**Lab 1**

**Name: Aryan Dilipbhai Langhanoja Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**##POST LAB##**

**AIM:- Write a program to demonstrate different number datatype in python**

**Datatype:- Data types specify the type of data that can be stored inside a variable.**

|  |  |  |
| --- | --- | --- |
| Data Types | Classes | Description |
| Numeric | int, float, complex | holds numeric values |
| String | str | holds sequence of characters |
| Sequence | list, tuple, range | holds collection of items |
| Mapping | dict | holds data in key-value pair form |
| Boolean | bool | holds either true or false |
| Set | set, frozeenset | hold collection of unique items |

**Task1: Print interger, float, string and boolean with their types in python.**

**Python Code:**

integer\_variable = 30

float\_variable = 1.7812

string\_variable = "Aryan Langhanoja"

boolean\_variable = True

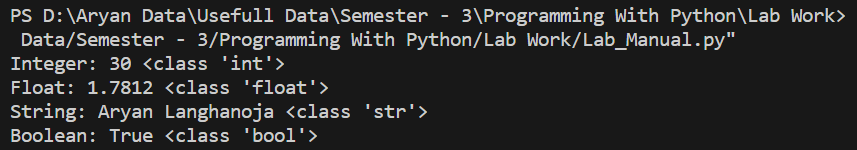
print("Integer:", integer\_variable, type(integer\_variable))

print("Float:", float\_variable, type(float\_variable))

print("String:", string\_variable, type(string\_variable))

print("Boolean:", boolean\_variable, type(boolean\_variable))

**Output:**

****

**Task2: Print list and tuple with their types in python.**

**Python Code:**

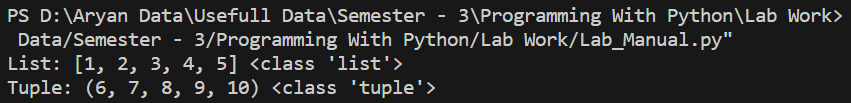
list\_variable = [1, 2, 3, 4, 5]

tuple\_variable = (6, 7, 8, 9, 10)

print("List:", list\_variable, type(list\_variable))

print("Tuple:", tuple\_variable, type(tuple\_variable))

**Output:**

****

**Task3: Print dictionary and set with their types in python.**

**Python Code:**

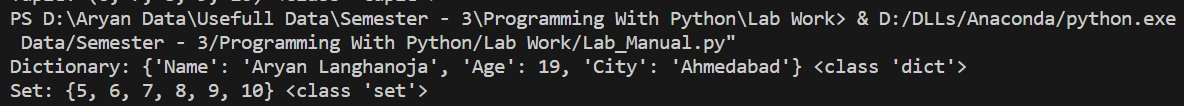
dictionary\_variable = {

"Name": "Aryan Langhanoja", "Age": 19, "City": "Ahmedabad"}

set\_variable = {5, 6, 7, 8, 9, 10}

print("Dictionary:", dictionary\_variable, type(dictionary\_variable))

print("Set:", set\_variable, type(set\_variable))

**Output:**

**Task4: Python program to demonstrate data type of given number**

**Python Code:**

a = 5

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

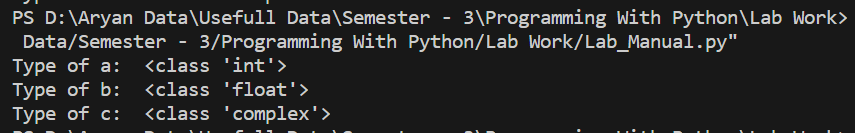
b = 5.0

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

c = 2 + 4j

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

**Output:**

****

**Task5: Python program to demonstrate data type of given number**

**Python Code:**

tuple1 = tuple([1, 2, 3, 4, 5])

print("First of tuple")

print(tuple1[0])

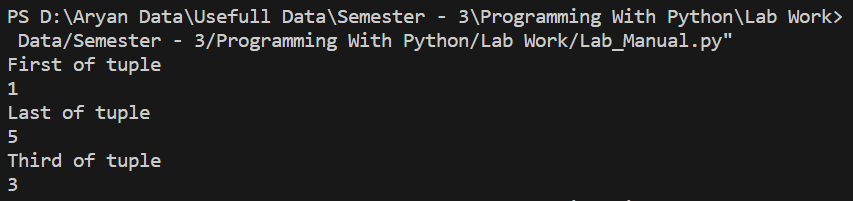
print("Last of tuple")

print(tuple1[-1])

print("Third of tuple")

print(tuple1[-3])

**Output:**

****

**Lab 2**

**Name: Aryan Dilipbhai Langhanoja**

**Date: 03-11-2023**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditionals, loops and functions**

**AIM:- Write a program to perform different arithmetic operations on numbers in python**

**Arithmetic Operations:- Mathematical calculations that we can perform on numeric values such as integers and floating-point numbers.**

| **Operator** | **Description** | **Syntax** |
| --- | --- | --- |
| **+** | **Addition: adds two operands** | **x + y** |
| **–** | **Subtraction: subtracts two operands** | **x – y** |
| **\*** | **Multiplication: multiplies two operands** | **x \* y** |
| **/** | **Division (float): divides the first operand by the second** | **x / y** |
| **//** | **Division (floor): divides the first operand by the second** | **x // y** |
| **%** | **Modulus: returns the remainder when the first operand is divided by the second** | **x % y** |
| **\*\*** | **Power (Exponent): Returns first raised to power second** | **x \*\* y** |

**Task1: Print arithmetic operations for two values.**

**Python Code:**

val1 = 5

val2 = 3

res = val1 + val2

print(res)

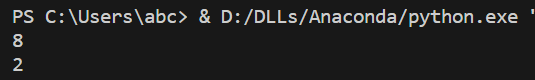
val1 = 5

val2 = 3

res = val1 - val2

print(res)

**Output:**

****

**Task2: Print arithmetic operations for two values.**

**Python Code:**

val1 = 5

val2 = 3

res = val1 \* val2

print(res)

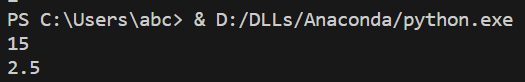
val1 = 5

val2 = 2

res = val1 / val2

print(res)

**Output:**

****

**Task3: Print arithmetic operations for two values.**

**Python Code:**

val1 = 5

val2 = 2

res = val1 % val2

print(res)

**Output:**

****

**Task4: Print arithmetic operations for two values.**

**Python Code:**

val1 = 5

val2 = 3

res = val1 \*\* val2

print(res)

**Output:**



**Task5: Print arithmetic operations for two values.**

**Python Code:**

val1 = 5

val2 = 2

res = val1 // val2

print(res)

**Output:**

****

**Lab 3**

**Name: Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditionals, loops and functions**

**AIM:- Write a program to create, concatenate and print a string and accessing sub string from a given string.**

**Introduction:- String Concatenation is the technique of combining two strings.**

**Task1: String Concatenation using + operator in python**

**Python Code:**

var1 = "Aryan "

var2 = "Langhanoja"

var3 = var1 + var2

print(var3)

**Output:**



**Task2: String Concatenation using join() method in python**

**Python Code:**

var1 = "Aryan "

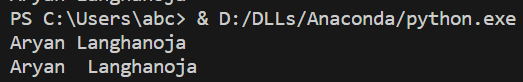
var2 = "Langhanoja"

print("".join([var1, var2]))

var3 = " ".join([var1, var2])

print(var3)

**Output:**

****

**Task3: String Concatenation using % operator in python**

**Python Code:**

var1 = "Aryan "

var2 = "Langhanoja "

print("% s % s" % (var1, var2))

**Output:**

****

**Task4: String Concatenation using format() function in python**

**Python Code:**

var1 = "Aryan"

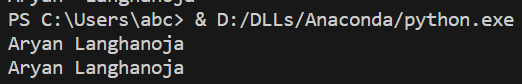
var2 = "Langhanoja"

print("{} {}".format(var1, var2))

var3 = "{} {}".format(var1, var2)

print(var3)

**Output:**



**Task5: String Concatenation using “,” in python**

**Python Code:**

var1 = "Aryan"

var2 = "Dilipbhai"

var3 = "Langhanoja"

print(var1, var2, var3)

**Output:**

****

**Lab 4**

**Name: Date:**

**Enrollment No:**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditionals, loops and functions**

**AIM:- Write a program script to print the current date in following format “Sat August 26 22:12:59 IST 2023”**

**Introduction:- Pytz brings the Olson tz database into Python and thus supports almost all time zones. This module serves the date-time conversion functionalities and helps user serving international client’s base. It enables time-zone calculations in our Python applications and also allows us to create timezone aware date time instances.**

**Task1: Write a program script to print the current date in following format “Sat August 26 22:12:59 IST 2023”**

**Python Code:**

from datetime import datetime

import pytz

format\_string = "%A, %d %B %Y %I:%M %p (%Z)"

desired\_timezone = pytz.timezone('America/New\_York')

curr\_time = datetime.now(desired\_timezone)

formatted\_time = curr\_time.strftime(format\_string)

print(formatted\_time)

**Output:**



**Task2: It returns the list and set of commonly used timezones with pytz.common\_timezones, pytz.common\_timezones\_set.**

**Python Code:**

import pytz

print('Commonly used time-zones:',

pytz.common\_timezones, '\n')

print('Commonly used time-zones-set:',

pytz.common\_timezones\_set, '\n')

**Output:**

****

**Task3: Libraries returns the list all the available timezones with pytz.all\_timezones.**

**Python Code:**

import pytz

print('the supported timezones by the pytz module:',

pytz.all\_timezones, '\n')

**Output:**

****

**Task4: To represent dictionary of country code as supported time zone as value.**

**Python Code:**

import pytz

print('country\_timezones =')

for key, val in pytz.country\_timezones.items():

print(key, '=', val, end=',')

print('\n')

print('Time-zones supported by Antarctica =', pytz.country\_timezones['AQ'])

**Output:**



**Task5: String Concatenation using “,” in python**

**Python Code:**

var1 = "Aryan"

var2 = "Dilipbhai"

var3 = "Langhanoja"

print(var1, var2, var3)

**Output:**

****

**Lab 5**

**Name: Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditionals, loops and functions**

**AIM:- Write a python program to create, append and remove lists in python**

**Introduction:- Lists are used to store multiple items in a single variable. Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are**[**Tuple**](https://www.w3schools.com/python/python_tuples.asp)**,**[**Set**](https://www.w3schools.com/python/python_sets.asp)**, and**[**Dictionary**](https://www.w3schools.com/python/python_dictionaries.asp)**, all with different qualities and usage. Lists are created using square brackets**

**Task1: Write a program to create list in python**

**Python Code:**

thislist = ["Aryan", "Dilipbhai", "Langhanoja"]

print(thislist)

**Output:**



**Task2: Write a python program th remove the word from a list using remove**

**Python Code:**

thislist = ["Aryan", "Dilipbhai", "Langhanoja"]

thislist.remove("Dilipbhai")

print(thislist)

**Output:**

****

**Task3: Write a python program to add in the list using append**

**Python Code:**

fruits = ["Aryan", "Langhanoja", "From"]

fruits.append("Ahmedabad")

print(fruits)

**Output:**

****

**Task4: Write a python program to find the length of a list**

**Python Code:**

thislist = ["Aryan", "Dilipbhai", "Langhanoja"]

print(len(thislist))

**Output:**



**Task5: Write a python program using the pop method to remove words by indexing**

**Python Code:**

thislist = ["Aryan", "Dilipbhai", "Langhanoja"]

thislist.pop()

print(thislist)

**Output:**

****

**Lab 6**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:-**

**Enrollment No:- 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Develop programs to understand the control structure in python**

**Introduction:- A code is written to allow making choices and several pathways through the program to be followed depending on shifts in variable values. All programming languages contain a pre-included set of control structures that enable these control flows to execute, which makes it conceivable.**

**Task1: Python program to show how a sequential control structure**

**Python Code:**

a = 25

b = 20

c = a - b

d = a + b

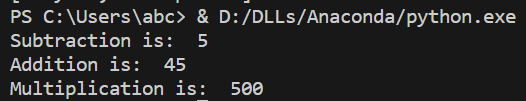
e = a \* b

print("Subtraction is: ", c)

print("Addition is: ", d)

print("Multiplication is: ", e)

**Output:**

****

**Task2: Write a python code for loop control statement**

**Python Code:**

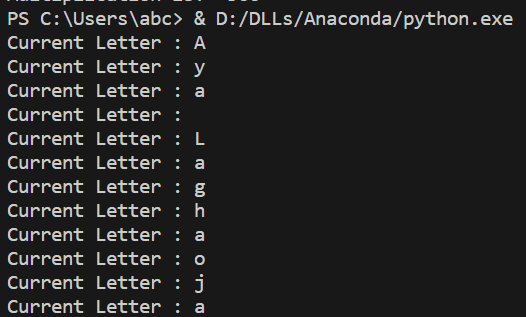
for letter in 'Aryan Langhanoja':

if letter == 'r' or letter == 'n':

continue

print('Current Letter :', letter)

**Output:**

****

**Task3: Write a python code to bring control out of the loop**

**Python Code:**

for letter in 'Aryan':

if letter == 'r' or letter == 'a':

break

print('Current Letter :', letter)

**Output:**

****

**Task4: Write a python code using for loop**

**Python Code:**

l = [1, 2, 3, 4, 5, 6]

for i in range(len(l)):

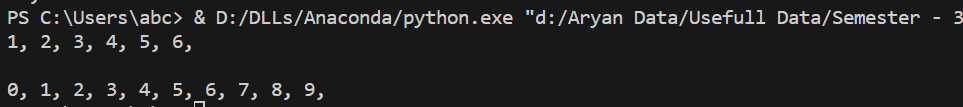
print(l[i], end = ", ")

print("\n")

for j in range(0,10):

print(j, end = ", ")

**Output:**



**Task5: Write a python code using while loop**

**Python Code:**

b = 9

a = 1

while a < b:

print(a, end = " ")

a = a + 1

print("completed")

**Output:**

****

**Lab 7**

**Name:- Aryan Dilipbhai Langhanoja**

**Date:-**

**Enrollment No:- 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Develop programs to understand the control structure in python**

**Introduction:- Tuples are used to store multiple items in a single variable. Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are**[**List**](https://www.w3schools.com/python/python_lists.asp)**,**[**Set**](https://www.w3schools.com/python/python_sets.asp)**, and**[**Dictionary**](https://www.w3schools.com/python/python_dictionaries.asp)**, all with different qualities and usage. A tuple is a collection which is ordered and unchangeable. Tuples are written with round brackets.**

**Task1: Write a python program to create a simple tuple**

**Python Code:**

thistuple = ("Aryan", "Dilipbhai", "Langhanoja")

print(thistuple)

**Output:**

****

**Task2: Write a python program to allow the tuple duplicates**

**Python Code:**

thistuple = ("Aryan", "Dilipbhai", "Langhanoja", "From", "Ahmedabad")

print(thistuple)

**Output:**



**Task3 : Write a python program to show tuple items and data types**

**Python Code:**

tuple1 = ("Aryan", "Dilipbhai", "Langhanoja")

tuple2 = (1, 5, 7, 9, 3)

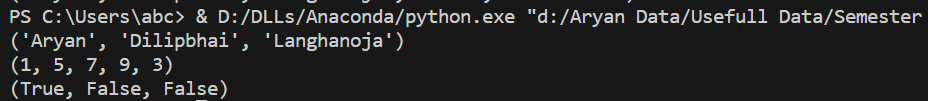
tuple3 = (True, False, False)

print(tuple1)

print(tuple2)

print(tuple3)

**Output:**

****

**Task4: Write a python program to join two transport using operator**

**Python Code:**

tuple1 = ("A", "B" , "C")

tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2

print(tuple3)

**Output:**

****

**Task5: Write a python program to multiply two numbers using operator**

**Python Code:**

fruits = ("Aryan", "Dilipbhai", "Langhanoja")

mytuple = fruits \* 2

print(mytuple)

**Output:**

****

**Lab 8**

**Name:- Aryan Dilipbhai Langhanoja**

**Date:-**

**Enrollment No:- 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Write a program to demonstrate working with dictionaries in python**

**Introduction:- Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.**

**Task1: Write a python program to create a simple tuple**

**Python Code:**

thisdict = {

"University": "Marwadi University",

"College": "Faculty of Technology",

"Year": 2023

}

print(thisdict)

**Output:**

****

**Task2: Write a python program for duplicates not allowed**

**Python Code:**

thisdict = {

"University": "Marwadi University",

"College": "Faculty of Technology",

"Year": 2022 ,

"Year-2" : 2026

}

print(len(thisdict))

**Output:**



**Task3: Write a python program to dict() constructor to make a dictionary**

**Python Code:**

thisdict = dict(name = "Aryan Langhanoja", year = 2023, country = "India")

print(thisdict)

**Output:** 

**Task4: Write a python program to add elements in the dictionary**

**Python Code:**

thisdict = {

"College": "Marwadi University",

"Aame": "Aryan Langhanoja",

"year": 2023

}

thisdict["Department"] = "ICT"

print(thisdict)

**Output:**



**Task5: Write a python program to remove items from a any element**

**Python Code:**

thisdict = {

"College": "Marwadi University",

"Name": "Aryan Langhanoja",

"year": 2023

}

thisdict.pop("year")

print(thisdict)

**Output:**

****

**Lab 9**

**Name:- Aryan Dilipbhai Langhanoja**

**Date:-**

**Enrollment No:- 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Write a python program to convert temperature to and from Celsius to Fahrenheit.**

**Introduction:- The centigrade scale, which is also called the Celsius scale, was developed by Swedish astronomer Andres Celsius. In the centigrade scale, water freezes at 0 degrees and boils at 100 degrees.**

**Task1: Write a python program to convert temperature into different measures.**

**Python Code:**

celsius = 45

fahrenheit = (celsius \* 1.8) + 32

print('%.2f Celsius is Equivalent to: %.2f Fahrenheit'

% (celsius, fahrenheit))

**Output:**

****

**Task2: Write a python program to convert temperature into different measures.**

**Python Code:**

fahrenheit = 273

celsius = (fahrenheit-32)/1.8

print('%.2f Fahrenheit is equivalent to: %.2f Celsius'

% (fahrenheit ,celsius))

**Output:**

****

**Lab 10**

**Name:- Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No:- 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Write a python program to define a module and import a specific function in that module to another program.**

**Introduction:- As our program grows bigger, it may contain many lines of code. Instead of putting everything in a single file, we can use modules to separate codes in separate files as per their functionality. This makes our code organized and easier to maintain. Module is a file that contains code to perform a specific task. A module may contain variables, functions, classes etc.**

**Task1: Write a python program to import math function**

**Python Code:**

import math

print("The value of pi is", math.pi)

**Output:**

****

**Task2: Write a python program to import function by renaming it .**

**Python Code:**

import math as m

print(m.pi)

**Output:**

****

**Task3: Write a python program to convert temperature into different measures.**

**Python Code:**

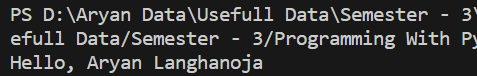
def greeting(name):

print("Hello, " + name)

import mymodule

mymodule.greeting("Aryan Langhanoja")

**Output:**

****

**Task4: Write a python program to use variables in the module.**

**Python Code:**

def person1(name):

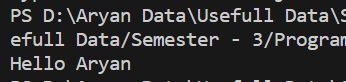
return "Hello " + name

import mymodule

a = mymodule.person1("Aryan")

print(a)

**Output:**

****

**Task5: Write a python program for built in function in the module**

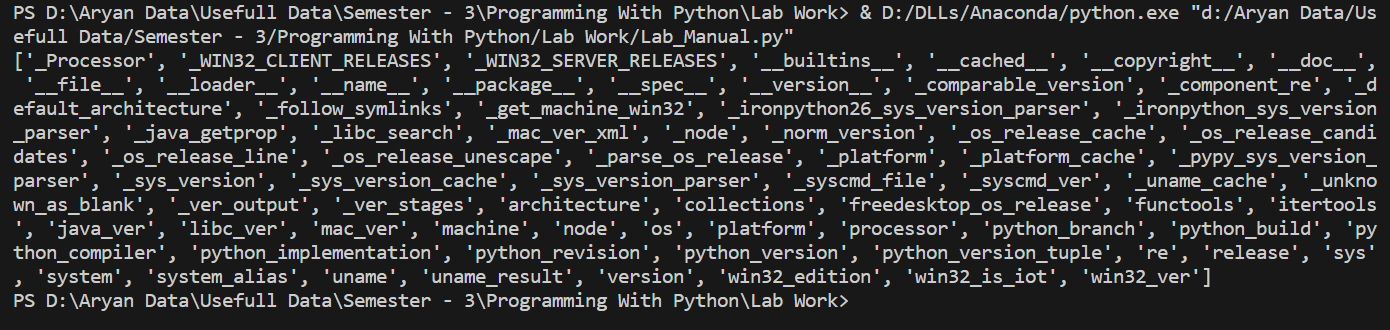
**Python Code:**

import platform

x = dir(platform)

print(x)

**Output:**

****

**Lab 11**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Write a python program to call data member and function using classes and object.**

**Introduction:- A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.**

**Task1: Write a python program to create object**

**Python Code:**

class name:

attr1 = "marwadi"

attr2 = "university"

def fun(self):

print("I'm study in", self.attr1)

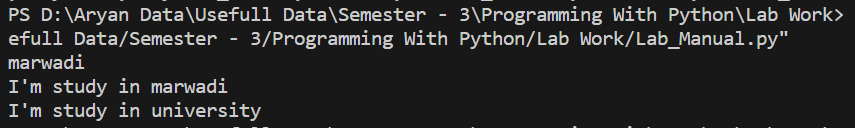
print("I'm study in", self.attr2)

Rodger = name()

print(Rodger.attr1)

Rodger.fun()

**Output:**

****

**Task2: Write a python program using int() method**

**Python Code:**

class Person:

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

self.name = name

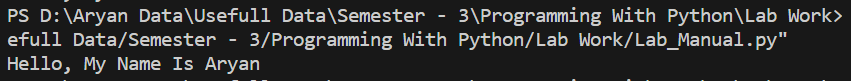
def say\_hi(self):

print('Hello, My Name Is', self.name)

p = Person('Aryan')

p.say\_hi()

**Output:**



**Task3: Write a python program using class with input.**

**Python Code:**

class Student:

'A student class'

stuCount = 0

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

self.name = input('Enter student name:')

self.rollno = input('Enter student rollno:')

Student.stuCount += 1

def displayStudent(self):

print("Name:", self.name, "Rollno:", self.rollno)

stu1 = Student()

stu2 = Student()

stu3 = Student()

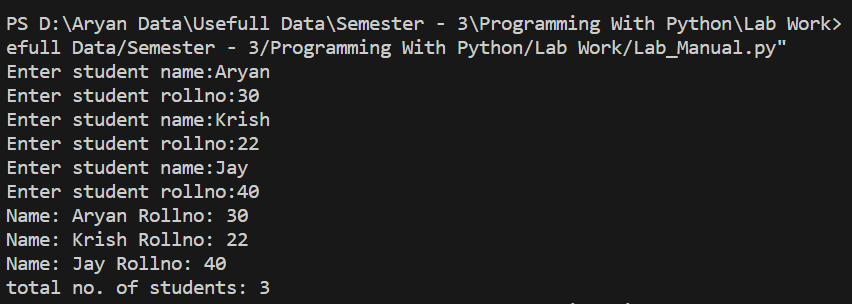
stu1.displayStudent()

stu2.displayStudent()

stu3.displayStudent()

print('total no. of students:', Student.stuCount)

**Output:**

****

**Task4: Write a python program using self parameter.**

**Python Code:**

class harsh:

def \_\_init\_\_(somename, name, company):

somename.name = name

somename.company = company

def show(somename):

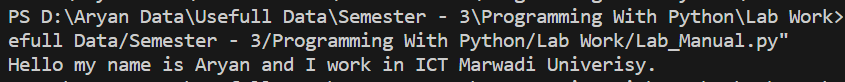
print("Hello my name is " + somename.name +

" and I work in "+somename.company+".")

obj = harsh("Aryan", "ICT Marwadi Univerisy")

obj.show()

**Output:**

****

**Task5: Write a python program using instance variables using a constructor**

**Python Code:**

class Dog:

animal = 'dog'

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

self.breed = breed

def setColor(self, color):

self.color = color

def getColor(self):

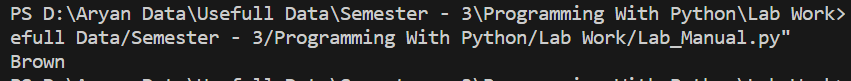
return self.color

Rodger = Dog("Pug")

Rodger.setColor("Brown")

print(Rodger.getColor())

**Output:**

****

**Lab 12**

**Name: Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Practical based on Numpy ndarray.**

**Introduction:- Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array.A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as ndarray. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.**

**Task1: Write a python program to demonstrate basic array**

**Python Code:**

import numpy as np

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

[4, 2, 5]])

print("Array is of type: ", type(arr))

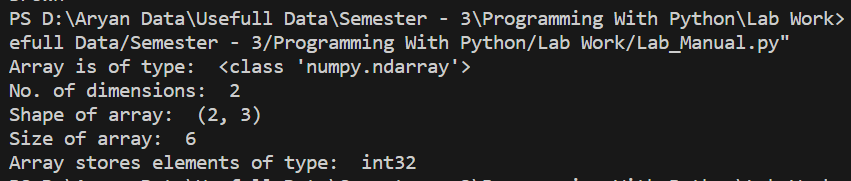
print("No. of dimensions: ", arr.ndim)

print("Shape of array: ", arr.shape)

print("Size of array: ", arr.size)

print("Array stores elements of type: ", arr.dtype)

**Output:**

****

**Tas2: Write a python program for array indexing.**

**Python Code:**

import numpy as np

arr = np.array([[-1, 2, 0, 4],

[4, -0.5, 6, 0],

[2.6, 0, 7, 8],

[3, -7, 4, 2.0]])

temp = arr[:2, ::2]

print("Array with first 2 rows and alternate"

"columns(0 and 2):\n", temp)

temp = arr[[0, 1, 2, 3], [3, 2, 1, 0]]

print("Elements at indices (0, 3), (1, 2), (2, 1),"

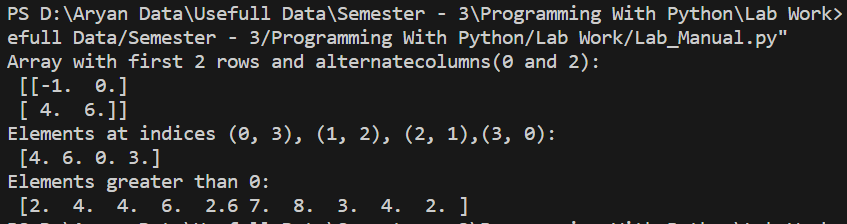
"(3, 0):\n", temp)

cond = arr > 0

temp = arr[cond]

print("Elements greater than 0:\n", temp)

**Output:**



**Task3: Write a python program to convert temperature into different measures.**

**Python Code:**

import numpy as np

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

print("Adding 1 to every element:", a+1)

print("Subtracting 3 from each element:", a-3)

print("Multiplying each element by 10:", a\*10)

print("Squaring each element:", a\*\*2)

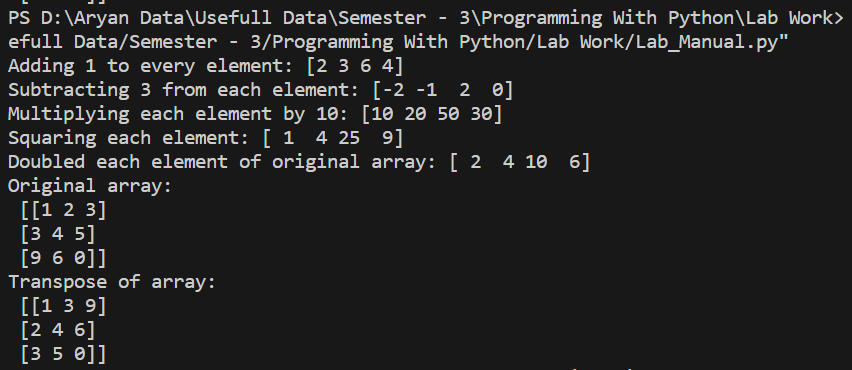
a \*= 2

print("Doubled each element of original array:", a)

a = np.array([[1, 2, 3], [3, 4, 5], [9, 6, 0]])

print("Original array:\n", a)

print("Transpose of array:\n", a.T)

**Output: **

**Task4: Write a python program for unary operator.**

**Python Code:**

import numpy as np

arr = np.array([[1, 5, 6],

[4, 7, 2],

[3, 1, 9]])

print("Largest element is:", arr.max())

print("Row-wise maximum elements:",

arr.max(axis=1))

print("Column-wise minimum elements:",

arr.min(axis=0))

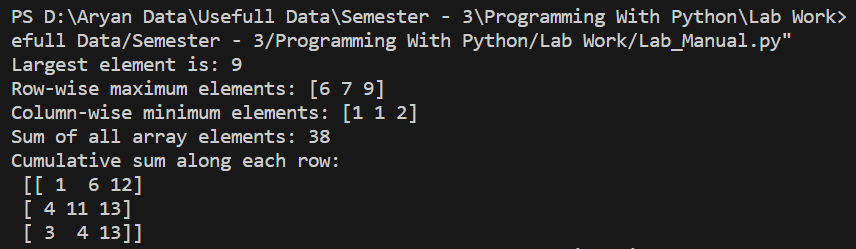
print("Sum of all array elements:",

arr.sum())

print("Cumulative sum along each row:\n",

arr.cumsum(axis=1))

**Output:**



**Task5: Write a python program for binary operator.**

**Python Code:**

import numpy as np

a = np.array([[1, 2],

[3, 4]])

b = np.array([[4, 3],

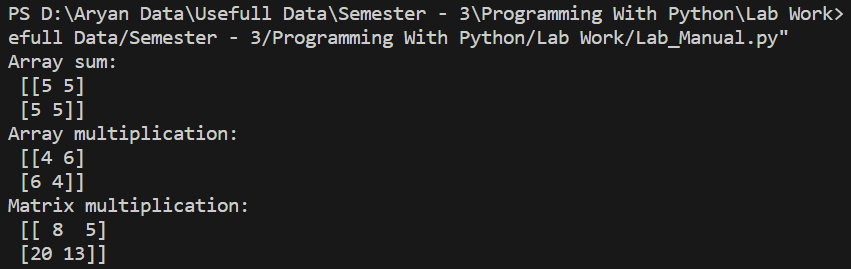
[2, 1]])

print("Array sum:\n", a + b)

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

print("Matrix multiplication:\n", a.dot(b))

**Output:**

****

**Lab 13**

**Name: Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Practical based on Pandas Data Structure**

**Introduction:-** [**Pandas**](https://www.geeksforgeeks.org/pandas-tutorial/)**is an open-source library that uses for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. It offers a tool for cleaning and processes your data. It is the most popular Python library that is used for data analysis.**

**Task1: Write a python program for series holding the char data type**

**Python Code:**

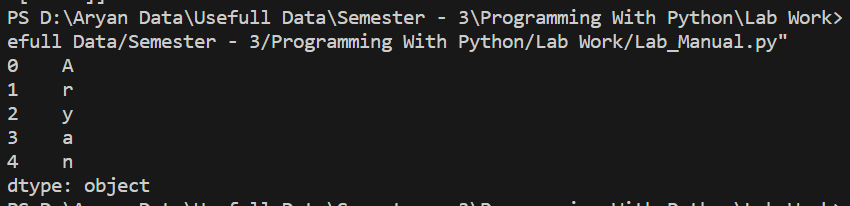
import pandas as pd

list = ['A', 'r', 'y', 'a', 'n']

res = pd.Series(list)

print(res)

**Output:**

****

**Tas2: Write a python program for series holding the int data type.**

**Python Code:**

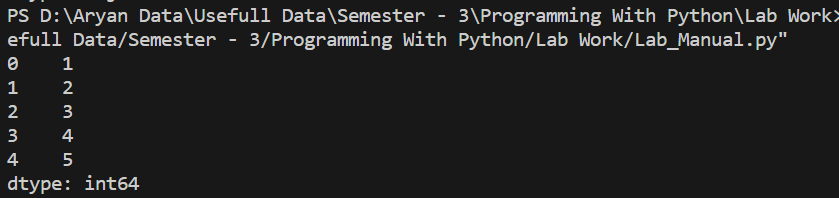
import pandas as pd

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

res = pd.Series(list)

print(res)

**Output:**

****

**Task3: Write a python program for holding a dictionary.**

**Python Code:**

import pandas as pd

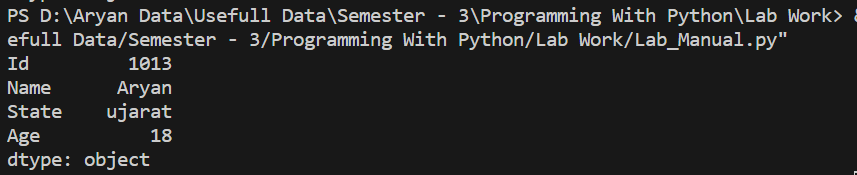
dic = {'Id': 1013, 'Name': 'Aryan',

'State': 'Maniput', 'Age': 24}

res = pd.Series(dic)

print(res)

**Output:**

****

**Task4: Write a python program to create dataframe.**

**Python Code:**

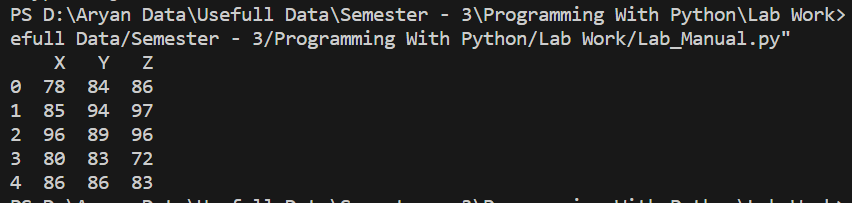
import pandas as pd

df = pd.DataFrame({'X': [78, 85, 96, 80, 86], 'Y': [

84, 94, 89, 83, 86], 'Z': [86, 97, 96, 72, 83]})

print(df)

**Output:**



**Task5: Write a python program using key / value object as series**

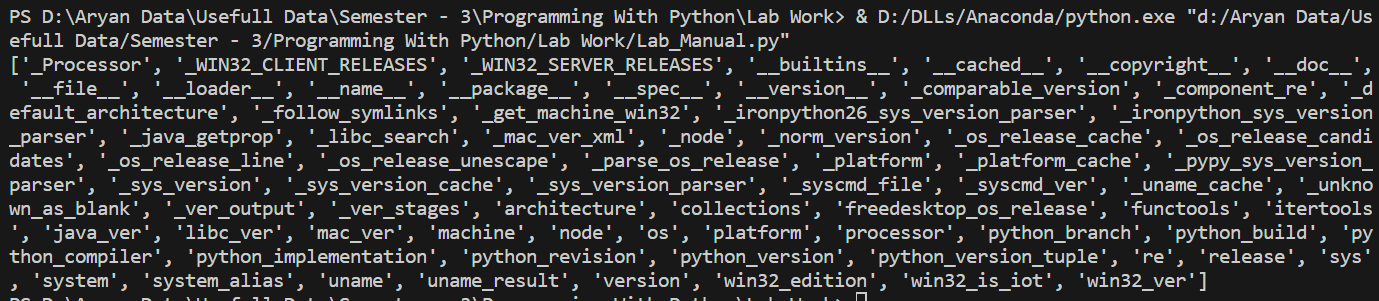
**Python Code:**

import platform

x = dir(platform)

print(x)

**Output:**

****

**Lab 14**

**Name: Aryan Diliphai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Simulate continues time elementary signals**

**Introduction:- To simulate continuous-time elementary signals in Python, you can use libraries like NumPy and Matplotlib for generating and plotting the signals.**

**Task1: Write a python program to represent unit impulse function.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

t = np.linspace(-1, 1, 201)

impulse = np.zeros\_like(t)

impulse[t == 0] = 1

plt.plot(t, impulse)

plt.title('Unit Impulse (Dirac Delta) Function')

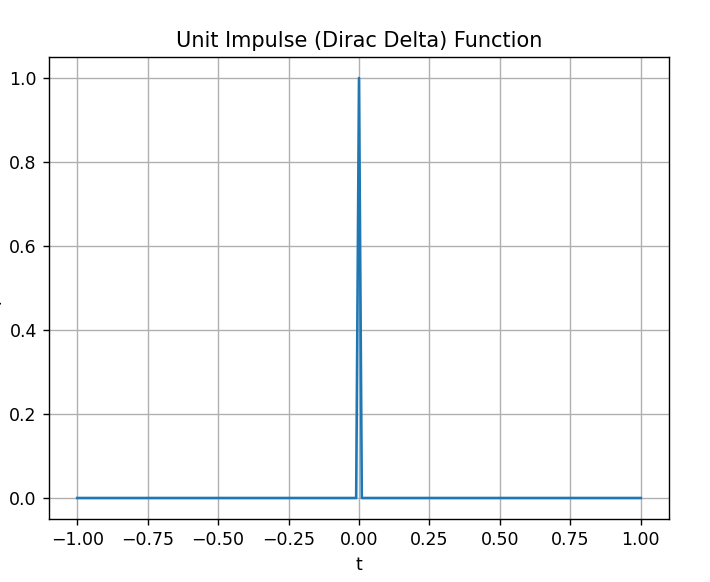
plt.xlabel('t')

plt.ylabel('Amplitude')

plt.grid(True)

plt.show()

**Output:**



**Tas2: Write a python program to represent ramp function.**

**Python Code:**

ramp = np.maximum(0, t)

plt.plot(t, ramp)

plt.title('Ramp Function')

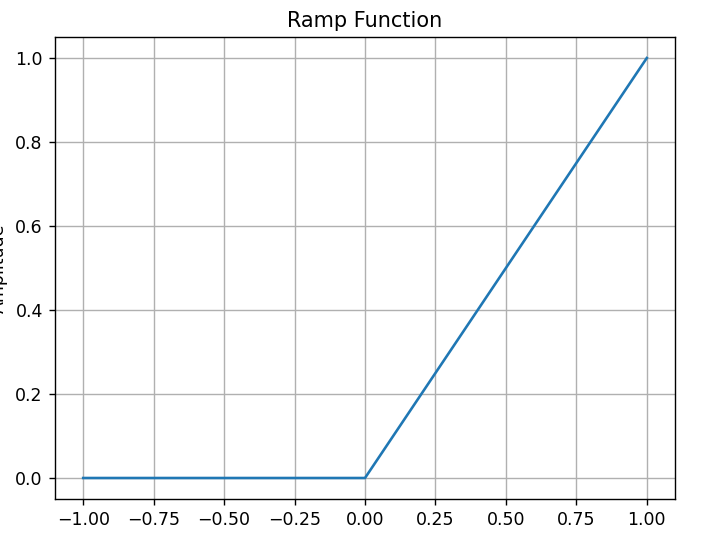
plt.xlabel('t')

plt.ylabel('Amplitude')

plt.grid(True)

plt.show()

**Output:**



**Task3: Write a python program to represent unit step function.**

**Python Code:**

unit\_step = np.heaviside(t, 1)

plt.plot(t, unit\_step)

plt.title('Unit Step Function')

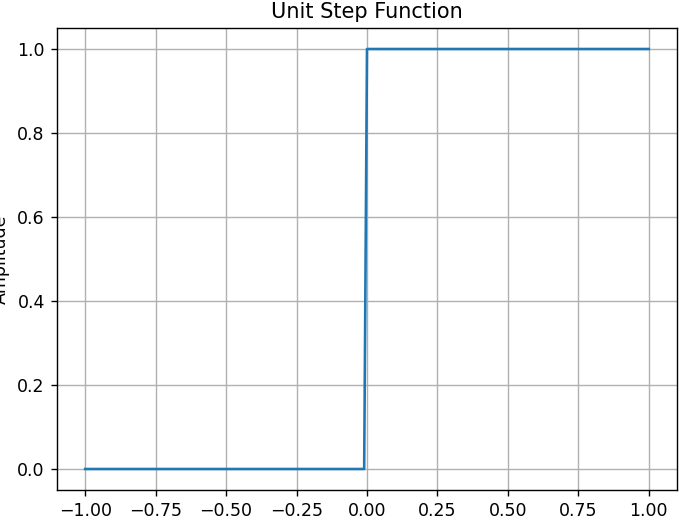
plt.xlabel('t')

plt.ylabel('Amplitude')

plt.grid(True)

plt.show()

**Output:**



**Task4: Write a python program to represent exponential decay function.**

**Python Code:**

decay = np.exp(-t)

plt.plot(t, decay)

plt.title('Exponential Decay Function')

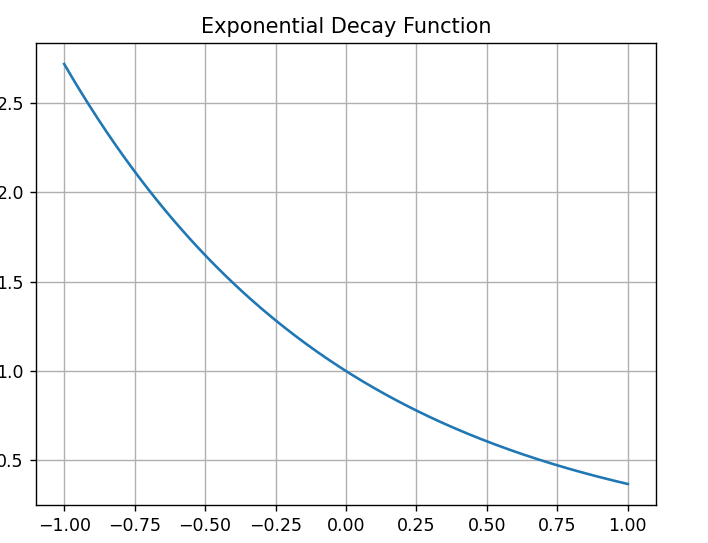
plt.xlabel('t')

plt.ylabel('Amplitude')

plt.grid(True)

plt.show()

**Output:**



**Task5: Write a python program to represent sine wave function.**

**Python Code:**

frequency = 2

sine\_wave = np.sin(2 \* np.pi \* frequency \* t)

plt.plot(t, sine\_wave)

plt.title('Sine Wave')

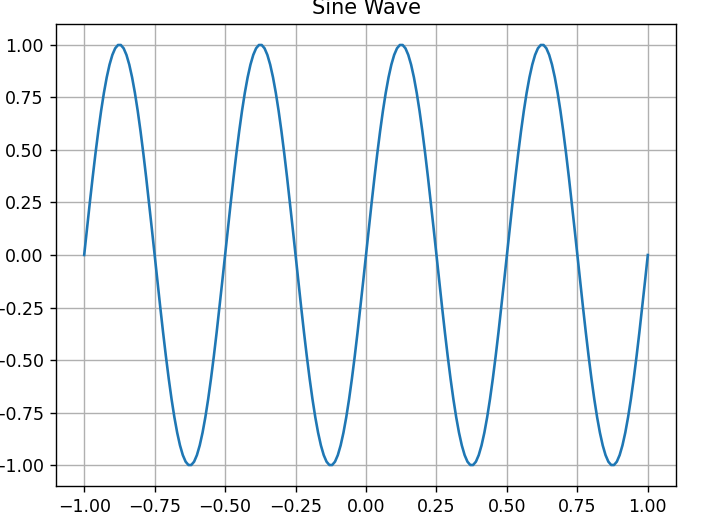
plt.xlabel('t')

plt.ylabel('Amplitude')

plt.grid(True)

plt.show()

**Output:**



**Lab 15**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Simulate basic operations on continuous time elementary signals**

**Introduction:- Continuous-time elementary signals simulated in Python. We'll cover addition, scaling, time shifting, differentiation, and integration using some common elementary signals. We'll use the numpy library for numerical calculations and matplotlib for plotting.**

**Task1: Write a python program to add of two signals(cosine+sine)**

**Python Code:**

signal1 = np.cos(2 \* np.pi \* t)

signal2 = np.sin(2 \* np.pi \* t)

result\_signal = signal1 + signal2

plt.figure()

plt.subplot(3, 1, 1)

plt.plot(t, signal1, label='cosine')

plt.legend()

plt.subplot(3, 1, 2)

plt.plot(t, signal2, label='sine')

plt.legend()

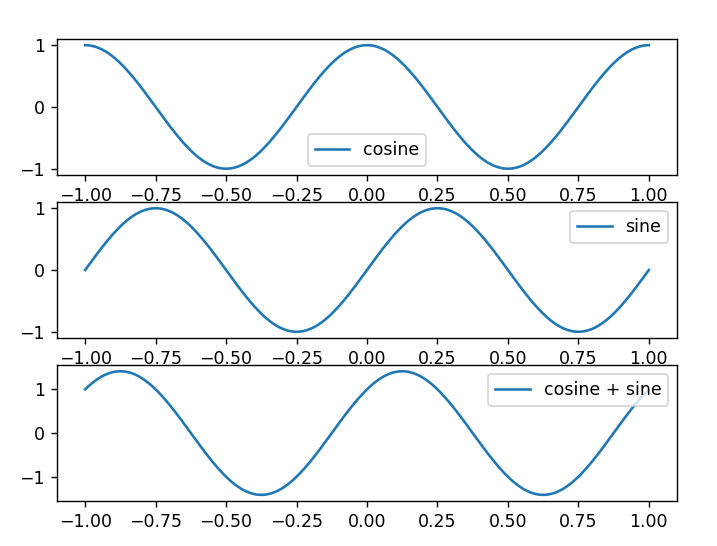
plt.subplot(3, 1, 3)

plt.plot(t, result\_signal, label='cosine + sine')

plt.legend()

plt.show()

**Output:**



**Tas2: Write a python program for scaling a signal.**

**Python Code:**

original\_signal = np.sin(2 \* np.pi \* t)

scaled\_signal = 2 \* original\_signal

plt.figure()

plt.subplot(2, 1, 1)

plt.plot(t, original\_signal, label='Original sine')

plt.legend()

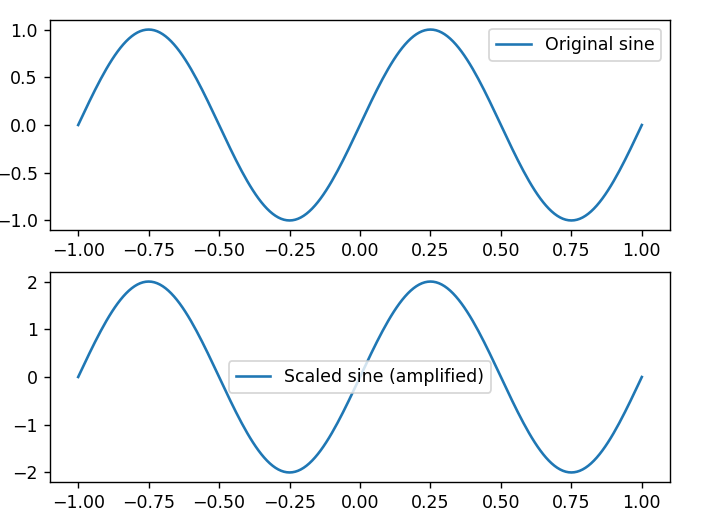
plt.subplot(2, 1, 2)

plt.plot(t, scaled\_signal, label='Scaled sine (amplified)')

plt.legend()

plt.show()

**Output:**



**Task3: Write a python program for time shifting.**

**Python Code:**

shifted\_signal = np.exp(-(t - 1)) \* np.sin(2 \* np.pi \* (t - 1))

plt.figure()

plt.subplot(2, 1, 1)

plt.plot(t, original\_signal, label='Original signal')

plt.legend()

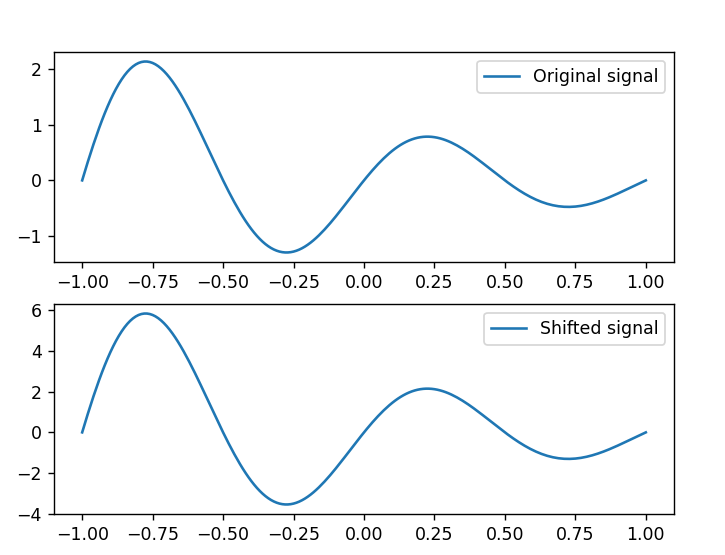
plt.subplot(2, 1, 2)

plt.plot(t, shifted\_signal, label='Shifted signal')

plt.legend()

plt.show()

**Output:**



**Task4: Write a python program for differentiation of a signal.**

**Python Code:**

original\_signal = np.exp(-t) \* np.sin(2 \* np.pi \* t)

derivative\_signal = np.gradient(original\_signal, t)

plt.figure()

plt.subplot(2, 1, 1)

plt.plot(t, original\_signal, label='Original signal')

plt.legend()

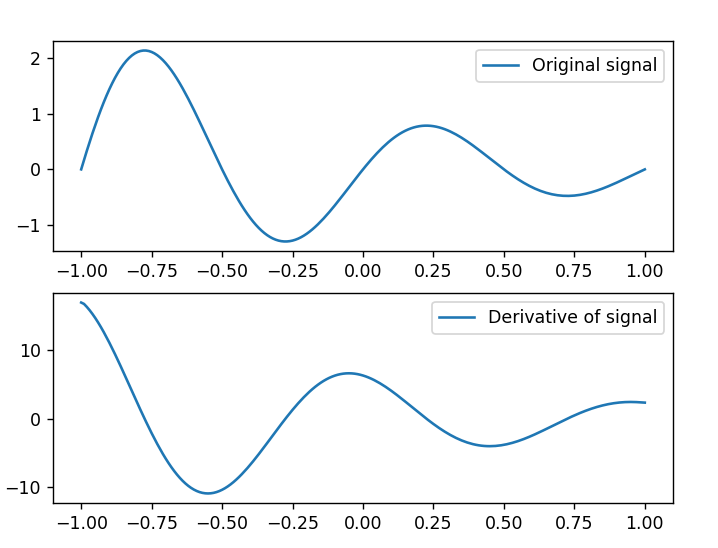
plt.subplot(2, 1, 2)

plt.plot(t, derivative\_signal, label='Derivative of signal')

plt.legend()

plt.show()

**Output:**



**Task5: Write a python program for integration of a signal**

**Python Code:**

original\_signal = np.exp(-t) \* np.sin(2 \* np.pi \* t)

integral\_signal = np.cumsum(original\_signal) \* (t[1] - t[0])

plt.figure()

plt.subplot(2, 1, 1)

plt.plot(t, original\_signal, label='Original signal')

plt.legend()

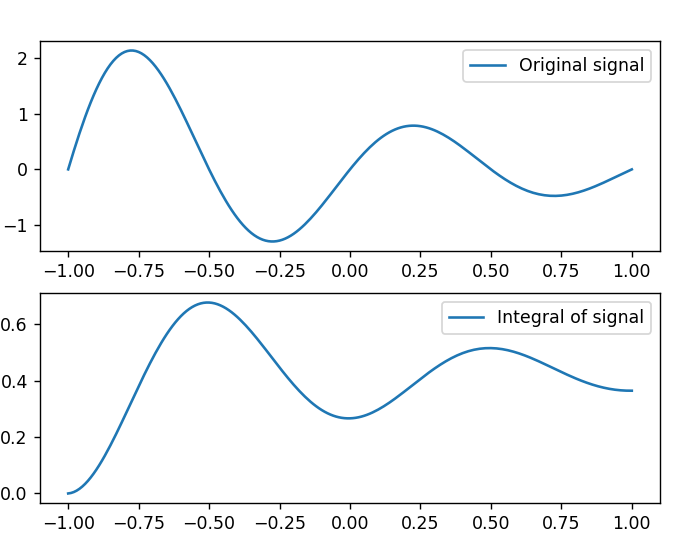
plt.subplot(2, 1, 2)

plt.plot(t, integral\_signal, label='Integral of signal')

plt.legend()

plt.show()

**Output:**



**Lab 16**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Simulate sampling on contiguous time signals and generate a frequency spectrum of signals before and after sampling**

**Introduction:- To simulate sampling of continuous-time signals and generate their frequency spectra before and after sampling, you can use Python with libraries such as numpy and matplotlib.**

**Task1: Write a python program for continuous time sinusoidal signal.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

Fs = 1000

T = 1.0

t\_continuous = np.linspace(0, T, int(T \* Fs), endpoint=False)

freq1 = 5

signal\_continuous1 = np.sin(2 \* np.pi \* freq1 \* t\_continuous)

Fs\_new1 = 50  # New sampling frequency (Hz)

t\_sampled1 = np.arange(0, T, 1 / Fs\_new1)

signal\_sampled1 = np.sin(2 \* np.pi \* freq1 \* t\_sampled1)

# Compute frequency spectra

fft\_continuous1 = np.fft.fft(signal\_continuous1)

fft\_sampled1 = np.fft.fft(signal\_sampled1)

# Frequency axis for plotting

frequencies1 = np.fft.fftfreq(len(t\_continuous), 1 / Fs)

frequencies\_sampled1 = np.fft.fftfreq(len(t\_sampled1), 1 / Fs\_new1)

plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)

plt.plot(t\_continuous, signal\_continuous1)

plt.title('Continuous Signal')

plt.subplot(2, 2, 2)

plt.stem(t\_sampled1, signal\_sampled1, markerfmt='ro', basefmt='r-')

plt.title('Sampled Signal')

plt.subplot(2, 2, 3)

plt.plot(frequencies1, abs(fft\_continuous1))

plt.title('Frequency Spectrum (Continuous)')

plt.subplot(2, 2, 4)

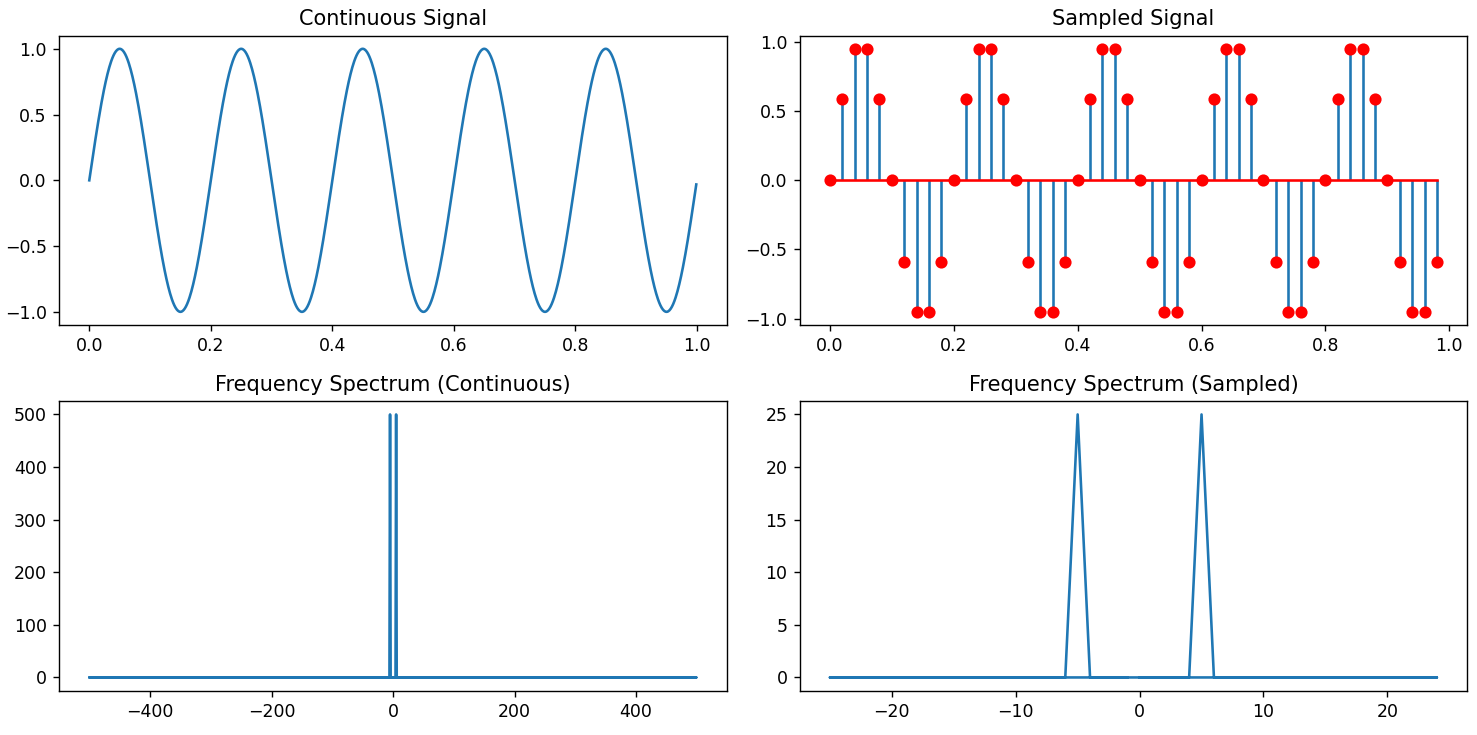
plt.plot(frequencies\_sampled1, abs(fft\_sampled1))

plt.title('Frequency Spectrum (Sampled)')

plt.tight\_layout()

plt.show()

**Output:**



**Tas2: Write a python program for continues time square wave signal.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define time parameters

Fs = 1000  # Sampling frequency (Hz)

T = 1.0    # Total time (seconds)

t\_continuous = np.linspace(0, T, int(T \* Fs), endpoint=False)

#Continuous-time square wave signal

freq2 = 10  # Frequency of the square wave (Hz)

signal\_continuous2 = np.sign(np.sin(2 \* np.pi \* freq2 \* t\_continuous))

# Sample the signal

Fs\_new2 = 50  # New sampling frequency (Hz)

t\_sampled2 = np.arange(0, T, 1 / Fs\_new2)

signal\_sampled2 = np.sign(np.sin(2 \* np.pi \* freq2 \* t\_sampled2))

# Compute frequency spectra

fft\_continuous2 = np.fft.fft(signal\_continuous2)

fft\_sampled2 = np.fft.fft(signal\_sampled2)

# Frequency axis for plotting

frequencies2 = np.fft.fftfreq(len(t\_continuous), 1 / Fs)

frequencies\_sampled2 = np.fft.fftfreq(len(t\_sampled2), 1 / Fs\_new2)

# Plot the results

plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)

plt.plot(t\_continuous, signal\_continuous2)

plt.title('Continuous Signal')

plt.subplot(2, 2, 2)

plt.stem(t\_sampled2, signal\_sampled2, markerfmt='ro', basefmt='r-')

plt.title('Sampled Signal')

plt.subplot(2, 2, 3)

plt.plot(frequencies2, abs(fft\_continuous2))

plt.title('Frequency Spectrum (Continuous)')

plt.subplot(2, 2, 4)

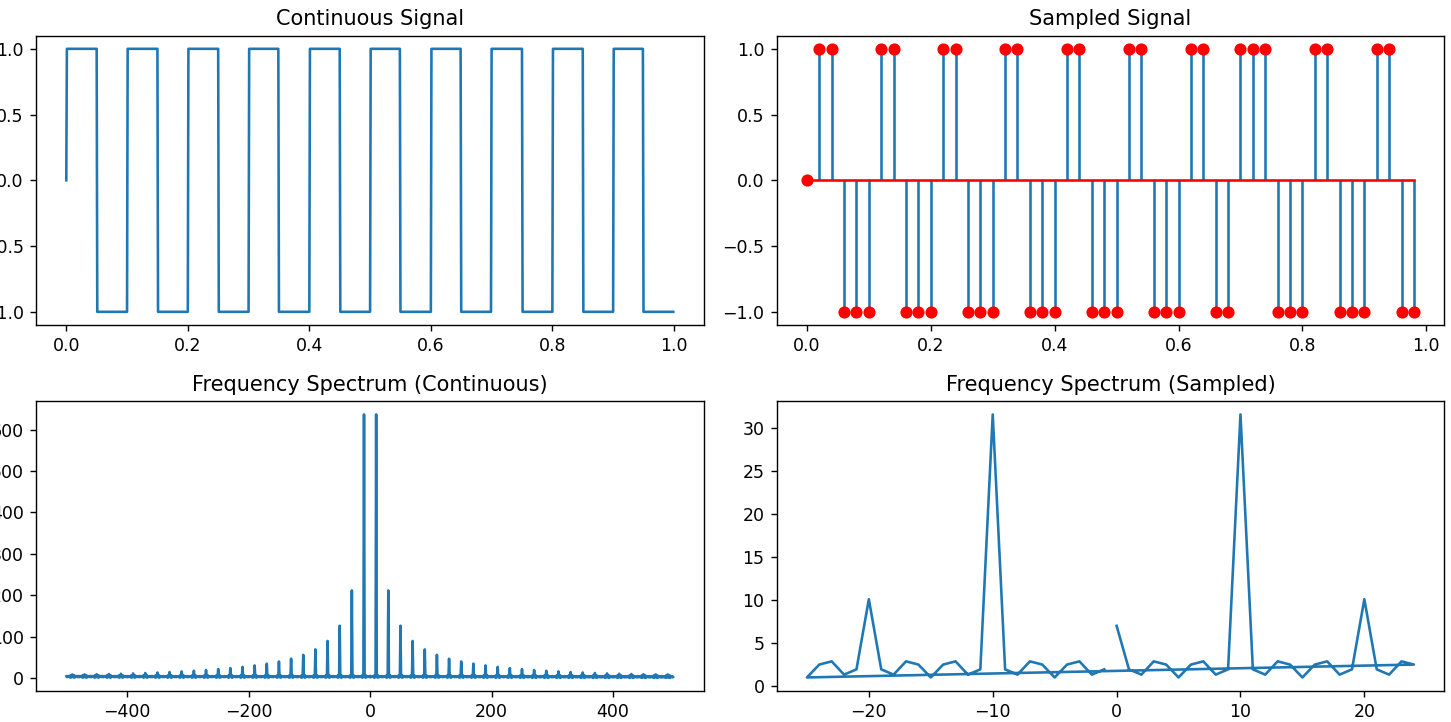
plt.plot(frequencies\_sampled2, abs(fft\_sampled2))

plt.title('Frequency Spectrum (Sampled)')

plt.tight\_layout()

plt.show()

**Output:**



**Task3: Write a python program for continues time gaussain signal .**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define time parameters

Fs = 1000  # Sampling frequency (Hz)

T = 1.0    # Total time (seconds)

t\_continuous = np.linspace(0, T, int(T \* Fs), endpoint=False)

#  Continuous-time Gaussian signal

mu = 0.5  # Mean of the Gaussian

sigma = 0.1  # Standard deviation of the Gaussian

signal\_continuous3 = np.exp(-(t\_continuous - mu)\*\*2 / (2 \* sigma\*\*2))

# Sample the signal

Fs\_new3 = 200  # New sampling frequency (Hz)

t\_sampled3 = np.arange(0, T, 1 / Fs\_new3)

signal\_sampled3 = np.exp(-(t\_sampled3 - mu)\*\*2 / (2 \* sigma\*\*2))

# Compute frequency spectra

fft\_continuous3 = np.fft.fft(signal\_continuous3)

fft\_sampled3 = np.fft.fft(signal\_sampled3)

# Frequency axis for plotting

frequencies3 = np.fft.fftfreq(len(t\_continuous), 1 / Fs)

frequencies\_sampled3 = np.fft.fftfreq(len(t\_sampled3), 1 / Fs\_new3)

# Plot the results

plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)

plt.plot(t\_continuous, signal\_continuous3)

plt.title('Continuous Signal')

plt.subplot(2, 2, 2)

plt.stem(t\_sampled3, signal\_sampled3, markerfmt='ro', basefmt='r-')

plt.title('Sampled Signal')

plt.subplot(2, 2, 3)

plt.plot(frequencies3, abs(fft\_continuous3))

plt.title('Frequency Spectrum (Continuous)')

plt.subplot(2, 2, 4)

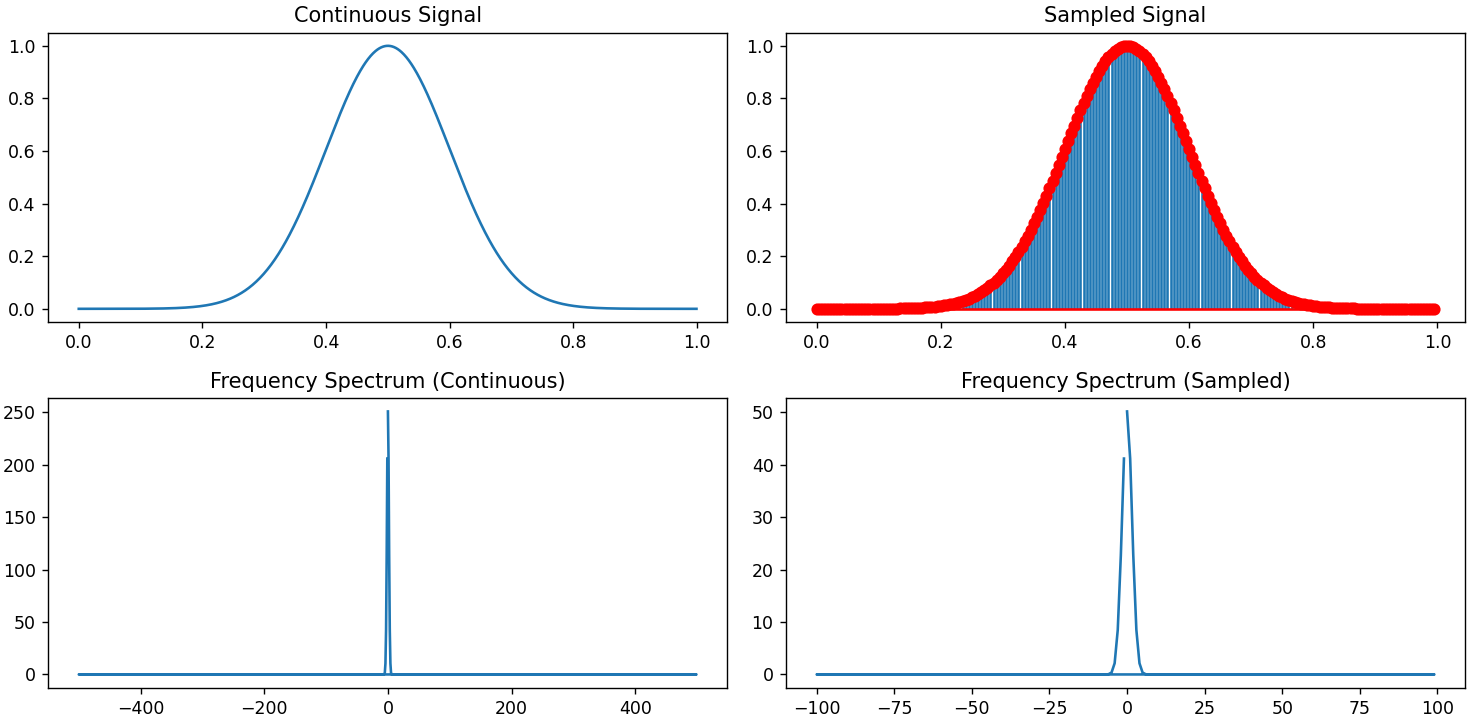
plt.plot(frequencies\_sampled3, abs(fft\_sampled3))

plt.title('Frequency Spectrum (Sampled)')

plt.tight\_layout()

plt.show()

**Output:**



**Task4: Write a python program for continuous time triangular signal .**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define time parameters

Fs = 1000  # Sampling frequency (Hz)

T = 1.0    # Total time (seconds)

t\_continuous = np.linspace(0, T, int(T \* Fs), endpoint=False)

# Example 4: Continuous-time triangular signal

signal\_continuous4 = np.abs(t\_continuous - T / 2) / (T / 2)

# Sample the signal

Fs\_new4 = 100  # New sampling frequency (Hz)

t\_sampled4 = np.arange(0, T, 1 / Fs\_new4)

signal\_sampled4 = np.abs(t\_sampled4 - T / 2) / (T / 2)

# Compute frequency spectra

fft\_continuous4 = np.fft.fft(signal\_continuous4)

fft\_sampled4 = np.fft.fft(signal\_sampled4)

# Frequency axis for plotting

frequencies4 = np.fft.fftfreq(len(t\_continuous), 1 / Fs)

frequencies\_sampled4 = np.fft.fftfreq(len(t\_sampled4), 1 / Fs\_new4)

# Plot the results

plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)

plt.plot(t\_continuous, signal\_continuous4)

plt.title('Continuous Signal')

plt.subplot(2, 2, 2)

plt.stem(t\_sampled4, signal\_sampled4, markerfmt='ro', basefmt='r-')

plt.title('Sampled Signal')

plt.subplot(2, 2, 3)

plt.plot(frequencies4, abs(fft\_continuous4))

plt.title('Frequency Spectrum (Continuous)')

plt.subplot(2, 2, 4)

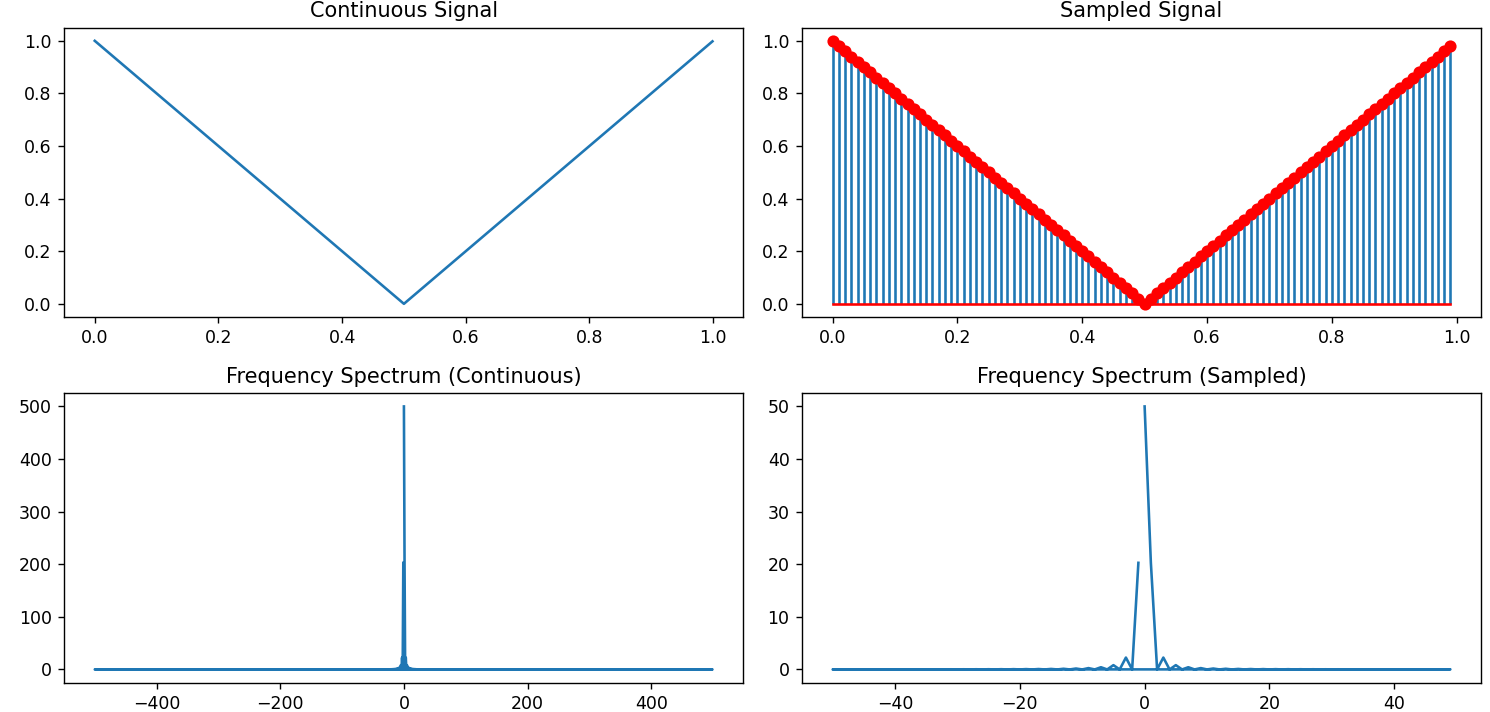
plt.plot(frequencies\_sampled4, abs(fft\_sampled4))

plt.title('Frequency Spectrum (Sampled)')

plt.tight\_layout()

plt.show()

**Output:**



**Task5: Write a python program for continuous time impulse signal.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define time parameters

Fs = 1000  # Sampling frequency (Hz)

T = 1.0    # Total time (seconds)

t\_continuous = np.linspace(0, T, int(T \* Fs), endpoint=False)

#Continuous-time impulse signal

signal\_continuous5 = np.zeros\_like(t\_continuous)

signal\_continuous5[len(t\_continuous) // 2] = 1  # Impulse at the midpoint

# Sample the signal

Fs\_new5 = 1000  # New sampling frequency (Hz)

t\_sampled5 = np.arange(0, T, 1 / Fs\_new5)

signal\_sampled5 = np.zeros\_like(t\_sampled5)

signal\_sampled5[len(t\_sampled5) // 2] = 1  # Impulse at the midpoint

# Compute frequency spectra

fft\_continuous5 = np.fft.fft(signal\_continuous5)

fft\_sampled5 = np.fft.fft(signal\_sampled5)

# Frequency axis for plotting

frequencies5 = np.fft.fftfreq(len(t\_continuous), 1 / Fs)

frequencies\_sampled5 = np.fft.fftfreq(len(t\_sampled5), 1 / Fs\_new5)

# Plot the results

plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)

plt.plot(t\_continuous, signal\_continuous5)

plt.title('Continuous Signal')

plt.subplot(2, 2, 2)

plt.stem(t\_sampled5, signal\_sampled5, markerfmt='ro', basefmt='r-')

plt.title('Sampled Signal')

plt.subplot(2, 2, 3)

plt.plot(frequencies5, abs(fft\_continuous5))

plt.title('Frequency Spectrum (Continuous)')

plt.subplot(2, 2, 4)

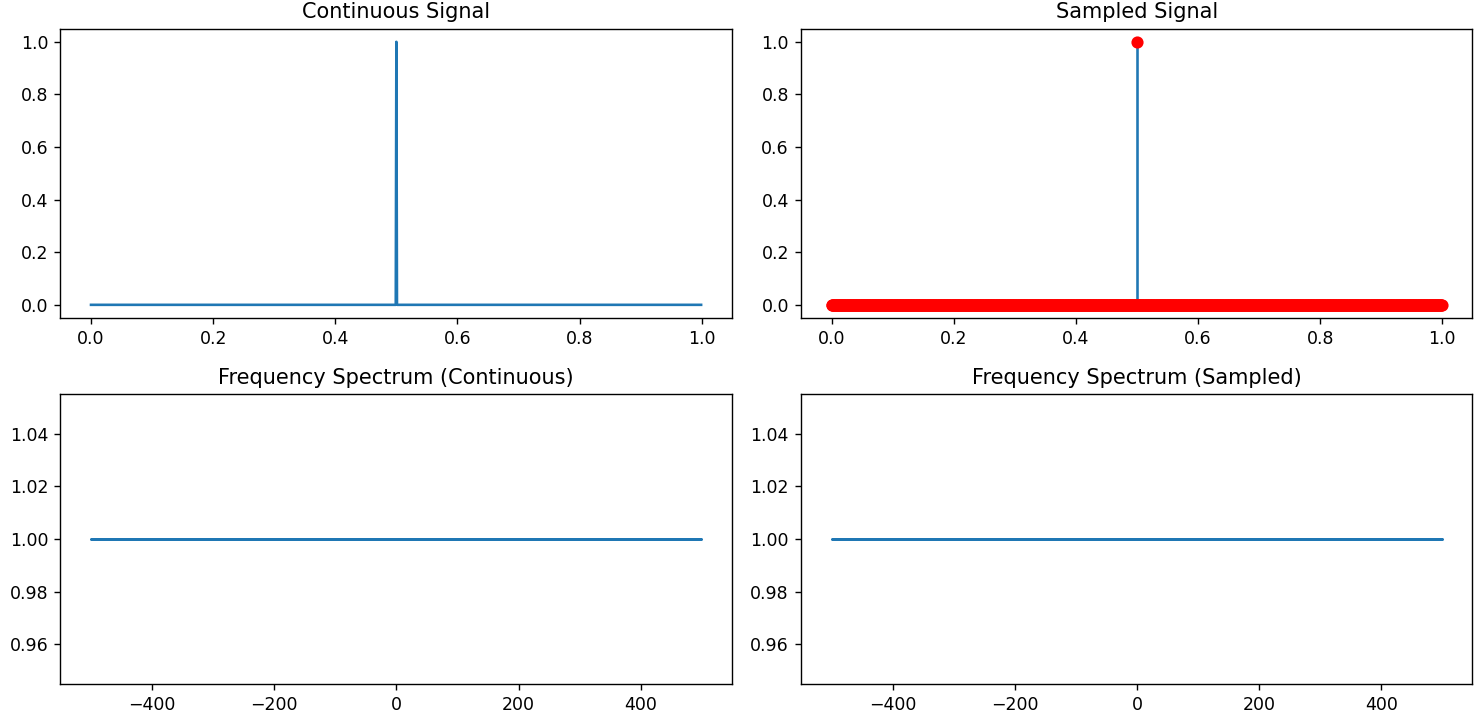
plt.plot(frequencies\_sampled5, abs(fft\_sampled5))

plt.title('Frequency Spectrum (Sampled)')

plt.tight\_layout()

plt.show()

**Output:**



**Lab 17**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- To understand image processing in python with Matplotlib.**

**Introduction:- Matplotlib is a low level graph plotting library in python that serves as a visualization utility.Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.**

**Task1: Write a python program to draw two points at different position.**

**Python Code:**

import sys

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

import numpy as np

xpoints = np.array([1, 8])

ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints, 'o')

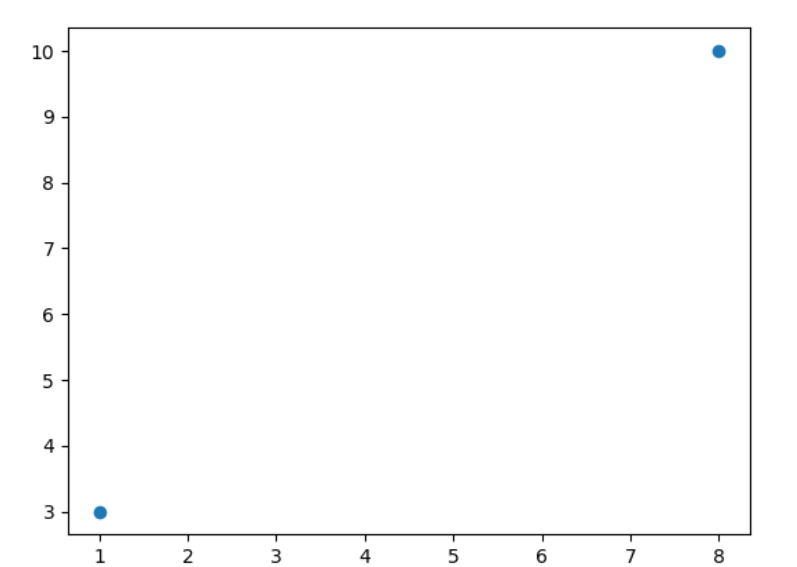
plt.show()

#Two  lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

**Output:**



**Tas2: Write a python program to draw a line diagram from position (0,0).**

**Python Code:**

import sys

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

import numpy as np

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

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

plt.plot(xpoints, ypoints)

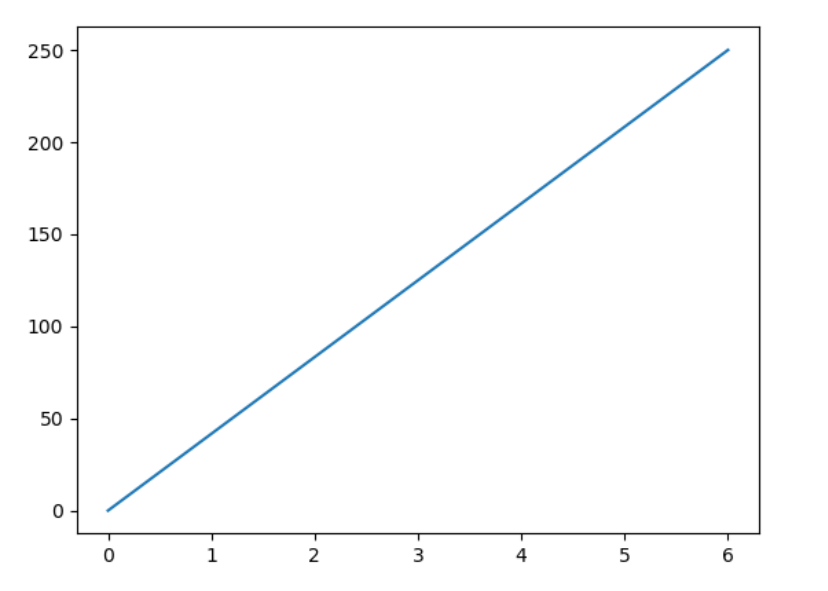
plt.show()

#Two  lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

**Output:**



**Task3: Write a python program for without using x-points, using defualt points.**

**Python Code:**

import sys

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)

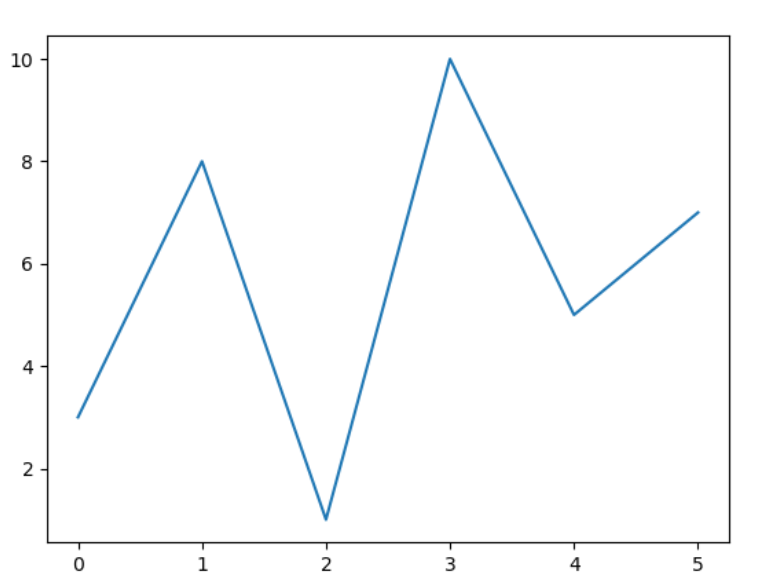
plt.show()

#Two  lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

**Output:**



**Task4: Write a python program to draw a different line style.**

**Python Code:**

import sys

import matplotlib

matplotlib.use('Agg')

import matplotlib.pyplot as plt

import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted')

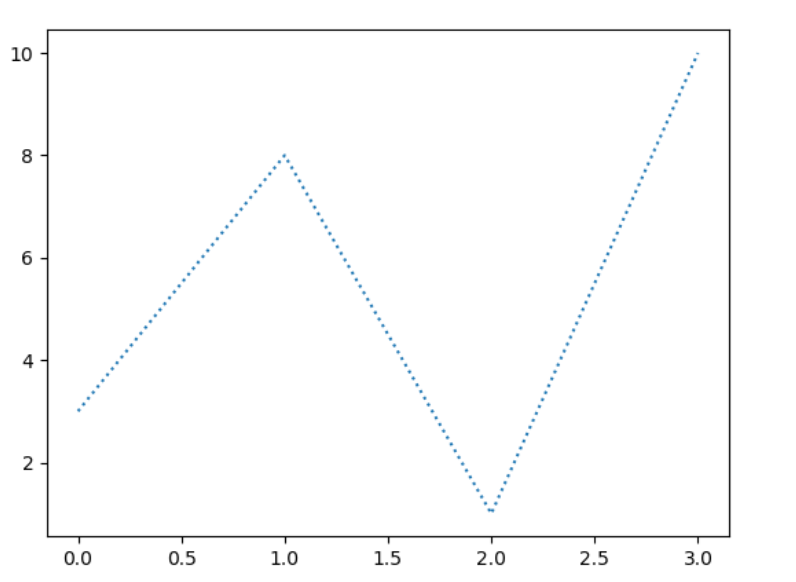
plt.show()

#Two  lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

**Output:**



**Task5: Write a python program with using the grid function.**

**Python Code:**

import sys

import matplotlib

matplotlib.use('Agg')

import numpy as np

import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])

y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.title("Sports Watch Data")

plt.xlabel("Average Pulse")

plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid()

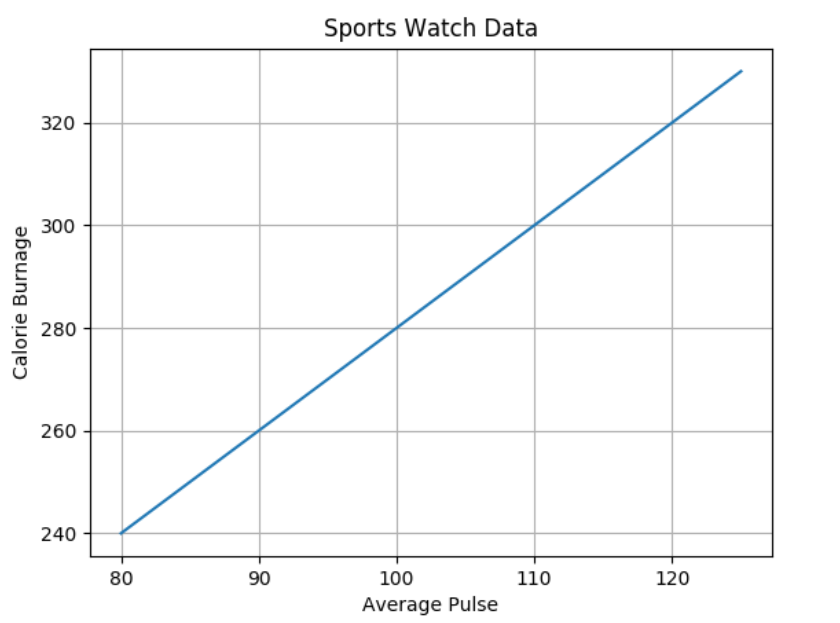
plt.show()

#Two  lines to make our compiler able to draw:

plt.savefig(sys.stdout.buffer)

sys.stdout.flush()

**Output:**



**Lab 18**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Simulate correlation and convolution operation discrete time sequences.**

**Introduction:- Correlation and convolution are fundamental operations in signal processing and mathematics, particularly in the context of discrete-time sequences. These operations are used to analyze, process, and manipulate signals in various applications, including image processing, audio processing, and communication systems.**

**Task1: Write a python program for simple correlation.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define two discrete-time sequences

sequence1 = np.array([1, 2, 1, 0])

sequence2 = np.array([1, -1, 2, -2])

#  Correlation

correlation\_result = np.correlate(sequence1, sequence2, mode='full')

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

plt.stem(sequence1, use\_line\_collection=True)

plt.title('Sequence 1')

plt.subplot(1, 3, 2)

plt.stem(sequence2, use\_line\_collection=True)

plt.title('Sequence 2')

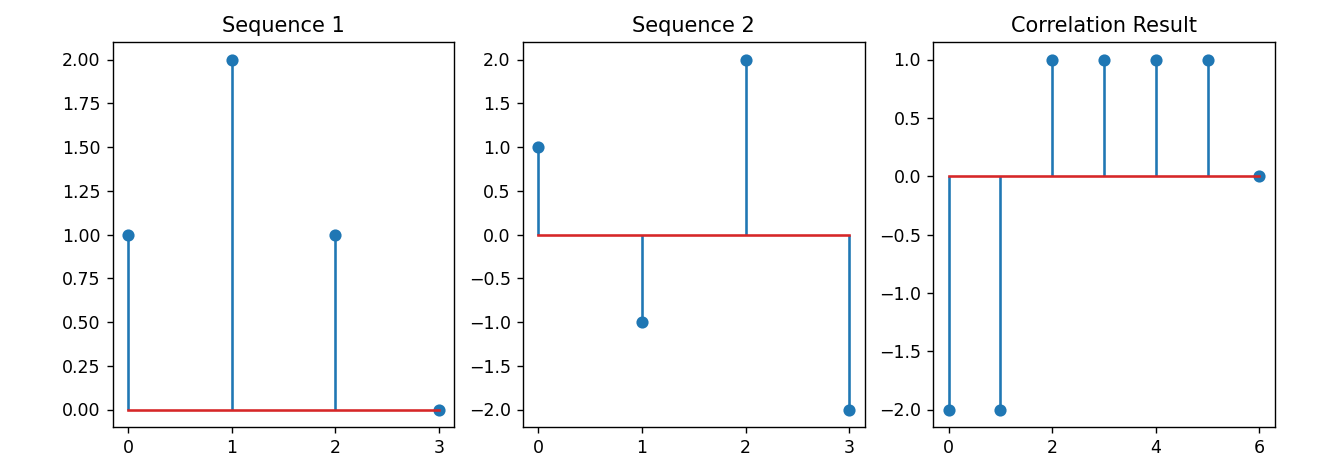
plt.subplot(1, 3, 3)

plt.stem(correlation\_result, use\_line\_collection=True)

plt.title('Correlation Result')

plt.show()

**Output:**



**Tas2: Write a python program for simple convolution.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define two discrete-time sequences

sequence1 = np.array([1, 2, 1, 0])

sequence2 = np.array([1, -1, 2, -2])

#Convolution

convolution\_result = np.convolve(sequence1, sequence2, mode='full')

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

plt.stem(sequence1, use\_line\_collection=True)

plt.title('Sequence 1')

plt.subplot(1, 3, 2)

plt.stem(sequence2, use\_line\_collection=True)

plt.title('Sequence 2')

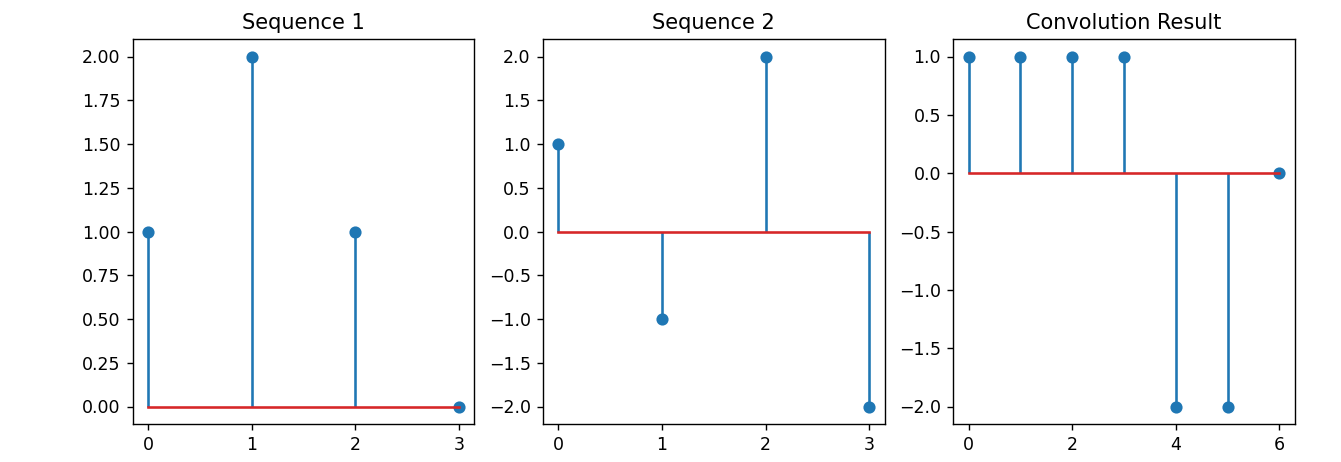
plt.subplot(1, 3, 3)

plt.stem(convolution\_result, use\_line\_collection=True)

plt.title('Convolution Result')

plt.show()

**Output:**



**Task3: Write a python program for correlation with different sequences.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define two discrete-time sequences

sequence1 = np.array([1, 2, 1, 0])

sequence2 = np.array([1, -1, 2, -2])

#Correlation with different sequences

sequence3 = np.array([1, 0, 0, 0, 1])

correlation\_result2 = np.correlate(sequence1, sequence3, mode='full')

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

plt.stem(sequence1, use\_line\_collection=True)

plt.title('Sequence 1')

plt.subplot(1, 3, 2)

plt.stem(sequence3, use\_line\_collection=True)

plt.title('Sequence 3')

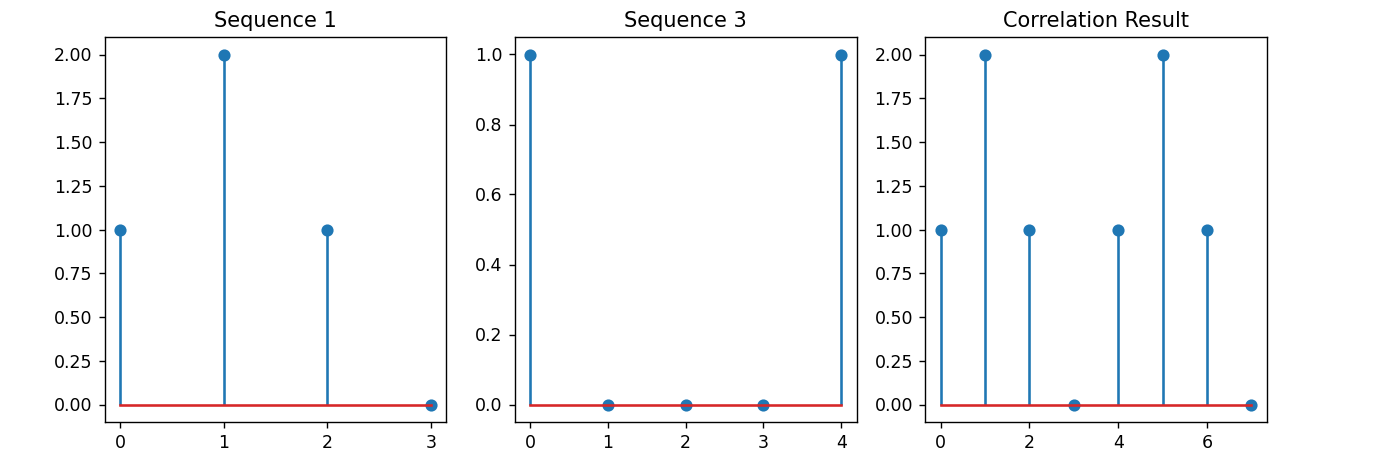
plt.subplot(1, 3, 3)

plt.stem(correlation\_result2, use\_line\_collection=True)

plt.title('Correlation Result')

plt.show()

**Output:**



**Task4: Write a python program for convolution with different sequences.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define two discrete-time sequences

sequence1 = np.array([1, 2, 1, 0])

sequence2 = np.array([1, -1, 2, -2])

#Convolution with different sequences

sequence4 = np.array([1, -1])

convolution\_result2 = np.convolve(sequence1, sequence4, mode='full')

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

plt.stem(sequence1, use\_line\_collection=True)

plt.title('Sequence 1')

plt.subplot(1, 3, 2)

plt.stem(sequence4, use\_line\_collection=True)

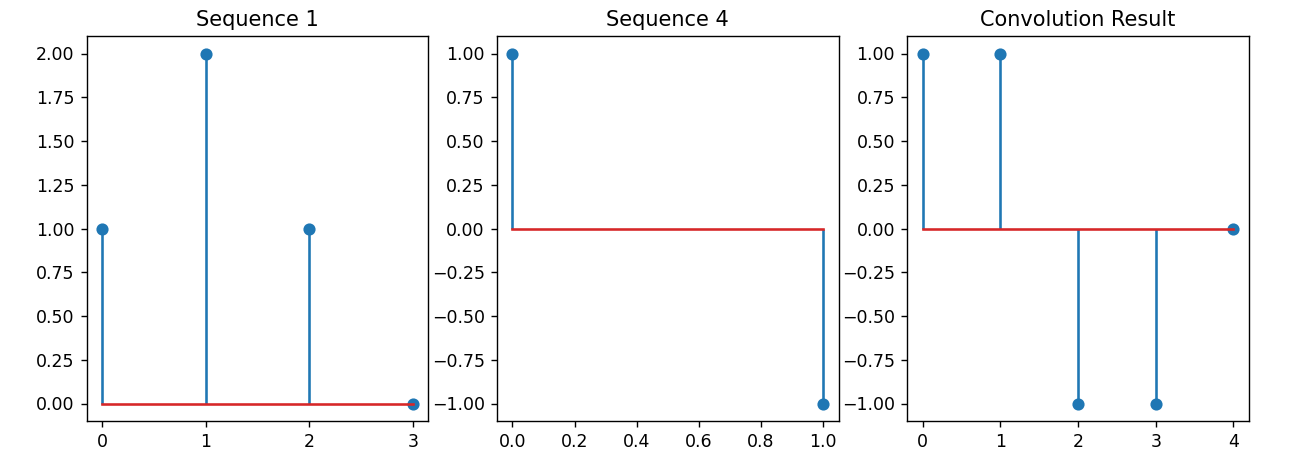
plt.title('Sequence 4')

plt.subplot(1, 3, 3)

plt.stem(convolution\_result2, use\_line\_collection=True)

plt.title('Convolution Result')

plt.show()

**Output:**

**Task5: Write a python program for correlation with a shift sequence.**

**Python Code:**

import numpy as np

import matplotlib.pyplot as plt

# Define two discrete-time sequences

sequence1 = np.array([1, 2, 1, 0])

sequence2 = np.array([1, -1, 2, -2])

#Correlation with a shifted sequence

shifted\_sequence = np.roll(sequence2, shift=2)  # Shift sequence2 by 2 positions

correlation\_result3 = np.correlate(sequence1, shifted\_sequence, mode='full')

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

plt.stem(sequence1, use\_line\_collection=True)

plt.title('Sequence 1')

plt.subplot(1, 3, 2)

plt.stem(shifted\_sequence, use\_line\_collection=True)

plt.title('Shifted Sequence 2')

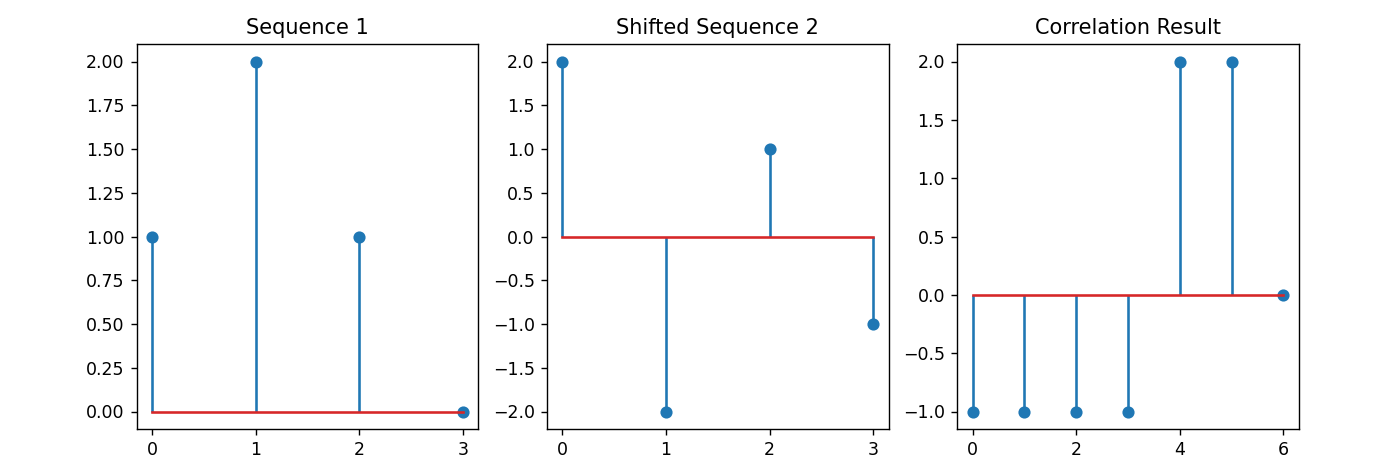
plt.subplot(1, 3, 3)

plt.stem(correlation\_result3, use\_line\_collection=True)

plt.title('Correlation Result')

plt.show()

**Output:**



**Lab 19**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Write script named copyfile.py. This script should prompt the user for the names of two text file. The contents of the first the second file.**

**Introduction:- As our program grows bigger, it may contain many lines of code. Instead of putting everything in a single file, we can use modules to separate codes in separate files as per their functionality. This makes our code organized and easier to maintain. Module is a file that contains code to perform a specific task. A module may contain variables, functions, classes etc.**

**Task1: Write a python program using file handling to read and append.**

**Python Code:**

# open both files

with open('first.txt','r') as firstfile, open('second.txt','a') as secondfile:

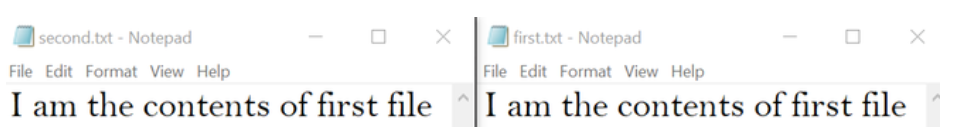
    # read content from first file

    for line in firstfile:

            # append content to second file

            secondfile.write(line)

**Output:**



**Task2: Write a python program using to read and write.**

**Python Code:**

# open both files

with open('first.txt','r') as firstfile, open('second.txt','w') as secondfile:

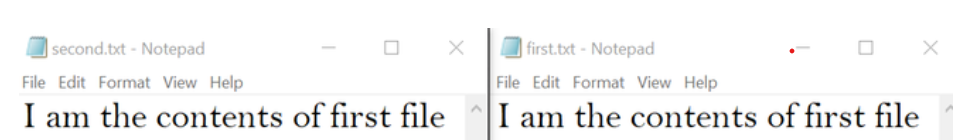
    # read content from first file

**for** line **in** firstfile:

             # write content to second file

             secondfile.write(line)

**Output:**



**Task3: Write a python program to convert temperature into different measures.**

**Python Code:**

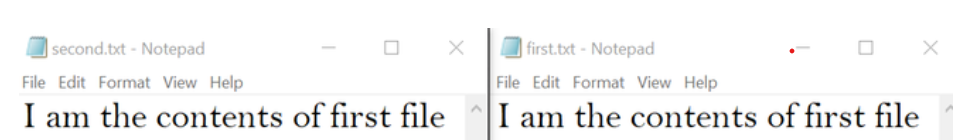
# import module

import shutil

# use copyfile()

shutil.copyfile('first.txt','second.txt')

**Output:**



**Lab 20**

**Name: - Aryan Dilipbhai Langhanoja**

**Date:**

**Enrollment No: 92200133030**

**CO1: To write, test, and debug simple Python programs**

**CO2: To implement Python programs with conditional, loops and functions**

**AIM:- Write a GUI program to create Tic-tac-toe in python.**

**Introduction:- Tic-tac-toe (American English), noughts and crosses (British English), or Xs and Os is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.tkinter Python library is used to create the GUI. Two options are available to play the game, along with the system or with another player.**

**Task1: Write a python program to create Tic-tac-toe game.**

**Python Code:**

import tkinter as tk

from tkinter import messagebox

# Initialize variables

current\_player = "X"

board = [""] \* 9

# Create the main game window

window = tk.Tk()

window.title("Tic-Tac-Toe")

# Function to handle button clicks

def button\_click(index):

global current\_player

if board[index] == "":

board[index] = current\_player

buttons[index].config(text=current\_player)

if check\_winner(current\_player):

messagebox.showinfo("Tic-Tac-Toe", f"Player {current\_player} wins!")

reset\_game()

elif "" not in board:

messagebox.showinfo("Tic-Tac-Toe", "It's a draw!")

reset\_game()

else:

current\_player = "O" if current\_player == "X" else "X

# Function to check for a winner

def check\_winner(player):

winning\_combinations = [

[0, 1, 2], [3, 4, 5], [6, 7, 8], # Rows

[0, 3, 6], [1, 4, 7], [2, 5, 8], # Columns

[0, 4, 8], [2, 4, 6] # Diagonals

]

for combo in winning\_combinations:

if board[combo[0]] == board[combo[1]] == board[combo[2]] == player:

return True

return False

# Function to reset the game

def reset\_game():

global board, current\_player

board = [""] \* 9

current\_player = "X"

for button in buttons:

button.config(text="", state="active")

# Create buttons for the game board

buttons = []

for i in range(9):

row = i // 3

col = i % 3

button = tk.Button(window, text="", width=10, height=3,

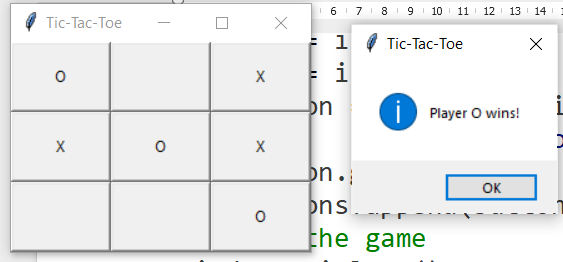
command=lambda i=i: button\_click(i))

button.grid(row=row, column=col)

buttons.append(button)

# Start the game

window.mainloop()

**Output:**

