No.: FCOG19117

Experiment No:.9

**Experiment Name:** Object Oriented Concepts in Python programming

K.J. Somaiya Polytechnic, Mumbai-77

**Experiment No.9**

**Experiment Name:** Object Oriented Concepts in Python programming

Theory:

Sequence of characters used for defining a pattern. This pattern could be used for searching, replacing and other operations. Regex is extensively utilized in applications that require input validation, Password validation, Pattern Recognition, search and replace utilities (found in word processors) etc. This is due to the fact that regex syntax stays the same across different programming languages and implementations. Therefore, one having the grasp of it provides longevity across languages. In this article, we will be creating a program for checking the validity of a regex string.

The method we would be using will require a firm understanding of the *try- except* construct of python. Therefore, it would be wise if we touch upon that before moving over to the actual code.

Implementation & Output:

**program to validate the filed using regular expression pressions in python**

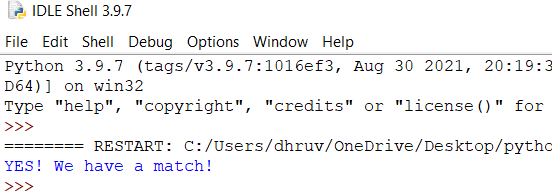
import re

txt = "The rain in Spain"

x = re.search("^The.\*Spain$", txt) if x:

print("YES! We have a match!") else:

print("No match")



-

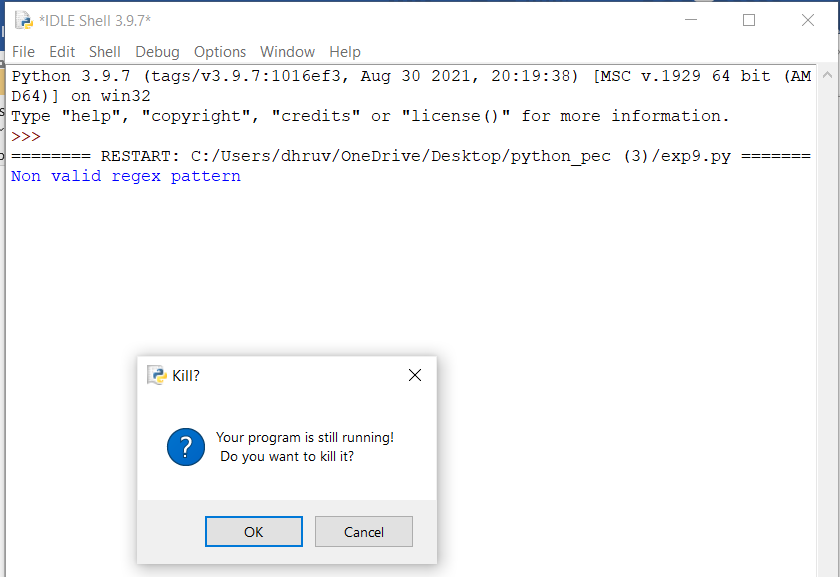
**import re**

**pat = re.compile(r"[A-Za-z0-9]+") test = input("Enter the string: ") if re.fullmatch(pat, test):**

**print(f"'{test}' is an alphanumeric string!")**

**else:**

**print(f"'{test}' is NOT a alphanumeric string!")**



-

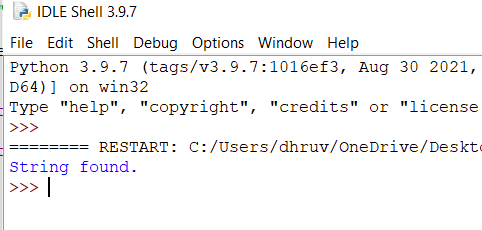
import re

input = raw\_input("Enter an input string:") m = re.match('\d{5}\Z',input)

if m:

print("True") else:

print("False")



**Conclusion:**

A regular expression is **a special sequence of characters that helps you match or find other strings or sets of strings**, using a specialized syntax held in a pattern. The Python module re provides full support for Perl-like regular expressions in Python. The re module raises the exception re



K.J. Somaiya Polytechnic, Mumbai-77

Batch No:C1

Enrollment No.: FCOG19117

Experiment No:.10

**Experiment Name: File Handling**

K.J.Somaiya Polytechnic, Mumbai-77

**Experiment No.10**

**Experiment Name: File Handling**

**Theory:**

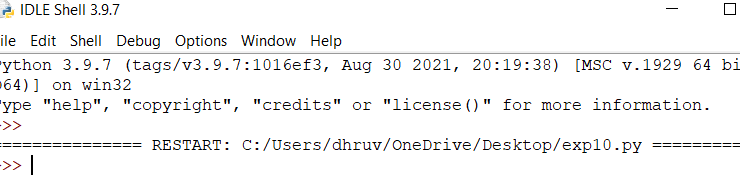
Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let’s start with Reading and Writing files.

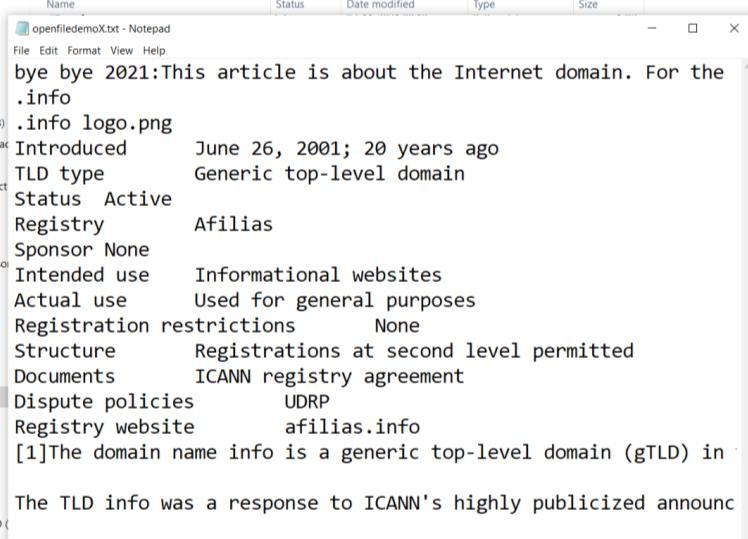
Implementation & Output:

Write python programs to understand different File handing Operations.

f=open("openfiledemoX.txt","w")

f.write ("hi to python lec ,you have pass the exam") f.close()



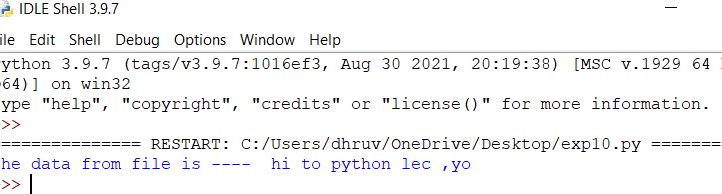


-

f=open("openfiledemoX.txt","r+") s=f.read(20)

print("the data from file is ",s)

f.write("bye bye 2021:") f.close()



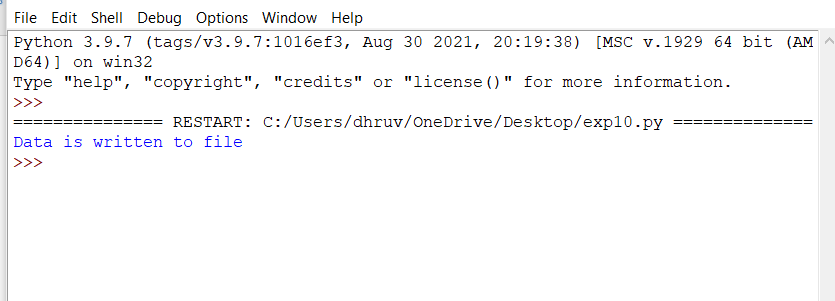
-

f=open("test222.txt","w")

f.write("oooo ooo ooooo ooo oooo oooooooooooop ugb\n lets celebrate festival")

f.close()

print("Data is written to file")



Conclusion:

File handle is like a cursor, **which defines from where the data has to be read or written in the file**. There are 6 access modes in python This is also the default mode in which file is opened. Read and Write ('r+') : Open the file for reading and writing. The handle is positioned at the beginning of the file