**Sahakar Maharshi Bhausaheb Santuji Thorat College Sangamner**

**Remark**

**Demonstrator’s Signature**

**Date:-**

**/**

**/20**

**DEPARTMENT OF COMPUTER SCIENCE**

**Sub : Mathematics**

**Name:-\_Gorde Yash Somnath Roll.No:- Date:- Title of the expt:- Slip no 3 Page.no:- Class:-BCS**

**Q1.Attemp any one of the following**

**a ) Using Python plot the graph of function f(x)=cos(x) on the interval [0,2π]**

->

import numpy as np

import matplotlib.pyplot as plt

def f(x):

return np.cos(x)

x = np.linspace(0, 2\*np.pi, 100) y = f(x)

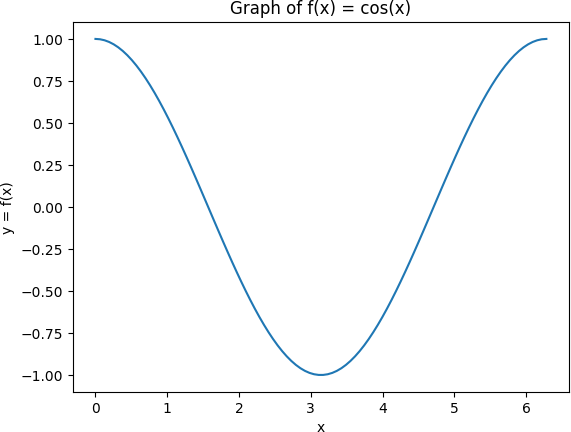
plt.plot(x, y)

plt.title("Graph of f(x) = cos(x)")

plt.xlabel("x")

plt.ylabel("y = f(x)") plt.show()

output :



# b ) Following is the information of students participating in varios games in a school,Represnet it by Bar graph with bar Width of 0.7 inches

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Game** | **Cricket** | **Football** | **Hockey** | **Chess** | **Tennis** |
| **Number of student** | **65** | **30** | **54** | **10** | **20** |

->

import matplotlib.pyplot as plt

games = ['Cricket', 'Football', 'Hockey', 'Chess', 'Tennis'] students = [65, 30, 54, 10, 20]

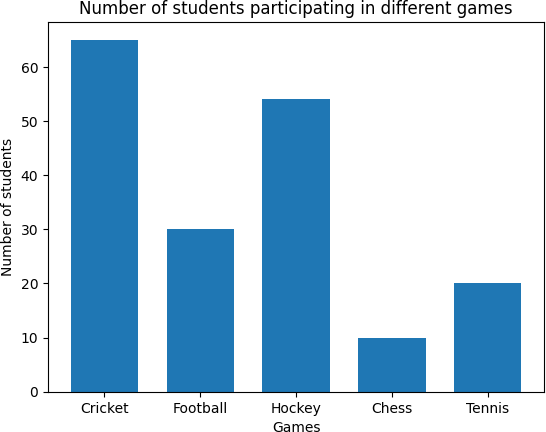
width = 0.7

plt.bar(games, students, width)

plt.title('Number of students participating in different games') plt.xlabel('Games')

plt.ylabel('Number of students') plt.show()

output:



# C ) Write a Python Program to generate 3D plot of the function z=sin x +cos y in - 10<x,y<10.

->

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D def f(x, y):

return np.sin(x) + np.cos(y)

x = np.arange(-10, 10, 0.1)

y = np.arange(-10, 10, 0.1) X, Y = np.meshgrid(x, y) Z = f(X, Y)

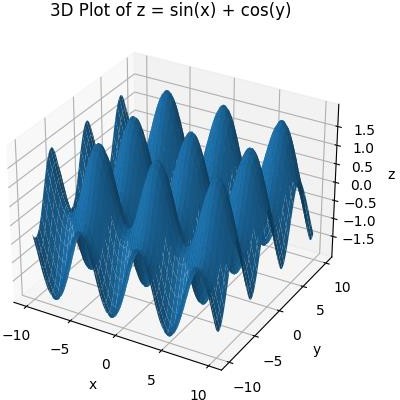
fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d') ax.plot\_surface(X, Y, Z)

ax.set\_title('3D Plot of z = sin(x) + cos(y)') ax.set\_xlabel('x')

ax.set\_ylabel('y') ax.set\_zlabel('z') plt.show()

output :



# Q2 ) Attempt any TWO of the following

1. **Write a python program to reflect the line segment joining the points A[5,3] & B[1,4] through the line y=x+1**

-

import numpy as np

A = np.array([5, 3])

B = np.array([1, 4]) m = 1

b = 1

midpoint = (A + B) / 2 slope = -1 / m

y\_intercept = midpoint[1] - slope \* midpoint[0]

x\_intersect = (b - y\_intercept) / (slope - m) y\_intersect = slope \* x\_intersect + y\_intercept P = np.array([x\_intersect, y\_intersect]) A\_reflected = 2 \* P - A

B\_reflected = 2 \* P - B

print("The reflection of A through the line y=x+1 is:", A\_reflected) print("The reflection of B through the line y=x+1 is:", B\_reflected)

output :

The reflection of A through the line y=x+1 is: [0.5 4.5] The reflection of B through the line y=x+1 is: [4.5 3.5]

# If the points A[2,1],B[4,-1] is transformed by the transformation matrix

**[T]=**𝟏 𝟐

𝟐 𝟏

# , then Using python ,find the euquation of transformation line



import numpy as np

A = np.array([2, 1])

B = np.array([4, -1])

T = np.array([[1, 2], [2, 1]])

A\_transformed = np.dot(T, A) B\_transformed = np.dot(T, B)

m = (B\_transformed[1] - A\_transformed[1]) / (B\_transformed[0] - A\_transformed[0]) b = A\_transformed[1] - m \* A\_transformed[0]

transformation\_line = f"y = {m}x + {b}"

print("Equation of transformation line:", transformation\_line)

output :

Equation of transformation line: y = -1.0x + 9.0

**C ) Generate line segment having endpoint (0,0) and (10,10) find midpoint of line segment.**

-

import numpy as np

A = np.array([0, 0])

B = np.array([10, 10]) midpoint = (A + B) / 2

print("Midpoint of the line segment:", midpoint)

output :

Midpoint of the line segment: [5. 5.]

**Q3) Attempt any of the following**

1. **Attempt any one of the following**

**i)Write A python program to solve the following LPP Min Z=3.5x+2y**

**Subject x+y≥5**

**x≥5 y≤2 x,y≥0**

-

import numpy as np

from scipy.optimize import linprog c = np.array([3.5, 2])

A = np.array([[-1, -1],

[-1, 0],

[0, 1]])

b = np.array([-5, -5, 2]) x\_bounds = (0, None) y\_bounds = (0, None)

result = linprog(c=c, A\_ub=A, b\_ub=b, bounds=[x\_bounds, y\_bounds]) print("Optimal solution:", result.x)

print("Optimal value:", result.fun)

output :

Optimal solution: [5. 0.]

Optimal value: 17.5

**Ii )Write a python program to display the following LPP by using pulp module and simplex method.Find optimal solution if exist**

**Max z=3x1 +5x2+4x3 Subject to 2x1+3x2≤8**

**2x2+5x3≤10**

**3x1+2x2+4x3≤15**

**x1,x2,x3 ≥0**

-

import pulp as lp

prob = lp.LpProblem("LP problem", lp.LpMaximize) x1 = lp.LpVariable("x1", lowBound=0)

x2 = lp.LpVariable("x2", lowBound=0) x3 = lp.LpVariable("x3", lowBound=0)

prob += 3\*x1 + 5\*x2 + 4\*x3 prob += 2\*x1 + 3\*x2 <= 8 prob += 2\*x2 + 5\*x3 <= 10

prob += 3\*x1 + 2\*x2 + 4\*x3 <=15 prob.solve(lp.solvers.PULP\_CBC\_CMD(msg=0)) print("Optimal solution:")

print("x1 =", lp.value(x1))

print("x2 =", lp.value(x2))

print("x3 =", lp.value(x3))

print("Optimal value: z =", lp.value(prob.objective))

1. **Attempt any ONE of the Following**

**I ) Apply Python program in each of the following transformation on the point P[4,-2]**

* 1. **Reflection through the Y-axis**
  2. **Scaling in X-cordinate by Factor 3**
  3. **Scaling in Y-cordinate by factor 2.5**
  4. **Reflection through the line y=-x**

-

P = [4, -2]

P\_reflect\_y = [-P[0], P[1]]

print("Reflection through the Y-axis:", P\_reflect\_y) P\_scale\_x = [3\*P[0], P[1]]

print("Scaling in X-coordinate by Factor 3:", P\_scale\_x) P\_scale\_y = [P[0], 2.5\*P[1]]

print("Scaling in Y-coordinate by Factor 2.5:", P\_scale\_y)

P\_reflect\_line = [-P[1], -P[0]]

print("Reflection through the line y=-x:", P\_reflect\_line)

output :

Reflection through the Y-axis: [-4, -2]

Scaling in X-coordinate by Factor 3: [12, -2]

Scaling in Y-coordinate by Factor 2.5: [4, -5.0] Reflection through the line y=-x: [2, -4]

**II ) Find the combined transformation of the line segment between the points A[2,-1] & B[5,4] by using Python program for the following sequence of transformation**

**I ) Rotation about origin through an angel π Ii ) Scaling in X-cordinate by 3 unit**

**III ) Shearing in X direction by 6 units**

**IV ) Reflection through the line y=x**



import numpy as np

AB = np.array([[2, -1], [5, 4]])

theta = np.pi

R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])

AB\_rotated = np.matmul(R, AB.T).T

print("After rotation about origin through an angle pi:", AB\_rotated) S\_x = np.array([[3, 0], [0, 1]])

AB\_scaled\_x = np.matmul(S\_x, AB.T).T

print("After scaling in X-coordinate by 3 units:", AB\_scaled\_x) Sh\_x = np.array([[1, 6], [0, 1]])

AB\_sheared\_x = np.matmul(Sh\_x, AB.T).T

print("After shearing in X-direction by 6 units:", AB\_sheared\_x) R\_yx = np.array([[0, 1], [1, 0]])

AB\_reflected\_yx = np.matmul(R\_yx, AB.T).T

print("After reflection through the line y=x:", AB\_reflected\_yx)

AB\_transformed = np.matmul(R\_yx, np.matmul(Sh\_x, np.matmul(S\_x, np.matmul(R, AB.T)))).T

print("After combining all transformations:", AB\_transformed)