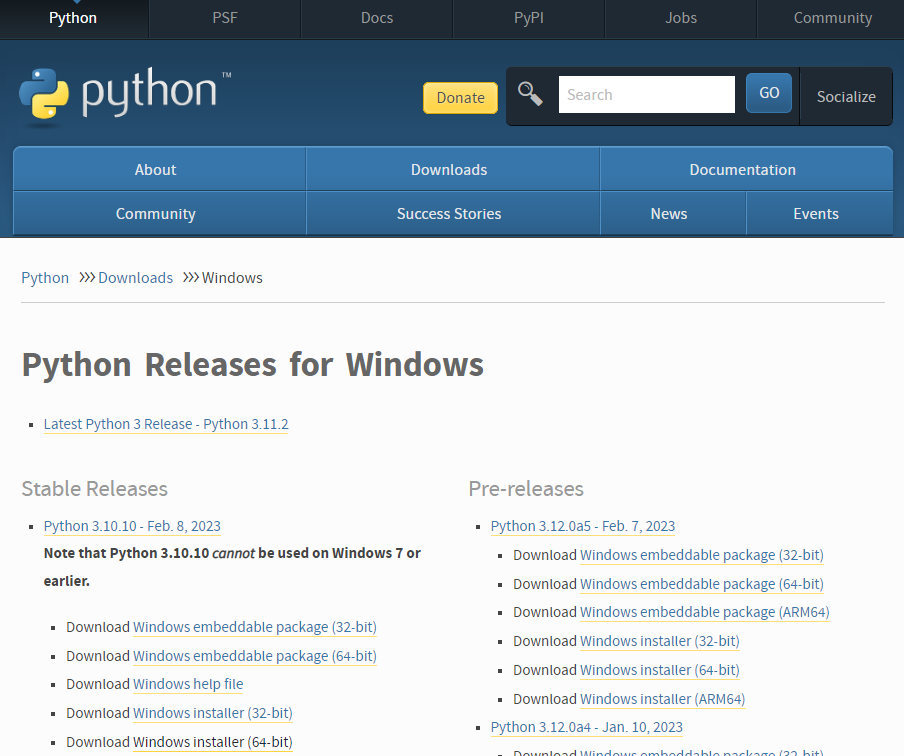
**Week -1: Basics**

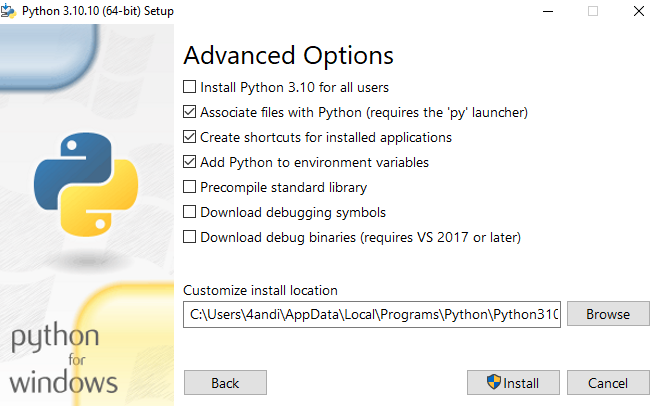
1)Python Installation:

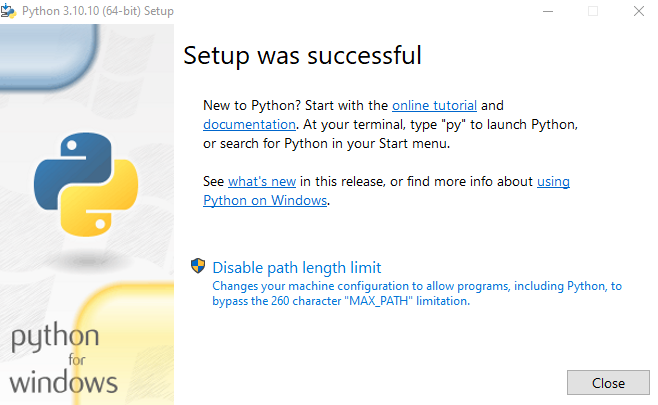
Windows:

1. Download Python: Go to the official python website at

<https://www.python.org/downloads/> and click on the “Download Python” button. Choose the latest version for windows.



1. Run Installer: Locate the downloaded installer file (usually named something like python-3.10.10.exe) and double-click it to run the installer. 
2. Customize Installer (Optional): On the first screen of the installer, make sure to check the box that says “Add Python 3.x to PATH.” This will make it easier to run Python from the Command Prompt. You can also customize the installation directory if needed. 
3. Install Python: Click the “Install now” button to start the installation process. The installer will copy the necessary files to your system.
4. Installation complete: Once the installation is complete, you’ll see a screen that says “Setup was successful.” You can now close the installer.



**2. Start a Python interpreter and use it as a Calculator.**

#Calculator

a=int(input("enter a number"))

b=int(input("enter a number"))

print("add:",a+b)

print("sub:",a-b)

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

print("division:",a/b)

**Output:**

enter a number 4

enter a number 5

add: 9

sub: -1

multiplication: 20

division: 0.8

**3. Write a program to purposefully raise Indentation Error and correct it.**

#Indentation Error

a=5

b=2

print("add:",a+b)

#correct program

a=5

b=2

Print(“add:”,a+b)

**Output:**

7

**4. i) Write a program to calculate compound interest when principal, rate and number**

**of periods are given.**

**Program:**

#compound interest

#p=principle,r=rate,t=time period,CI=compound interest,A=amount after time period

p=1000

r=2

t=3

A=p\*((1+r/100)\*\*t)

CI=A-p

print("amount after time period",A)

print("compound intrest:",CI)

**output:**

amount after time period 1061.208

compound intrest: 61.208000000000084

**ii) Given coordinates (x1, y1), (x2, y2) find the distance between two points**

**Program:**

x1=int(input("enter the x1 value"))

x2=int(input("enter the x2 value"))

y1=int(input("enter the y1 value"))

y2=int(input("enter the y2 value"))

d=((x1-x2)\*\*2+(y1-y2)\*\*2)\*\*(1/2)

print("distance between two given points is ",d)

**output:**

enter the x1 value0

enter the x2 value2

enter the y1 value0

enter the y2 value2

distance between two given points is 2.8284271247461903

**5. Read name, address, email and phone number of a person through keyboard and**

**print the details.**

**Program:**

name=input("enter your name ")

email\_id=input("enter your email-id :")

phone\_no=int(input("enter your phone number :"))

address=input("enter your address :")

print("Name:",name)

print("Email-id:",email\_id)

print("Phone-no:",phone\_no)

print("Address:",address)

**output:**

enter your name :Divya

enter your email-id :divya@gmail.com

enter your phone number :987654321

enter your address: Hyderabad,Telangana

Name: Divya

Email-id: divya@gmail.com

Phone-no: 987654321

Address: Hyderabad,Telangana

**Week - 2: Operations**

**1. Print the below triangle using for loop.**

**5**

**4 4**

**3 3 3**

**2 2 2 2**

**1 1 1 1 1**

**Program:**

num=5

for i in range(0,5):

for j in range(0,i+1):

print(num,end=" ")

print("\n")

num=num-1

**output:**

5

4 4

3 3 3

2 2 2 2

1 1 1 1 1

**2. Write a program to check whether the given input is digit or lowercase character**

**or uppercase character or a special character (use 'if-else-if' ladder)**

**Program**:  
a=input("enter any element")

if a.islower():

print("given input is of lower case characters")

elif a.isupper():

print("given input is of upper case characters")

elif a.isalnum():

print("Given input are numbers")

else:

print("special characters")

**output:**

enter any element%^

special characters

**3. Python Program to Print the Fibonacci sequence using while loop**

**Program:**

#fibonacci series

a=int(input("enter the number of elements in fibonacci series"))

l=0

i=0

j=1

k=i+j

for l in range(0,a):

print(i)

i=j

j=k

k=i+j

**output:**

enter the number of elements in fibonacci series 7

0

1

1

2

3

5

8

**4. Python program to print all prime numbers in a given interval (use break)**

**Program:**  
count=0

a=int(input("enter the lowest range"))

b=int(input("enter the upper range"))

for n in range(a,b):

if(n>0):

for j in range(1,n+1):

if(n%j==0):

count=count+1

if(count==2):

print(n," is a prime number")

count=0

else:

break

output:

enter the lowest range1

enter the upper range30

2 is a prime number

3 is a prime number

5 is a prime number

7 is a prime number

11 is a prime number

13 is a prime number

17 is a prime number

19 is a prime number

23 is a prime number

1. is a prime number

**5.Write a program to compute LCM of two numbers by taking input from the user**

**Program:**import math as m

num1=int(input(“enter a number”))

num2=int(input(“enter another number”))

HCF=m.gcd(num1,num2)

LCM=int((num1\*num2)/(HCF))

print(“LCM of given numbers :”,LCM)

**output:**

enter a number3

enter another number4

LCM of given numbers : 12

**6.Write a program add.py that takes 2 numbers as command line arguments and**

**prints its sum.**

**Program:**

import sys as s

sum=float(s.argv[1])+float(s.argv[2])

print("sum is:",sum)

**output:**

C:\Users\abhil\OneDrive\Documents\cls.py>py add.py 1 2

Sum is 3

**Week - 3: Lists & Tuples**

1. **i) Write a program to convert a list and tuple into arrays.**

**Program:**

from array import array

list=[1,2,3]

list\_array=array("i",list)

print(“list\_array:”,list\_array)

tuple=(4,9,0)

tuple\_array=array("i",tuple)

print(“tuple\_array:”,tuple\_array)

**output:**

list\_array: array(‘i’,[1,2,3])

tuple\_array:array(‘i’,[2,9,0])

**ii) Write a program to find common values between two arrays.**

**Program:**

from array import \*

array1=array('i',[7,8,9,8])

array2=array('i',[23,33,11,8])

common\_values=[]

for i in array1:

for j in array2:

if i==j:

common\_values.append(i)

common\_values=set(common\_values)

print(common\_values)

**output:**

8

**2. Write a function called gcd that takes parameters a and b and returns their greatest common divisor.**

**Program:**def gcd(a, b):

while b:

a, b = b, a % b

return a

num1 = 48

num2 = 18

result = gcd(num1, num2)

print("GCD of", num1, "and", num2, "is:", result)

**output:**

GCD of 48 and 18 is: 6

**3. Write a function called palindrome that takes a string argument and returns True if it is a palindrome and False otherwise. Remember that you can use the built-in function len to check the length of a string.**

**Program:**

def palindrome(string):

len\_string=len(string)

for i in range(0,int(len\_string/2)):

if(string[i]==string[len\_string-1]):

len\_string=len\_string-1

return True

else:

return False

a=input("enter a string ")

c=palindrome(a)

print(c)

**output:**

enter a string madam

True

**4. Find mean, median, mode for the given set of numbers in a list.**

**Program:**

import statistics as s

list=[]

n=int(input("enter the number of elements"))

for i in range(0,n):

print("enter the element")

elements=int(input())

list.append(elements)

print(s.mean(list),"=mean")

print(s.mode(list),"=mode")

print(s.median(list),"=median")

**output:**

enter the number of elements5

enter the element

3

enter the element

5

enter the element

3

enter the element

6

enter the element

2

3.8 =mean

3 =mode

3 =median

**5. Write a Python program to create a tuple.**

**Program:**

tuple1 = (1, 2, 3, 4, 5)

# Creating an empty tuple

tuple2 = ()

# Creating a tuple with a single element

tuple3 = (10,) # Note the comma after the single element

tuple4 = tuple([6, 7, 8, 9, 10])

tuple5 = ("apple", 3.14, True)

print("Tuple 1:", tuple1)

print("Tuple 2:", tuple2)

print("Tuple 3:", tuple3)

print("Tuple 4:", tuple4)

print("Tuple 5:", tuple5)

output:

Tuple 1: (1, 2, 3, 4, 5)

Tuple 2: ()

Tuple 3: (10,)

Tuple 4: (6, 7, 8, 9, 10)

Tuple 5: ('apple', 3.14, True)

**6.Write a Python program to create a tuple with different data types.**

**Program:**  
tuple1=("Apple",True,30)

print("Tuple : ",tuple1)

**output:**

Tuple : ('Apple', True, 30)

**7.Write a Python program to check whether an element exists within a tuple.**

**Program:**a=input("enter the element you want to search in the tuple: ")

tuple1=("Dog","cat",True,50,20)

len\_tuple1=len(tuple1)

for i in range(0,len\_tuple1):

if(a==tuple1[i]):

print("Element exist within the tuple")

**output:**

enter the element you want to search in the tuple: Dog

Element exist within the tuple

**Week - 4: Sets, Dictionaries and Strings**

**1. Write a function called is\_sorted that takes a list as a parameter and returns True if the list is sorted in ascending order and False otherwise.**

**Program:**  
def is\_sorted(input\_list):

for i in range(len(input\_list) - 1):

if input\_list[i] > input\_list[i + 1]:

return False

return True

list1 = [1, 2, 3, 4, 5]

list2 = [5, 3, 8, 2, 10]

print(is\_sorted(list1))

print(is\_sorted(list2))

**output:**  
True

False

**2. Write a function called has\_duplicates that takes a list and returns True if there is any element that appears more than once. It should not modify the original list.**

**Program:**  
def as\_duplicate(list1):

a=len(list1)

for i in range(0,a+1):

for j in range(i+1,a):

if(list1[i]==list1[j]):

return True

return False

list1=[1,2,6,4,5,1]

c=as\_duplicate(list1)

print(c)

**output:**

True

**i). Write a function called remove\_duplicates that takes a list and returns a new list with only the unique elements from the original. Hint: they don’t have to be in the same order.**

**Program:**

def remove\_duplicates(input\_list):

unique\_list = list(set(input\_list))

return unique\_list

original\_list = [2, 3, 2, 5, 6, 5, 8, 2, 9]

new\_list = remove\_duplicates(original\_list)

print(new\_list)

**output:**

[2, 3, 5, 6, 8, 9]

**ii). The wordlist I provided, words.txt, doesn’t contain single letter words. So you might want to add “I”, “a”, and the empty string.**

**Program:**

**Output:**

**iii). Write a python code to read dictionary values from the user. Construct a function to invert its content. i.e., keys should be values and values should be keys.**

**Program:**

original\_dict = {'a': 1, 'b': 2, 'c': 3}

interchanged\_dict = {value: key for key, value in original\_dict.items()}

print(interchanged\_dict)

**output:**

{1: 'a', 2: 'b', 3: 'c'}

**3. i) Add a comma between the characters. If the given word is 'Apple', it should**

**become 'A,p,p,l,e'.**

**program:**x="apple"

y=','.join(x)

print(y)

**output:**

A,p,p,l,e

**ii) Remove the given word in all the places in a string?**

**Program:**  
x="This is just a test."

y=x.replace("is","")

print(y)

**output:**

Th just a test.

**iii) Write a function that takes a sentence as an input parameter and replaces the first letter of every word with the corresponding upper case letter and the rest of the letters in the word by corresponding letters in lower case without using a built-in function?**

**Program:**

def capitalizing(sentence):

words = sentence.split()

cap\_words=[]

for i in words:

cap\_words.append(i.capitalize())

cap\_sentence = ' '.join(cap\_words)

print(cap\_sentence)

input\_sentence = input("Enter a sentence: ")

capitalizing(input\_sentence)

**Output:**

Enter a sentence: hello world hiii!

Hello World Hii!

**4. Writes a recursive function that generates all binary strings of n-bit length**

**Program:**def generate\_binary\_strings(n, prefix=""):

if n == 0:

print(prefix)

else:

generate\_binary\_strings(n - 1, prefix + "0")

generate\_binary\_strings(n - 1, prefix + "1")

n\_bits = 4

generate\_binary\_strings(n\_bits)

**output:**

0000

0001

0010

0011

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

**5. Write a Python program to implement all set operations**

**Program:**

set1={1,2,3,4,5}

set2={3,4,5,6,7}

union\_result=set1.union(set2)

union\_intersection=set1.intersection(set2)

union\_difference=set1.difference(set2)

symmetric\_difference\_result = set1.symmetric\_difference(set2)

print("Union: ",union\_result)

print("Intersection: ",union\_intersection)

print("Difference : ",union\_difference)

print("symmetric difference : ",symmetric\_difference\_result)

**output:**

Union: {1, 2, 3, 4, 5, 6, 7}

Intersection: {3, 4, 5}

Difference : {1, 2}

symmetric difference : {1, 2, 6, 7}

**6. Write a program to check whether a string is palindrome or not.**

**Program:**

def palindrome(x):

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

if(x[i]==x[len(x)-i-1]):

return True

else:

return False

Input=input("enter a string ")

c=palindrome(Input)

print(c)

**output:**

enter a string madam

True

**Week - 5: Functions and Multi-D**

1. **i) Write a python program that defines a matrix and prints**

**program:**matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

for row in matrix:

print(row)

**output:**

[1, 2, 3]

[4, 5, 6]

[7, 8, 9]

**ii) Write a python program to perform addition of two square matrices**

**program:**  
def add\_matrices(matrix1, matrix2):

rows = len(matrix1)

columns = len(matrix1[0])

result = []

for i in range(rows):

row = []

for j in range(columns):

row.append(matrix1[i][j] + matrix2[i][j])

result.append(row)

return result

# Define two square matrices

matrix1 = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

matrix2 = [

[9, 8, 7],

[6, 5, 4],

[3, 2, 1]

]

# Perform matrix addition

result\_matrix = add\_matrices(matrix1, matrix2)

# Print the result

for row in result\_matrix:

print(row)

**Output:**

[10, 10, 10]

[10, 10, 10]

[10, 10, 10]

**iii) Write a python program to perform multiplication of two square matrices.**

**Program:**

def multiply\_matrices(matrix1, matrix2):

rows1 = len(matrix1)

columns1 = len(matrix1[0])

rows2 = len(matrix2)

columns2 = len(matrix2[0])

if columns1 != rows2:

raise ValueError("Matrices cannot be multiplied due to incompatible dimensions.")

result = []

for i in range(rows1):

row = []

for j in range(columns2):

value = 0

for k in range(columns1):

value += matrix1[i][k] \* matrix2[k][j]

row.append(value)

result.append(row)

return result

# Define two square matrices

matrix1 = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

matrix2 = [

[9, 8, 7],

[6, 5, 4],

[3, 2, 1]

]

# Perform matrix multiplication

result\_matrix = multiply\_matrices(matrix1, matrix2)

# Print the result

for row in result\_matrix:

print(row)

output:  
[30, 24, 18]

[84, 69, 54]

[138, 114, 90]

1. Simple Calculator program by making use of functions

Program:  
def add(x, y):

return x + y

def subtract(x, y):

return x - y

def multiply(x, y):

return x \* y

def divide(x, y):

if y == 0:

return "Cannot divide by zero"

return x / y

print("Select operation:")

print("1. Add")

print("2. Subtract")

print("3. Multiply")

print("4. Divide")

choice = input("Enter choice (1/2/3/4): ")

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

if choice == '1':

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

elif choice == '2':

print(num1, "-", num2, "=", subtract(num1, num2))

elif choice == '3':

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

elif choice == '4':

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

else:

print("Invalid Input")

output:

Select operation:

1. Add

2. Subtract

3. Multiply

4. Divide

Enter choice (1/2/3/4): 1

Enter first number: 1

Enter second number: 2

* 1. + 2.0 = 3.0

3.Find the factorial of a number using recursion

Program:

def fact(a):

if(a==0 or a==1):

return 1

else:

return a\*fact(a-1)

num=int(input("enter a number "))

factnum=fact(num)

print(factnum)

output:

enter a number 5

120

**4. Write a function cumulative\_ product to compute cumulative product of a list of**

**numbers.**

**Program:**

def cumulative\_product(list1):

a=len(list1)

for i in range(0,a):

if list1[i]==0 or list1[i]==1 :

a=1

else:

a= a\*list1[i]

return a

list1=[1,2,3,4,5]

c=cumulative\_product(list1)

print(c)

**output:**

120

**5. Write a function reverse to print the given list in the reverse order.**

**Program:**

def reverse\_list(list1):

list2=[]

for i in range(len(list1) -1,-1,-1):

list2.append(list1[i])

return list2

list1=[1,2,3,4,5]

list2=reverse\_list(list1)

print(list2)

**output:**

[5, 4, 3, 2, 1]

**Week-6: Exceptions in Python**

1. **Write a program that detects an Exception**

**Program:**

try:

num1=int(input("enter an number"))

num2=int(input("enter other number"))

result=num1/num2

print("result:",result)

except ZeroDivisionError:

print("Division by zero is not accepted")

except ValueError:

print("enter only integers")

except Exception as e:

print("an error occured",str(e))

output:

enter an number1

enter other number$

enter only integers

**output:**

enter an number1

enter other number0

Division by zero is not accepted

1. **Write a program that raise an Exception ( divide by zero error,voter’s age validity**

**Program:**  
def voter\_age(age):

try:

num=int(age)

if(num>18):

print("valid age")

elif(num<18):

print("not a valid age")

except Exception as e:

print("enter only integers:",str(e))

num=input("enter the age")

voter\_age(num)

**output:**

enter the age&

enter only integers: invalid literal for int() with base 10: '&'

1. **Write a program that raise an Exception as string(), student mark range validation**

**Program:**

def students\_mark(marks):

try:

num=int(marks)

if(num>0):

print("valid number")

elif(num<0 or num==0):

print("not valid")

except Exception as e:

print("enter only integers :",str(e))

num=input("enter students marks")

students\_mark(num)

**output:**

enter students marks#$

enter only integers : invalid literal for int() with base 10: '#$'

1. **Use the structure of exception handling all general purpose exceptions.**

**Program:**try:

num1=int(input("enter a number"))

num2=int(input("enter other number"))

result=num1/num2

print("result : ",result)

except Exception as e:

print("error occured :",str(e))

else:

print("division successful")

finally:

print("program completed successfully")

**output:**

enter a number2

enter other number#

error occured : invalid literal for int() with base 10: '#'

program completed successfully

**5. Write a python code to read a phone number and email-id from the user and**

**validate it for correctness**

**program:**  
import re

def Validate\_phone(phone):

pattern=r'^\d{10}$'

if re.match(pattern,phone):

return True

else:

return False

try:

phone=input("enter phone number")

if Validate\_phone(phone):

print("phone number is valid")

else:

print("phone number is not valid")

except Exception as e:

print("exception : ",str(e))

**output:**

enter phone number!@#$%^&\*()

phone number is not valid

program:

import re

def email\_id(id):

pattern=r'^[a-zA-Z0-9.\_+\*-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

if re.match(pattern,id):

return True

else:

return False

try:

id=input("enter your email-id")

if(email\_id(id)):

print("id is correct")

else:

print("id is incorrect")

except Exception as e:

print("Exception : ",str(e))

**output:**

enter your email-iddivya@gmail.com

id is correct

**Week-7: Modules and Inheritance**

**1. How do you make a module? Give an example of construction of a module using different geometrical shapes and operations on them as its functions.**

# geometry.py

import math

def square\_area(side):

return side \* side

def square\_perimeter(side):

return 4 \* side

def circle\_area(radius):

return math.pi \* radius\*\*2

def circle\_circumference(radius):

return 2 \* math.pi \* radius

def triangle\_area(base, height):

return 0.5 \* base \* height

def triangle\_perimeter(side1, side2, side3):

return side1 + side2 + side3

# main.py

import geometry

side\_length = 5

radius = 3

base = 4

height = 6

side1 = 3

side2 = 4

side3 = 5

print("Square area:", geometry.square\_area(side\_length))

print("Square perimeter:", geometry.square\_perimeter(side\_length))

print("Circle area:", geometry.circle\_area(radius))

print("Circle circumference:", geometry.circle\_circumference(radius))

print("Triangle area:", geometry.triangle\_area(base, height))

print("Triangle perimeter:", geometry.triangle\_perimeter(side1, side2, side3))

Square area: 25

Square perimeter: 20

Circle area: 28.274333882308138

Circle circumference: 18.84955592153876

Triangle area: 12.0

Triangle perimeter: 12

**2. a. Write a function called draw\_rectangle that takes a Canvas and a Rectangle as arguments and draws a representation of the Rectangle on the Canvas.**

**Program:**

class Canvas:

def \_init\_(self, width, height):

self.width = width

self.height = height

self.grid = [[' ' for \_ in range(width)] for \_ in range(height)]

def draw(self):

for row in self.grid:

print(' '.join(row))

class Rectangle:

def \_init\_(self, x, y, width, height):

self.x = x

self.y = y

self.width = width

self.height = height

def draw\_rectangle(canvas, rectangle):

for row in range(rectangle.y, rectangle.y + rectangle.height):

for col in range(rectangle.x, rectangle.x + rectangle.width):

if row == rectangle.y or row == rectangle.y + rectangle.height - 1 or col == rectangle.x or col == rectangle.x + rectangle.width - 1:

canvas.grid[row][col] = '#'

canvas = Canvas(10, 6)

rectangle = Rectangle(2, 1, 5, 3)

draw\_rectangle(canvas, rectangle)

canvas.draw()

**Output:**

**# # # # #**

**# #**

**# # # # #**

**b. Add an attribute named color to your Rectangle objects and modify draw\_rectangle so that it uses the color attribute as the fill color.**

import matplotlib.pyplot as plt

import matplotlib.patches as patches

class Canvas:

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

self.fig, self.ax = plt.subplots()

def draw\_rectangle(self, rectangle):

x = rectangle.x

y = rectangle.y

width = rectangle.width

height = rectangle.height

color = rectangle.color

rect = patches.Rectangle((x, y), width, height, color=color)

self.ax.add\_patch(rect)

def show(self):

plt.axis('equal')

plt.show()

class Rectangle:

def \_\_init\_\_(self, x, y, width, height, color):

self.x = x

self.y = y

self.width = width

self.height = height

self.color = color

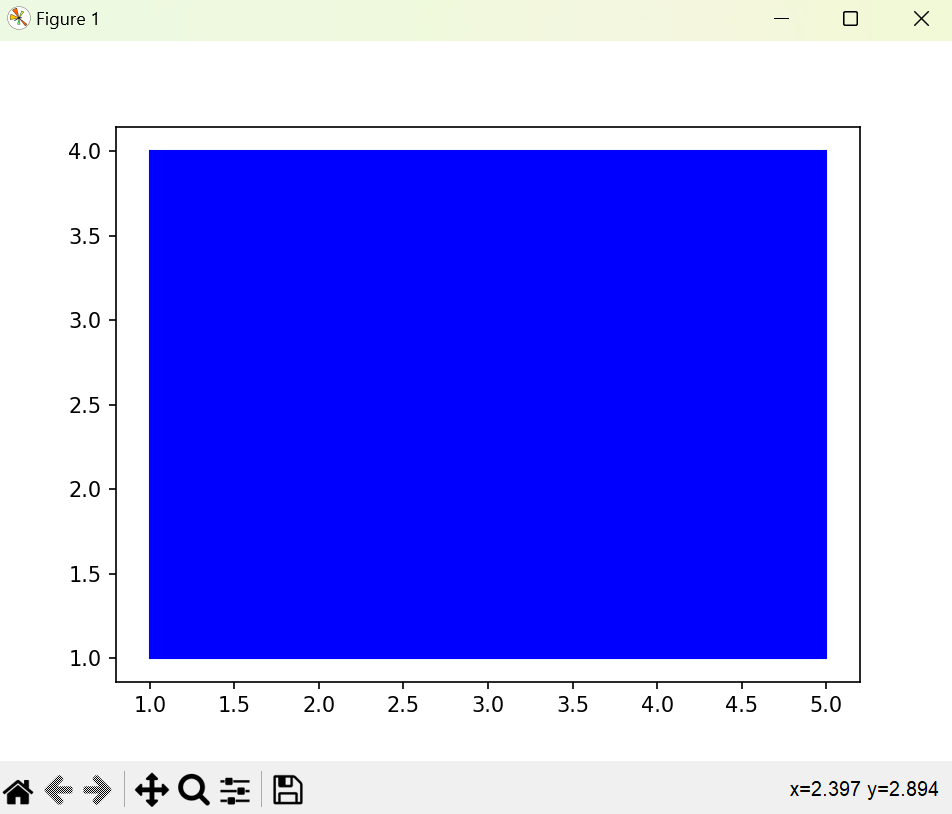
# Example usage:

canvas = Canvas()

rectangle1 = Rectangle(1, 1, 4, 3, "blue")

canvas.draw\_rectangle(rectangle1)

canvas.show()

****

**c. Write a function called draw\_point that takes a Canvas and a Point as arguments and draws a representation of the Point on the Canvas. \**

import matplotlib.pyplot as plt

import matplotlib.patches as patches

class Canvas:

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

self.fig, self.ax = plt.subplots()

def draw\_point(self, point):

x = point.x

y = point.y

plt.plot(x, y, 'ro')

def show(self):

plt.axis('equal')

plt.show()

class Point:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

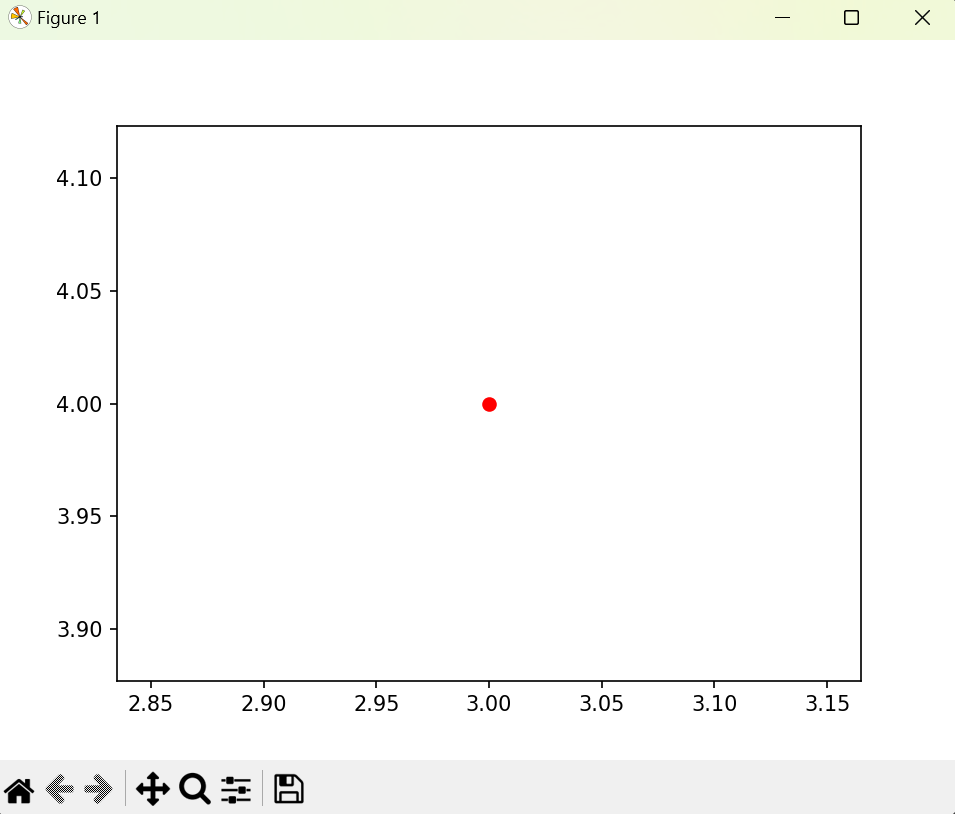
# Example usage:

canvas = Canvas()

point1 = Point(3, 4)

canvas.draw\_point(point1)

canvas.show()

****

**d. Define a new class called Circle with appropriate attributes and instantiate a few Circle objects. Write a function called draw\_circle that draws circles on the canvas.**

import matplotlib.pyplot as plt

import matplotlib.patches as patches

class Canvas:

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

self.fig, self.ax = plt.subplots()

def draw\_circle(self, circle):

x = circle.x

y = circle.y

radius = circle.radius

color = circle.color

circle = patches.Circle((x, y), radius, color=color)

self.ax.add\_patch(circle)

def show(self):

plt.axis('equal')

plt.show()

class Circle:

def \_\_init\_\_(self, x, y, radius, color):

self.x = x

self.y = y

self.radius = radius

self.color = color

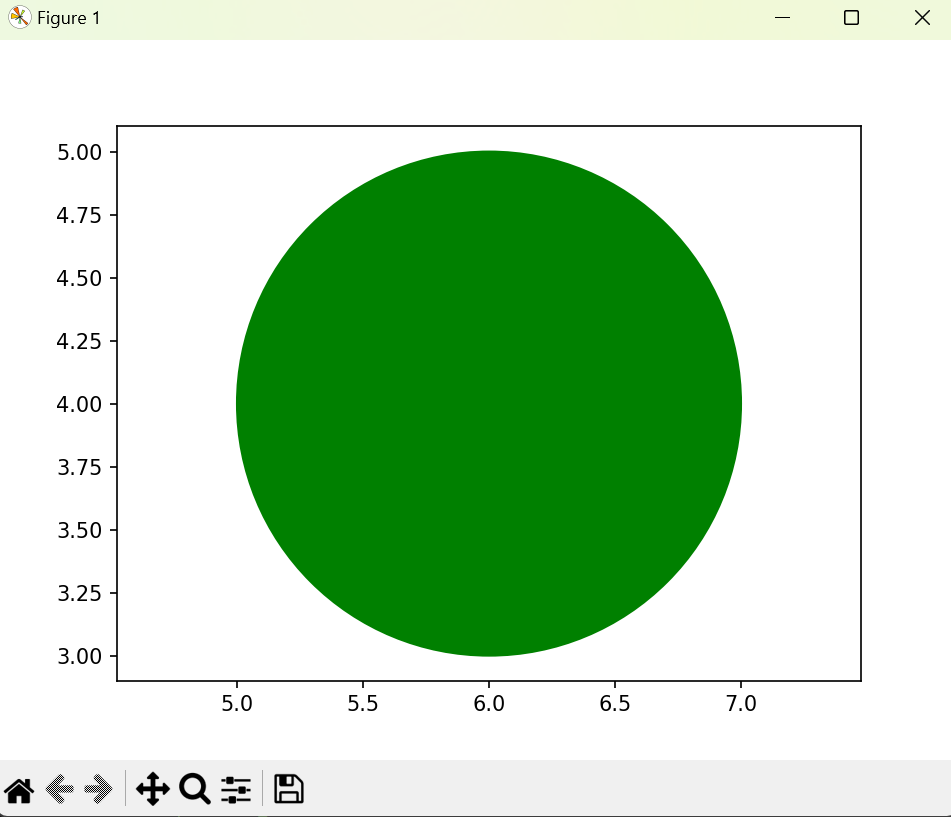
# Example usage:

canvas = Canvas()

circle1 = Circle(6, 4, 1, "green")

canvas.draw\_circle(circle1)

canvas.show()



**3. Write a Python program to demonstrate the usage of Method Resolution Order (MRO) in multiple levels of Inheritance.**

**Program:**

class A:

def show(self):

print("A class method")

class B(A):

def show(self):

print("B class method")

class C(A):

def show(self):

print("C class method")

class D(B, C):

def show(self):

print("D class method")

# Instantiate the D class object and call the show method

d\_obj = D()

d\_obj.show()

# Print the Method Resolution Order for class D

print(D.mro())

**Output:**

D class method

[<class '\_\_main\_\_.D'>, <class '\_\_main\_\_.B'>, <class '\_\_main\_\_.C'>, <class '\_\_main\_\_.A'>, <class 'object'>]

**Week- 8: Files**

**1. Write a Python code to merge two given file contents into a third file.**

**Program:**

file1=open("t1.txt","w",encoding="utf-8")

file2=open("t2.txt","w")

file1.write("hello")

file2.write("world!")

file1.close()

file2.close()

file1=open("t1.txt","r")

data1=file1.read()

file2=open("t2.txt","r")

data2=file2.read()

file1.close()

file2.close()

file3=open("t3.txt","w")

file3.write(data1+" "+data2)

file3.close()

file3=open("t3.txt","r")

data=file3.read()

file3.close()

**Output:**

t1.txt:

hello

t2.txt:

world!

t3.txt:

hello world!

**2. Write a Python code to open a given file and construct a function to check for given words present in it and display on found.**

**Program:**

file=open("t3.txt","r")

data=file.read() #t3.txt: hello world!

file.close()

def check\_word(data,word):

words=data.split(" ")

for i in words:

if word==i:

print("word found")

break

else:

print("not found")

word=input("Enter word to be searched:")

check\_word(data,word)

**Output:**

Enter word to be searched:hello

Word found

**3. Write a Python code to Read text from a text file, find the word with most number of occurrences.**

**Program:**

count=0

max\_count=0

words=[]

file=open("text.txt","r") # apple banana apple mango banana mango orange mango

for line in file:

string=line.lower().replace(',','').replace('.','').split(" ")

for s in string:

words.append(s)

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

count=1

for j in range (i+1,len(words)):

if words[i]==words[j]:

count=count+1

if count>max\_count:

max\_count=count

word=words[i]

print("Most repeated word:",word)

**output**

Most repeated word:mango

**4. Write a function that reads a file file1 and displays the number of words, number of vowels, blank spaces, lower case letters and uppercase letters.**

**Program:**

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

data=file1.read() #hello world

file1.close()

def diaplay():

vowel= 0

space=0

con=0

lower=0

upper=0

vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']

words=data.split()

print("Total words:",len(words))

for letter in data:

if letter in vowels:

vowel = vowel+1

elif letter==" ":

space=space+1

else:

con=con+1

if letter.islower():

lower=lower+1

if letter.isupper():

upper=upper+1

print("Total Vowels:")

print(vowel)

print("Total con:")

print(con)

print("Total lower:")

print(lower)

print("Total upper:")

print(upper)

print("Total space:")

print(space)

diaplay()

**optput:**

Total words: 2

Total Vowels:

3

Total con:

7

Total lower:

10

Total upper:

0

Total space:

1

**5. Write a program to print each line of a file in reverse order.**

**Program:**

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

r=file.read()

word=r.split()

print(word)

l1 =[]

for i in word:

l1.append(i[::-1])

print (l1)

**Output:**

['hello', 'world']

['olleh', 'dlrow']

**Week - 9: Exploration of NumPy Package**

**1. Import numpy, and explore their functionalities.**

**Import NumPy:**

Impot numpy as np

**Functionalities:**

**Creating Arrays:**

**Program:**

import numpy as np

arr1 = np.array([10,20,30,40,50])

print(arr1)

print(type(arr1))

**Output:**

[10 20 30 40 50]

<class ‘numpy.ndarray’>

**1-D Arrays**

**Program:**

import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

print(arr[0]

**Output:**

['1' '2' '3' '4' '5']

1

**2-D Arrays**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr)

print(arr[1][1])

print(arr[1])

**Output:**

[[1 2 3]

[4 5 6]]

5

[4 5 6]

**Dimension**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr.ndim)

**Output:**

2

**Size of each element (in bytes)**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr.itemsize)

**Output:**

4

**Datatype**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr.dtype)

**Output:**

Int32

**Size and Shape**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr.size)

print(arr.shape)

**Output:**

6

(2, 3)

**Reshape**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

a=arr.reshape(3,2)

print(a)

**Output:**

[[1 2]

[3 4]

[5 6]]

**Slicing**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr[0:,2])

**Output:**

[3 6]

**Min/Max/Sum**

**Program:**

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr.min())

print(arr.max())

print(arr.sum())

**Output:**

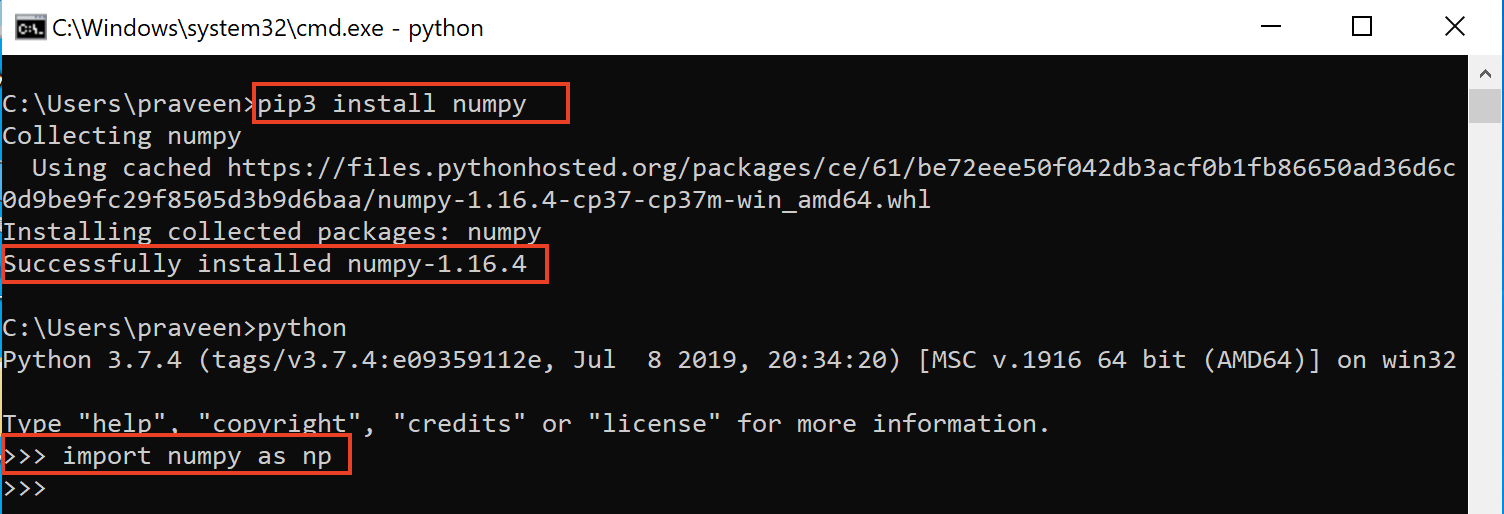
1

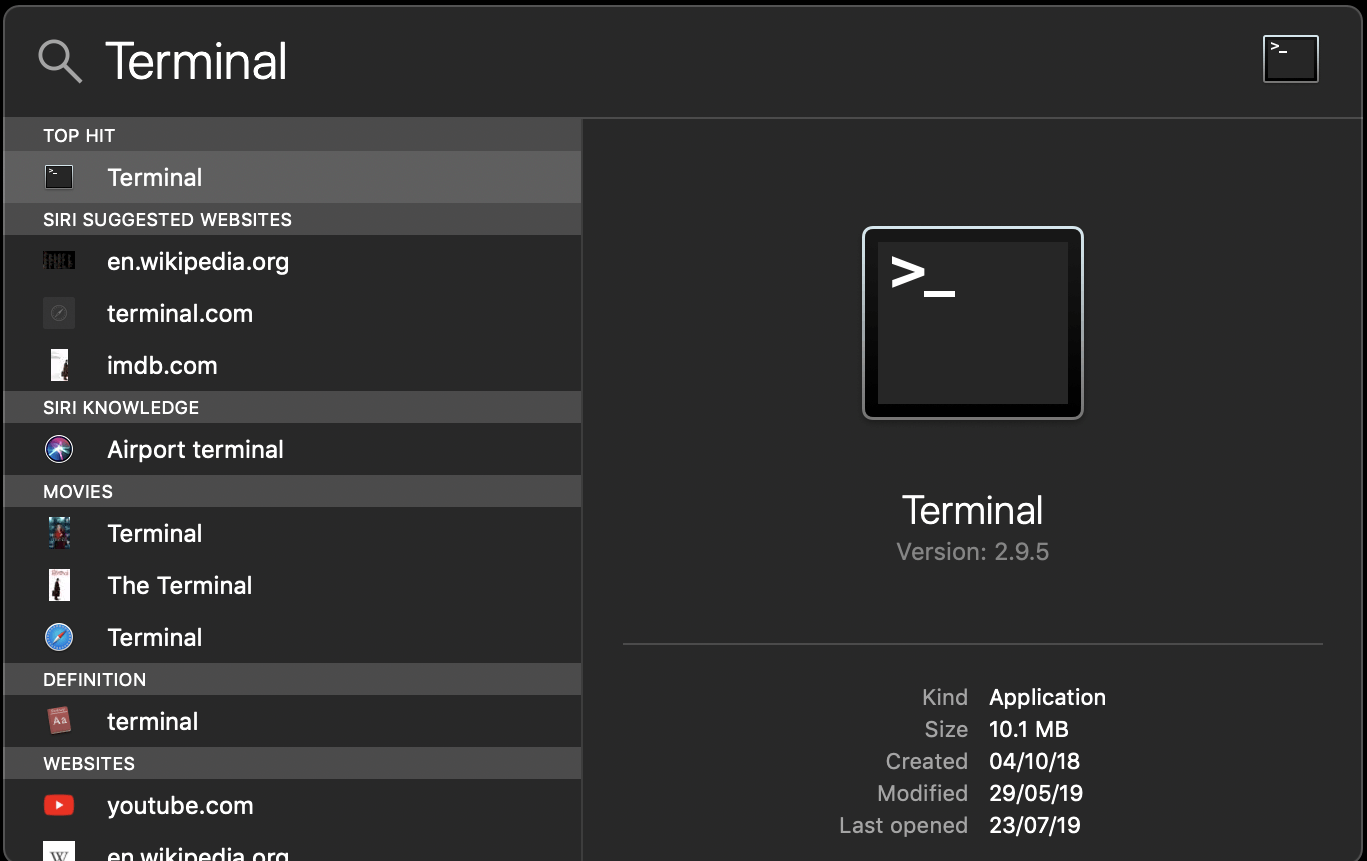
6

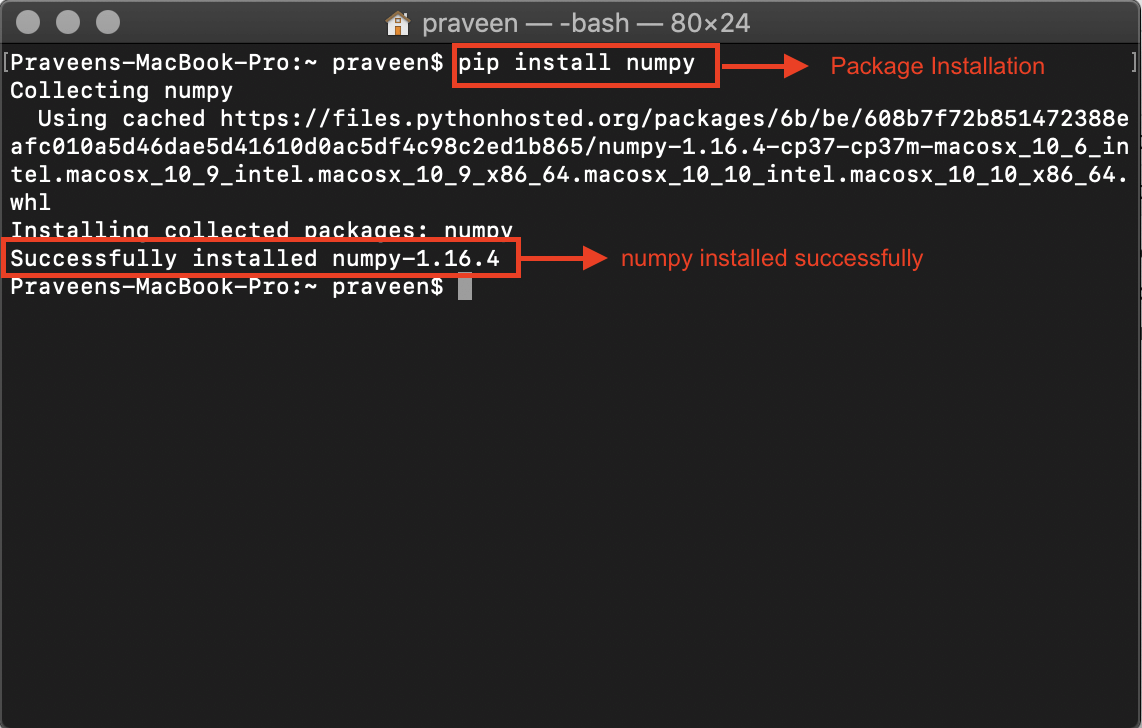
21

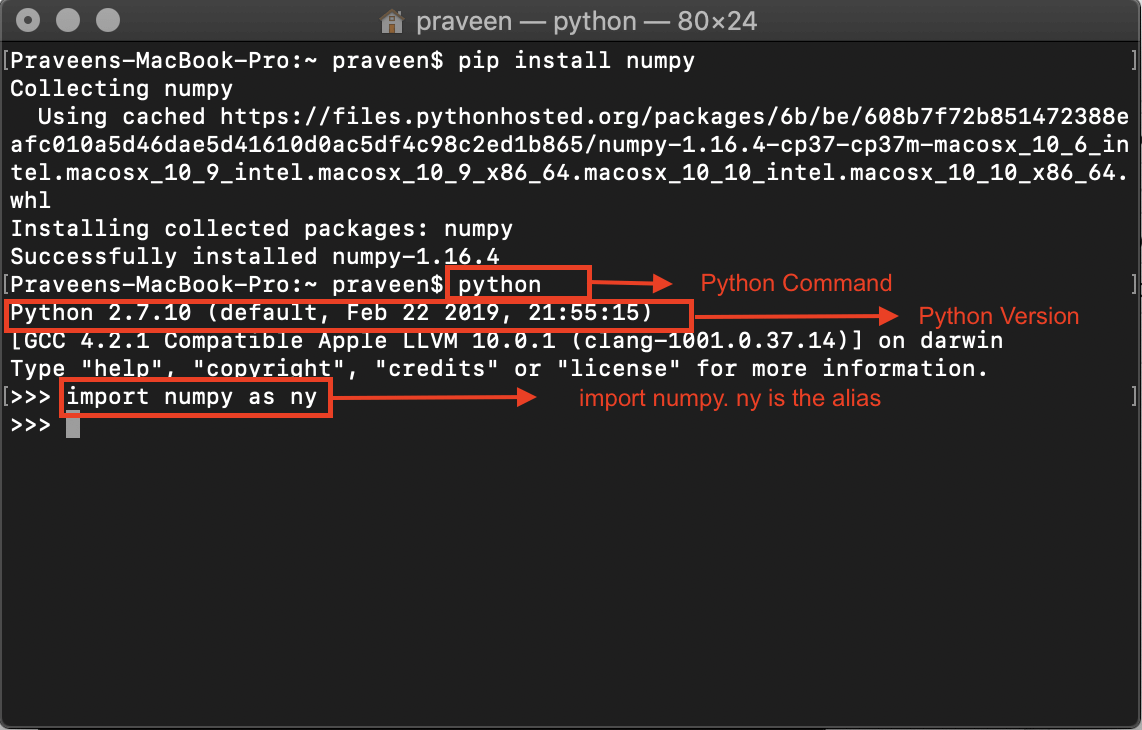
**2. a) Install NumPy package with pip and explore it.**

**Python is not installed by default in windows operating system. You can download the required version of python from python.org. Once python is installed successfully, open command prompt and use pip to install numpy.**



****

****

****

**3. Write a program for slicing arrays using numpy .**

**Program:**

import numpy as np arr = np.

array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])

print(arr[4:])

print(arr[:4])

print(arr[-3:-1])

print(arr[1:5:2])

print(arr[::2])

**Output:**

[2 3 4 5]

[5 6 7]

[1 2 3 4]

[5 6]

[2 4]

[1 3 5 7]

**4. Write a program for Math operations on array using numpy.**

* **Square Root and Standard Deviation**

**Program:**

import numpy as np

arr=np.array([[1,2,3],[4,5,6]])

print(np.sqrt(arr))

print(np.std(arr))

**Output:**

[[1. 1.41421356 1.73205081]

[2. 2.23606798 2.44948974]]

1.707825127659933

* **Addition, subtraction, multiplication and division of the two matrices**

**Program:**

import numpy as np

arr1=np.array([[1,2,3],[4,5,6]])

arr2=np.array([[1,2,3],[4,5,6]])

print(arr1+arr2)

print(arr1-arr2)

print(arr1\*arr2)

print(arr1/arr2)

**Output:**

[[ 2 4 6]

[ 8 10 12]]

[[0 0 0]

[0 0 0]]

[[ 1 4 9]

[16 25 36]]

[[1. 1. 1.]

[1. 1. 1.]]

* **Vertical and Horizontal Stacking**

**Program:**

import numpy as np

arr1=np.array([[1,2,3],[4,5,6]])

arr2=np.array([[7,8,9],[10,11,12]])

print("vstack:",np.vstack((arr1,arr2)))

print("hstack:",np.hstack((arr1,arr2)))

**Output:**

vstack: [[ 1 2 3]

[ 4 5 6]

[ 7 8 9]

[10 11 12]]

hstack: [[ 1 2 3 7 8 9]

[ 4 5 6 10 11 12]]

* **Ravel:**

**Program:**

import numpy as np

arr=np.array([[1,2,3],[4,5,6]])

print(np.ravel(arr))

**Output:**

[1 2 3 4 5 6]

**5. Write a program for searching .**

**Program:**

import numpy as np

arr = np.array([10, 32, 30, 50, 20, 82, 91, 45])

i = np.where(arr == 30)

print(i)

**Output:**

(array([2], dtype=int64),)

**6. Write a program for sorting.**

**Program:**

import numpy as np

arr=np.array([[1,4,2,3],[9,13,61,1],[43,24,88,22]])

sort\_arr=np.sort(arr)

print(sort\_arr)

**Output:**

[[ 1 2 3 4]

[ 1 9 13 61]

[22 24 43 88]]

**Week - 10: Exploration of Pandas Package**

**1. Import Pandas and Plotpy and explore their functionalities.**

* **Pandas**

**Import panda**

**Or**

**Import pandas as pd**

**Pandas Series:**

**Program:**

Import pandas as pd

a=[1,2,3]

s=pd.series(a)

print(s)

**Output:**

0 1

1 2

2 3

dtype: int64

**Labels:**

**Program:**

import pandas as pd

a=[1,2,3]

s=pd.Series(a)

print(s[0])

**Output:**

1

**Creating Labels:**

**Program**

import pandas as pd

a=[1,2,3]

s=pd.Series(a,index= 'a','b','c')

print(s)

**Output:**

a 1

b 2

c 3

* **Plotpy**

**import matplotlib.pyplot as plt**

**Matplotlib Line Plot:**

**Program:**

import matplotlib.pyplot as plt

import numpy as np

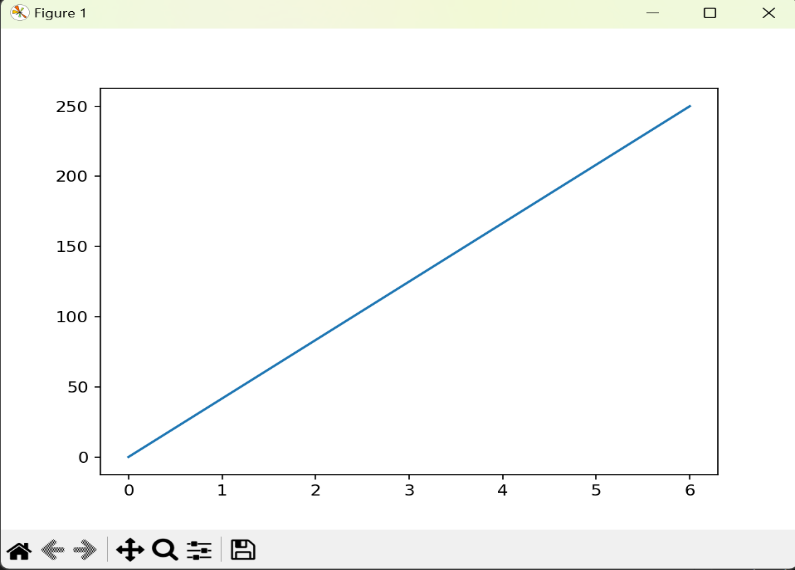
xpoints = np.array([0, 6])

ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)

plt.show()

**output:**



**2. Python Data Frame**

**DataFrames:**

**Program:**

import pandas as pd

a={"Fruits":["apple","mango","kiwi"], "Qty.":[1,2,3], "color":["red","Yellow","Rust"] }

df=pd.DataFrame(a)

print(df)

**Output:**

Fruits Qty. color

0 apple 1 red

1 mango 2 Yellow

2 kiwi 3 Rust

**Index in DataFrame:**

**Program:**

import pandas as pd

a={

"Fruits":["apple","mango","kiwi"],

"Qty":[1,2,3]

}

df=pd.DataFrame(a,index=["x","y","z"])

print(df)

**output:**

Fruits Qty

x apple 1

y mango 2

z kiwi 3

**Loc :**

**Program:**

import pandas as pd

a={

"Fruits":["apple","mango","kiwi"],

"Qty":[1,2,3]

}

df=pd.DataFrame(a,index=["x","y","z"])

**Output:**

print(df.loc["x"])

Fruits apple

Qty 1

Name: x, dtype: object

**Week 11**

**1. Import SciPy and explore their functionalities.**

**Program**

from scipy import special

a = special.exp10(3)

print(a)

 b = special.exp2(3)

print(b)

 c = special.sindg(90)

print(c)

 d = special.cosdg(45)

print(d)

**output:**

1000.0

8.0

1.0

0.7071067811865475

**2. Write a GUI program to create a window wizard having two text labels, two text**

**fields and two buttons as Submit and Reset.**

**Program:**

import tkinter as tk

def submit():

value1 = entry1.get()

value2 = entry2.get()

value3 = entry3.get()

label\_result.config(text=f"Value 1: {value1}\nValue 2: {value2}\nValue 3: {value3}")

def reset():

entry1.delete(0, tk.END)

entry2.delete(0, tk.END)

entry3.delete(0, tk.END)

label\_result.config(text="Result:")

# Create the main window

root = tk.Tk()

root.geometry("300x300")

root.title("Window Wizard")

# Create labels

label1 = tk.Label(root, text="Enter Value 1:")

label2 = tk.Label(root, text="Enter Value 2:")

label3 = tk.Label(root, text="Enter Value 3:")

# Create entry fields

entry1 = tk.Entry(root)

entry2 = tk.Entry(root)

entry3 = tk.Entry(root)

# Create buttons

submit\_button = tk.Button(root, text="Submit", command=submit)

reset\_button = tk.Button(root, text="Reset", command=reset)

# Arrange widgets using grid layout

label1.grid(row=0, column=0, padx=10, pady=5)

entry1.grid(row=0, column=1, padx=10, pady=5)

label2.grid(row=1, column=0, padx=10, pady=5)

entry2.grid(row=1, column=1, padx=10, pady=5)

label3.grid(row=2, column=0, padx=10, pady=5)

entry3.grid(row=2, column=1, padx=10, pady=5)

submit\_button.grid(row=3, column=0, columnspan=2, pady=10)

reset\_button.grid(row=4, column=0, columnspan=2, pady=10)

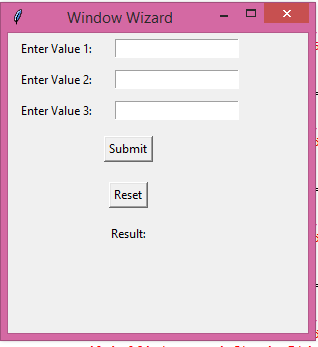
label\_result = tk.Label(root, text="Result:")

label\_result.grid(row=5, column=0, columnspan=2, pady=5)

# Start the main event loop

root.mainloop().

**Output:**



**Week 12**

**1.Write a program to implement Digital Logic Gates – AND, OR, NOT, EX-OR**

**Program(and):**

def AND(a,b):

if a==1 and b==1:

return True

else:

return False

print(AND(1,1),"=AND(1,1)")

print(AND(0,0),"=AND(0,0)")

print(AND(1,0),"=AND(1,0)")

print(AND(0,1),"=AND(0,1)")

**output:**

True =AND(1,1)

False =AND(0,0)

False =AND(1,0)

False =AND(0,1)

**Program(or):**

def OR(a,b):

if a==1 or b==1:

return True

else:

return False

print(OR(1,1),"=OR(1,1)")

print(OR(0,0),"=OR(0,0)")

print(OR(1,0),"=OR(1,0)")

print(OR(0,1),"=OR(0,1)")

**output:**

True =OR(1,1)

False =OR(0,0)

True =OR(1,0)

True =OR(0,1)

**Program(not):**

def NOT(a):

if a==1 :

return False

else:

return True

print(NOT(1),"=NOT(1)")

print(NOT(0),"=NOT(0)")

**output:**  
False =NOT(1)

True =NOT(0)

**Program(EX-OR):**

def XOR\_gate(a, b):

if a != b:

return 1

else:

return 0

print("XOR Gate:", XOR\_gate(5,5))

**output:**

XOR Gate: 0

1. **Write a program to implement Half Adder, Full Adder, and Parallel Adder.**

**Program(Half Adder):**def XOR(a,b):

if a!=b:

return 1

else:

return 0

def AND(a,b):

if a==b:

return 1

else:

return 0

def half\_adder(a,b):

sum=XOR(a,b)

carry=AND(a,b)

return sum,carry

sum,carry=half\_adder(0,0)

print(sum,carry)

sum,carry=half\_adder(0,1)

print(sum,carry)

sum,carry=half\_adder(1,0)

print(sum,carry)

sum,carry=half\_adder(1,1)

print(sum,carry)

**output:**

0 1

1 0

1 0

0 1

**Program(full adder):**  
def half\_adder(a,b):

sum=a^b

carry = a and b

return carry,sum

def full\_adder(carry\_in,a,b):

carry1,sum1=half\_adder(carry\_in,a)

carry2,sum=half\_adder(sum1,b)

carry=carry1 or carry2

return carry,sum

carry,sum=full\_adder(0,0,0)

print(sum,carry)

carry,sum=full\_adder(0,0,1)

print(sum,carry)

carry,sum=full\_adder(0,1,0)

print(sum,carry)

carry,sum=full\_adder(0,1,1)

print(sum,carry)

carry,sum=full\_adder(1,0,0)

print(sum,carry)

carry,sum=full\_adder(1,0,1)

print(sum,carry)

carry,sum=full\_adder(1,1,0)

print(sum,carry)

carry,sum=full\_adder(1,1,1)

print(sum,carry)

output:

0 0

1 0

1 0

0 1

1 0

0 1

0 1

1 1

**Prallel Adder:**

#

# To use these functions, you can run python and then import like -

# from binary\_adder import \*

#

# These methods carry out binary addition via 'digital logic'

# This is really what happens at the logic circuit level.

# So, this is a pretty abstract use of programming to illustrate

# what happens on silicon using code many, many, levels above that!

#

# A binary half adder -- performing addition only using logic operators,

# A half adder simply adds two bits and outputs a sum and carry

#

def half\_adder(a, b):

# ^ is logical xor in python

sum = a ^ b

carry = a and b

return carry,sum

# A binary full adder

# The full adder can add 3 bits (can handle an incoming carry)

# Also returns a sum and carry

#

def full\_adder(carry\_in, a, b):

carry1,sum1 = half\_adder(carry\_in,a)

carry2,sum = half\_adder(sum1,b)

carry = carry1 or carry2

return carry,sum

# This method virtually chains together binary full adders in order

# to add binary numbers of arbitrary size.

#

# a and b are expected to be strings representing binary integers.

#

#

def binary\_adder(a,b):

an = len(a)

bn = len(b)

# Convert strings to list of bits -- very functional syntax here

al = list(int(x,2) for x in list(a))

bl = list(int(x,2) for x in list(b))

# Pad smaller list with 0's

dif = an - bn

# more digits in a than b

if dif > 0:

for i in range(dif):

bl.insert(0,0)

else:

for i in range(abs(dif)):

al.insert(0,0)

print(al)

print(bl)

result = []

carry = 0

# Iterate through list right to left, calling full\_adder each time and

# inserting the sum each time

for i in range(len(al)-1,-1,-1):

carry,sum = full\_adder(carry,al[i],bl[i])

result.insert(0,sum)

print (result)

result.insert(0,carry)

return ''.join(str(x) for x in result)

def test\_binary\_adder(a,b):

result = binary\_adder(a,b)

print(result)

if (int(a,2) + int(b,2)) == int(result,2):

print("Woo hoo! It works")

else:

print("FAIL!!")

print(str(int(a,2)) + " + " + str(int(b,2)) + " = " + str(int(result,2)))

test\_binary\_adder('11111','11111')

**Output:**

[1, 1, 1, 1, 1]

[1, 1, 1, 1, 1]

[0]

[1, 0]

[1, 1, 0]

[1, 1, 1, 0]

[1, 1, 1, 1, 0]

111110

Woo hoo! It works

31 + 31 = 62