**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **srno** | **Name** | **Page no** | **remark** |
|  | WAP to demonstrate overloading class |  |  |
|  | WAP to find factorial using return function |  |  |
|  | WAP to find prime or not (user input) |  |  |
|  | WAP to find prime numbers between 100 |  |  |
|  | WAP to denomstrate types of variables |  |  |
|  | WAP program to demonstrate Arithmatic Operations |  |  |
|  | WAP to check if the number is an Armstrong number or not |  |  |
|  | WAP to demonstrate String Methods |  |  |
|  | WAP to demonstrate list methods. |  |  |
|  | WAP to raise zeroDivision Exception with sutaible example. |  |  |
|  | WAP to raise zeroDivision Exception with sutaible example. |  |  |
|  | WAP to create Multiplication table (user input number) |  |  |
|  | WAP to demonstrate multilevel inheritance. |  |  |
|  | WAP to demonstrate multiple inheritance |  |  |
|  | WAP to demonstrate hierarchical inheritance. |  |  |
|  | WAP to demonstrate Hybrid\_inheritance |  |  |
|  | WAP to demonstrate polymorphism |  |  |
|  | WAP to Demonstrate operator overloading |  |  |
|  | WAP to swap value using two variable |  |  |
|  | Write a program to display the Fibonacci sequence(using recursion) |  |  |
|  | WAP to Remove extra chater from the string |  |  |

1.WAP to demonstrate overloading class

# parent class

class Person:

  #costructor made of two varriables

  def \_\_init\_\_(self, fname, lname):

    self.firstname = fname

    self.lastname = lname

  def printname(self):

    print(self.firstname, self.lastname)

#child class Student drived from Person

class Student(Person):

  def \_\_init\_\_(self, fname, lname, year):

    super().\_\_init\_\_(fname, lname)

    self.graduationyear = year

  def welcome(self):

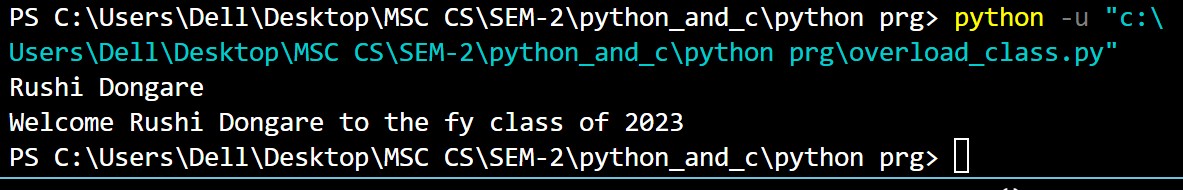
    print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

x = Student("Mike", "Olsen",2019)

x.printname() #this function is drived from its base class

x.welcome()

Output



2.WAP to find factorial using return function

# funtion return withoght argument

def find\_factorial():

    num=int(input("Enter a number to find factorial :"))

    fact=1

    for i in range(1,num+1):

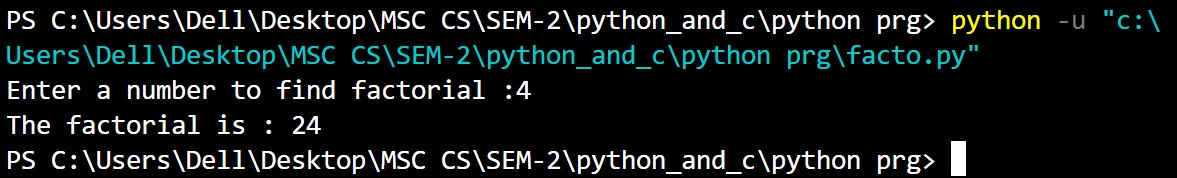
        fact=fact\*i

    return (fact)

f1=find\_factorial()

print("The factorial is :",f1)

Output:



3.WAP to find prime or not (user input)

num=int(input("Enter any Number :"))

flag=0

if (num==1):

    print(num," is not prime number")

elif (num>=2):

    for i in range(2,num):

        if (num%i==0):

            flag=1

            break

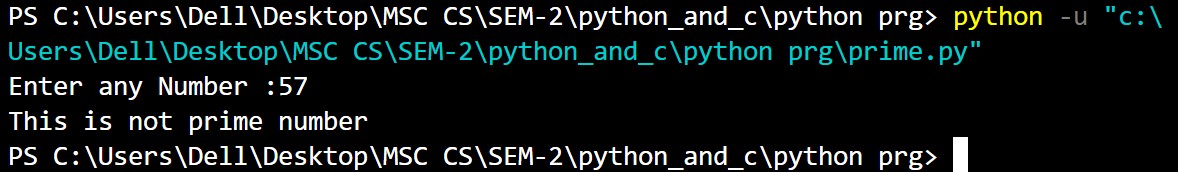
    if (flag==0):

        print("This is prime number")

    else:

        print("This is not prime number")

Output:



4.WAP to find prime numbers between 100

for num in range(2,100):

    flag=0

    if (num==1):

        print(num," is not prime number")

    elif (num>=2):

        for i in range(2,num):

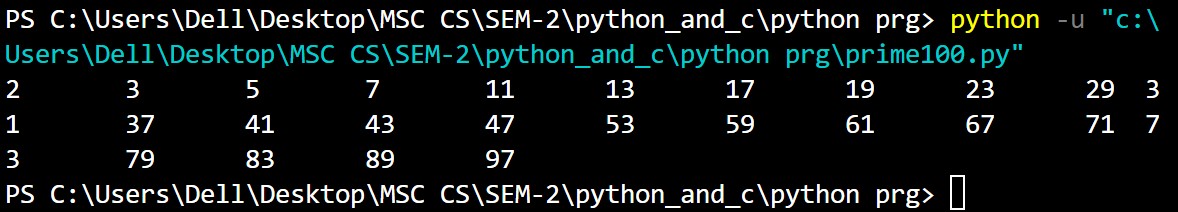
            if (num%i==0):

                flag=1

                break

        if (flag==0):

            print(num, end="\t")

Output:

5.WAP to types of variables

# integer variable.

a=100

print("The type of variable having value", a, " is ", type(a))

# float variable.

b=20.345

print("The type of variable having value", b, " is ", type(b))

# complex variable.

c=10+3j

print("The type of variable having value", c, " is ", type(c))

#bool

d=True

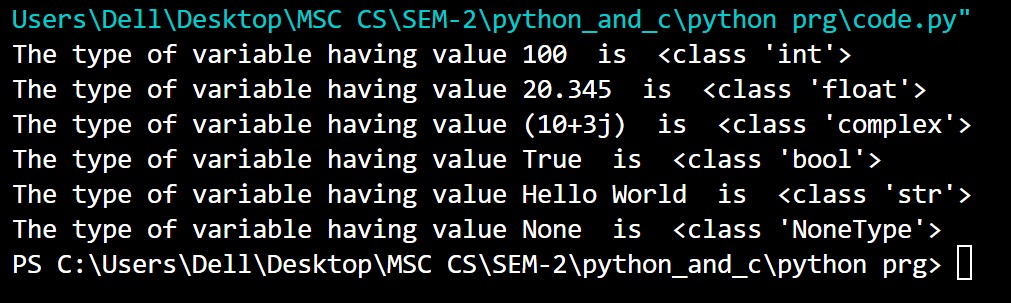
print("The type of variable having value", d, " is ", type(d))

e= “Hello World”

print("The type of variable having value", e, " is ", type(e))

f=None

print("The type of variable having value", f, " is ", type(f))

Output:

6.WAP program to demonstrate Arithmatic Operations

a = 21

b = 10

# Addition

print ("a + b : ", a + b)

# Subtraction

print ("a - b : ", a - b)

# Multiplication

print ("a \* b : ", a \* b)

# Division

print ("a / b : ", a / b)

# Modulus

print ("a % b : ", a % b)

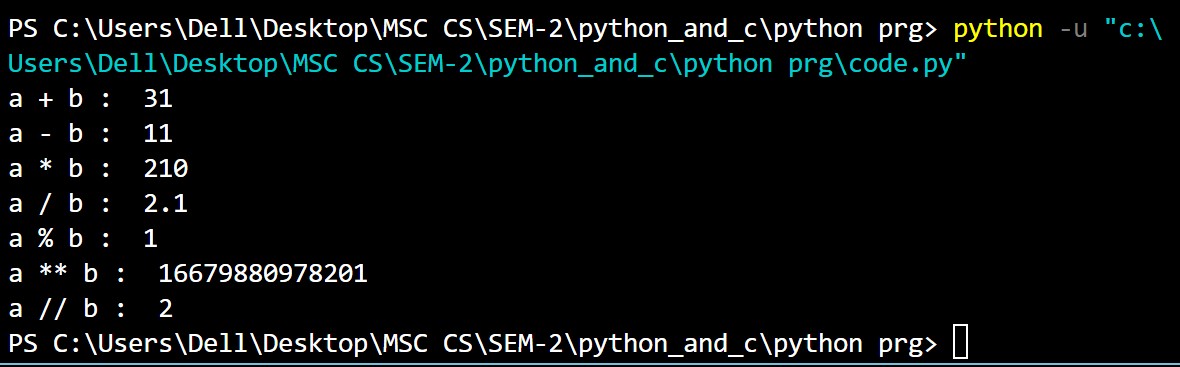
# Exponent

print ("a \*\* b : ", a \*\* b)

# Floor Division

print ("a // b : ", a // b)

Output:



7. WAP to check if the number is an Armstrong number or not

# take input from the user

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

# initialize sum

sum = 0

# find the sum of the cube of each digit

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* 3

temp //= 10

# display the result

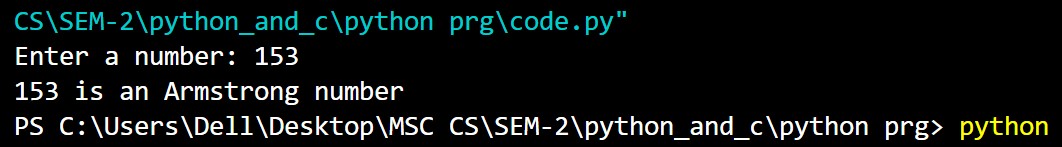
if num == sum:

print(num,"is an Armstrong number")

else:

print(num,"is not an Armstrong number")

Output:



8.WAP to demonstrate String Methods

str1 = " AbcDEfghIJ "

print(str1.upper())

print(str1.lower())

print(str1.strip)

str2 = "Silver Spoon"

print(str2.replace("Sp", "M"))

str2 = "Silver Spoon"

print(str2.split(" "))      #Splits the string at the whitespace " ".

str1 = "hello"

capStr1 = str1.capitalize()

print(capStr1)

str2 = "hello WorlD"

capStr2 = str2.capitalize()

print(capStr2)

str1 = "Welcome to the Console!!!"

print(str1.center(50))

str2 = "Abracadabra"

countStr = str2.count("a")

print(countStr)

Output:

ABCDEFGHIJ

abcdefghij

Silver Spoon

Silver Moon

['Silver', 'Spoon']

Hello

Hello world

            Welcome to the Console!!!

4

9.WAP to demonstrate list methods.

Colors1 = ["violet", "indigo", "blue", "green"]

Colors1.sort()

print(Colors1)

colors.sort(reverse=True)

print(colors)

colors = ["violet", "green", "indigo", "blue", "green"]

print(colors.index("green"))

print(colors.count("green"))

newlist = colors.copy()

print(colors)

print(newlist)

colors.append("purple")

print(colors)

colors.extend(Colors1)

print(colors)

10. WAP on decision making example

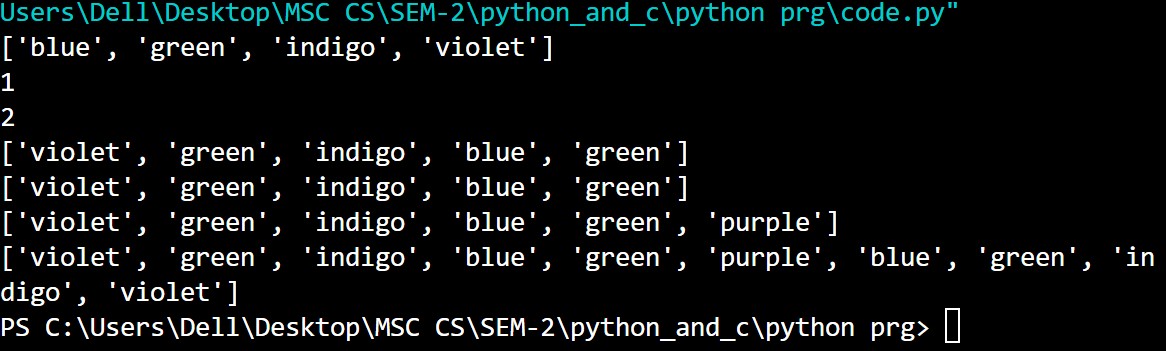
country = ("Spain", "Italy", "India", "England", "Germany")

if "Russia" in country:

print("Russia is present.")

else:

print("Russia is absent.")



11.WAP to raise zeroDivision Exception with sutaible example.

def divide\_numbers(x, y):

    try:

        result = x / y

        print("Result:", result)

    except ZeroDivisionError:

        print("The division by zero operation is not allowed.")

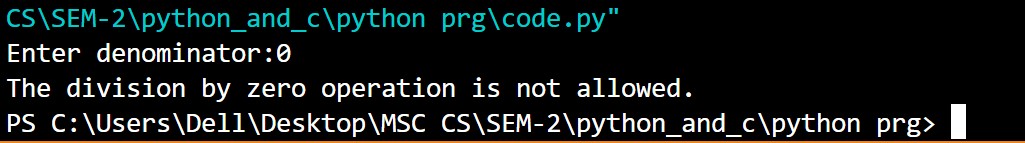
# Usage

numerator = 100

denominator =int(input("Enter denominator:"))

divide\_numbers(numerator, denominator)

Outpur:



12. WAP to create Multiplication table (user input number)

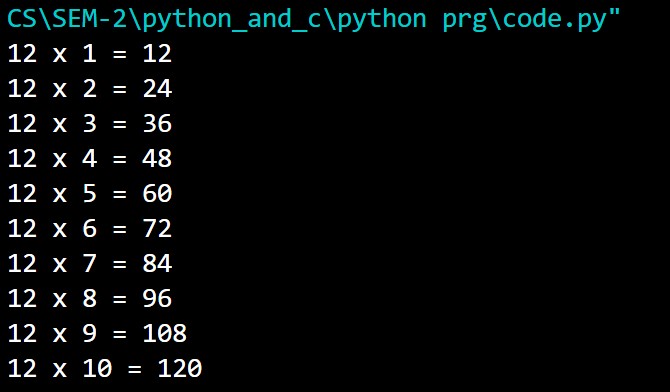
num = 12

# To take input from the user

# Iterate 10 times from i = 1 to 10

for i in range(1, 11):

print(num, 'x', i, '=', num\*i)

Output:

13.WAP to demonstrate multilevel inheritance

class Animal:

    def speak(self):

        print("Animal Speaking")

#The child class Dog inherits the base class Animal

class Dog(Animal):

    def bark(self):

        print("dog barking")

#The child class Dogchild inherits another child class Dog

class DogChild(Dog):

    def eat(self):

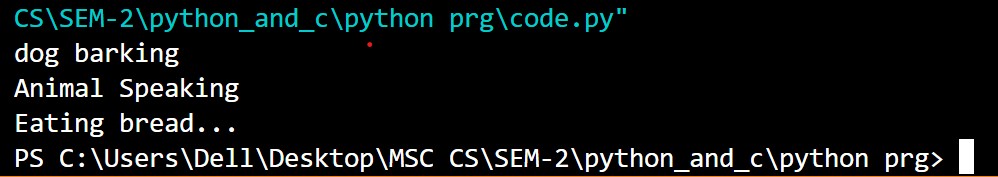
        print("Eating bread...")

d = DogChild()

d.bark()

d.speak()

d.eat()

Output:

14. WAP to demonstrate multiple inheritance

class Calculation1:

    def Summation(self,a,b):

        return a+b;

class Calculation2:

    def Multiplication(self,a,b):

        return a\*b;

class Derived(Calculation1,Calculation2):

    def Divide(self,a,b):

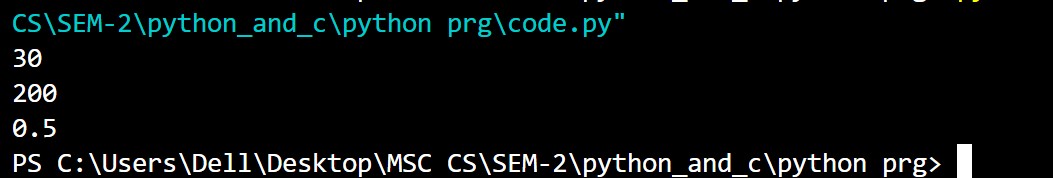
        return a/b;

d = Derived()

print(d.Summation(10,20))

print(d.Multiplication(10,20))

print(d.Divide(10,20))

Output:

15. WAP to demonstrate hierarchical inheritance

class Brands:                      #parent\_class

    brand\_name\_1 = "Amazon"

    brand\_name\_2 = "Ebay"

    brand\_name\_3 = "OLX"

class Products(Brands):            #child\_class

    prod\_1 = "Online Ecommerce Store"

    prod\_2 = "Online Store"

    prod\_3 = "Online Buy Sell Store"

class Popularity(Brands):        #grand\_child\_class

    prod\_1\_popularity = 100

    prod\_2\_popularity = 70

    prod\_3\_popularity = 60

class Value(Brands):

    prod\_1\_value = "Excellent Value"

    prod\_2\_value = "Better Value"

    prod\_3\_value = "Good Value"

obj\_1 = Products()          #Object\_creation

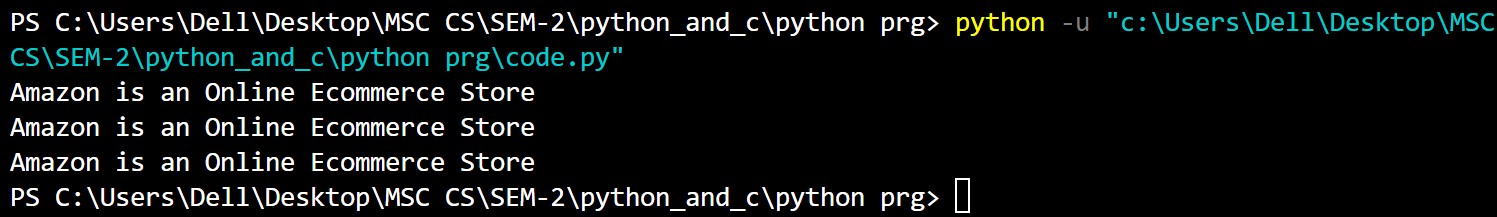
obj\_2 = Popularity()

obj\_3 = Value()

print(obj\_1.brand\_name\_1+" is an "+obj\_1.prod\_1)

print(obj\_1.brand\_name\_1+" is an "+obj\_1.prod\_1)

print(obj\_1.brand\_name\_1+" is an "+obj\_1.prod\_1)

Output:

16. WAP to demonstrate Hybrid\_inheritance

class PC:

def fun1(self):

print(“This is PC class”)

class Laptop(PC):

def fun2(self):

print(“This is Laptop class inheriting PC class”)

class Mouse(Laptop):

def fun3(self):

print(“This is Mouse class inheriting Laptop class”)

class Student(Mouse, Laptop):

def fun4(self):

print(“This is Student class inheriting PC and Laptop”)

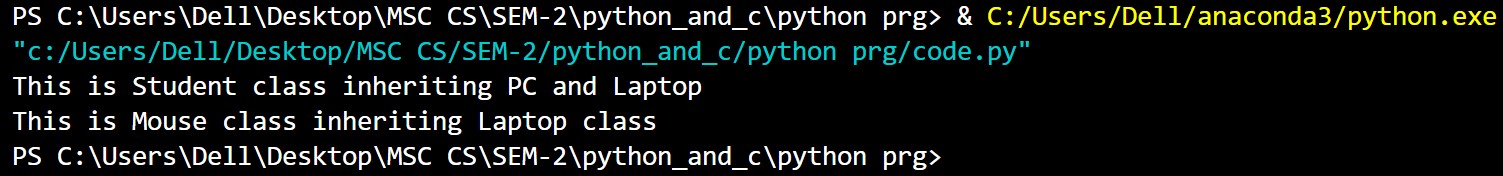
# Driver’s code

obj = Student()

obj1 = Mouse()

obj.fun4()

obj.fun3()

Output:

17.WAP to demonstrate polymorphism

class Vehicle:

  def \_\_init\_\_(self, brand, model):

    self.brand = brand

    self.model = model

  def move(self):

    print("Move!")

class Car(Vehicle):

  pass

class Boat(Vehicle):

  def move(self):

    print("Sail!")

class Plane(Vehicle):

  def move(self):

    print("Fly!")

car1 = Car("Ford", "Mustang") #Create a Car object

boat1 = Boat("Ibiza", "Touring 20") #Create a Boat object

plane1 = Plane("Boeing", "747") #Create a Plane object

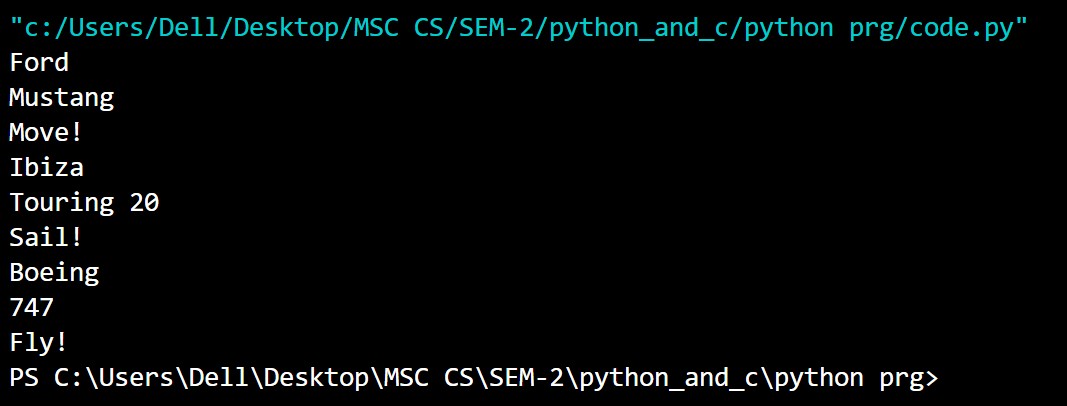
for x in (car1, boat1, plane1):

  print(x.brand)

  print(x.model)

  x.move()

Output:



18.WAP to Demonstrate operator overloading

class complex\_1:

    def \_\_init\_\_(self, X, Y):

        self.X = X

        self.Y = Y

 # Now, we will add the two objects

    def \_\_add\_\_(self, U):

        return self.X + U.X, self.Y + U.Y

Object\_1 = complex\_1(23, 12)

Object\_2 = complex\_1(21, 22)

Object\_3 = Object\_1 + Object\_2

print (Object\_3)

Output:

(44, 34)

19.WAP to swap value using two variable

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

b=int(input("Enter any number for b:"))

print("A: ",a)

print("B: ",b)

a=a+b

b=a-b

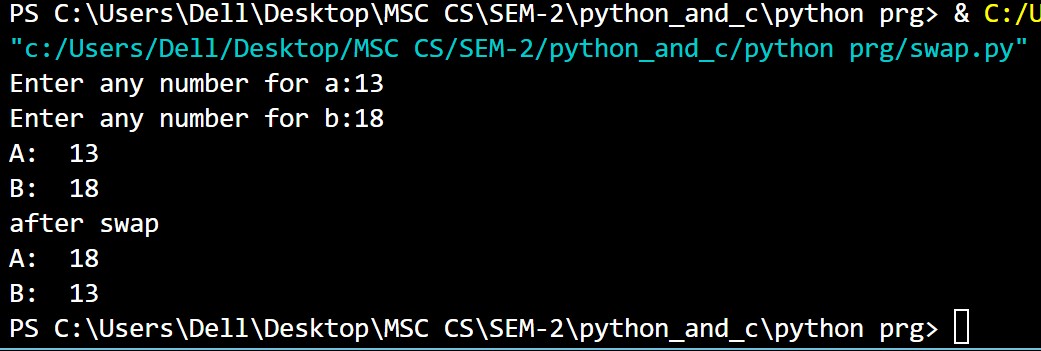
a=a-b

print("after swap")

print("A: ",a)

print("B: ",b)

Output:



20. Write a program to display the Fibonacci sequence(using recursion)

def recur\_fibo(n):

if n <= 1:

return n

else:

return(recur\_fibo(n-1) + recur\_fibo(n-2))

nterms = 12

# check if the number of terms is valid

if nterms <= 0:

print("Plese enter a positive integer")

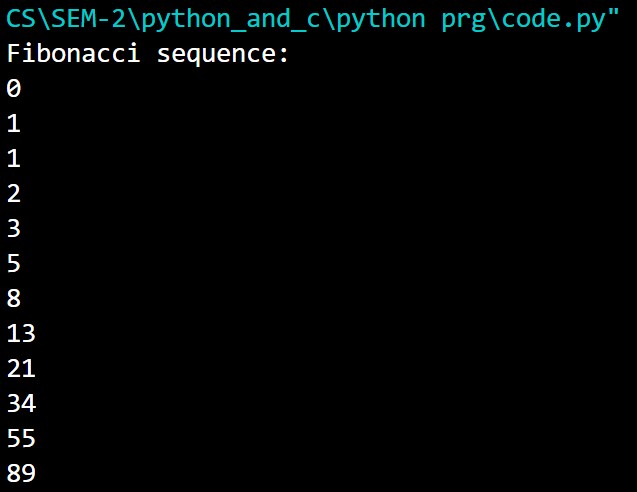
else:

print("Fibonacci sequence:")

for i in range(nterms):

print(recur\_fibo(i))

Output:



21.WAP to Remove extra chater from the string

# define punctuation

punctuations = '''!()-[]{};:'"\,<>./?@#$%^&\*\_~'''

my\_str = "Hello!!!, he said ---and went."

# To take input from the user

# my\_str = input("Enter a string: ")

# remove punctuation from the string

no\_punct = ""

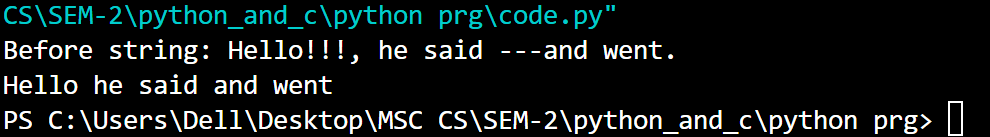
for char in my\_str:

if char not in punctuations:

no\_punct = no\_punct + char

# display the unpunctuated string

print(no\_punct)

Output: