Sahakar Maharshi Bhausaheb Santuji Thorat College Sangamner

DEPARTMENT OF COMPUTER SCIENCE

MATHEMATICS

Remark
Demonstrators Signature
Date :- / /2023

Name :- Prem Vijay Vajare **Batch No. :- D** **Roll No:- 75** **Date:-** / /2023

Title of the:- Practical 14 **Expt. No .** 14 **Class :-** S.Y.BCS

Q.1)Write a Python program to plot 2D graph of the functions $f(x) = x^2$ and $g(x) = x^3$ in [-1, 1]

Syntax:

```
import matplotlib.pyplot as plt

import numpy as np

def f(x):

    return x**2

def g(x):

    return x**3

# Generate x values in the range [-1, 1]

x = np.linspace(-1, 1, 100)

# Calculate y values for f(x) and g(x)

y_f = f(x)

y_g = g(x)

# Create a figure and axes

fig, ax = plt.subplots()

# Plot f(x) and g(x) on the same graph

ax.plot(x, y_f, label='f(x) = x^2')

ax.plot(x, y_g, label='g(x) = x^3')

# Add labels and legend

ax.set_xlabel('x')

ax.set_ylabel('y')

ax.legend()
```

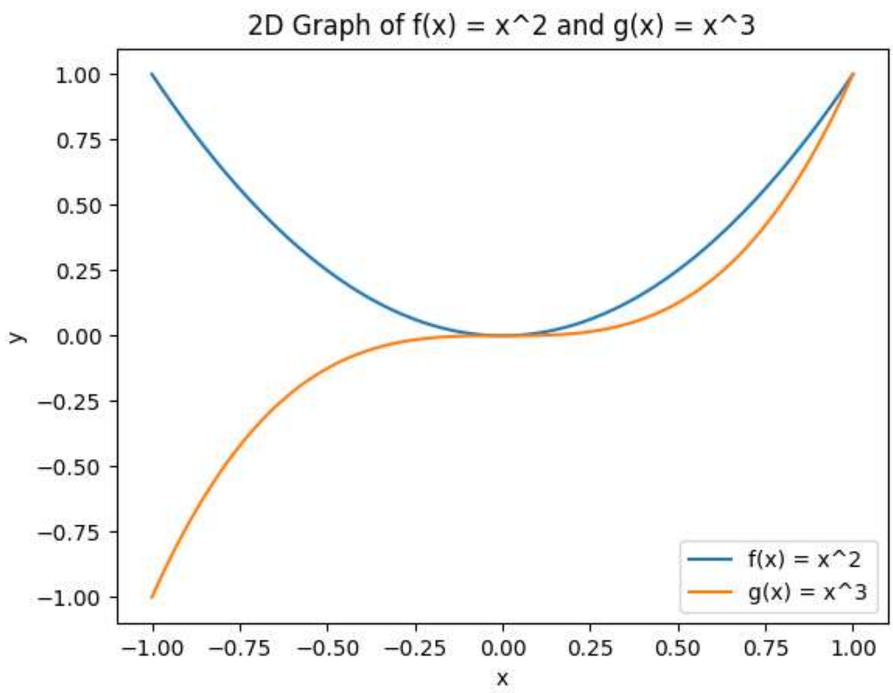
Set title

ax.set_title('2D Graph of f(x) = x^2 and g(x) = x^3')

Show the plot

plt.show()

OUTPUT:



Q.2) Write a Python program to plot 3D graph of the function f(x) = e**x**3 in [-5, 5] with green dashed points line with upward pointing triangle.

Syntax:

import numpy as np

import matplotlib.pyplot as plt

Generate x values

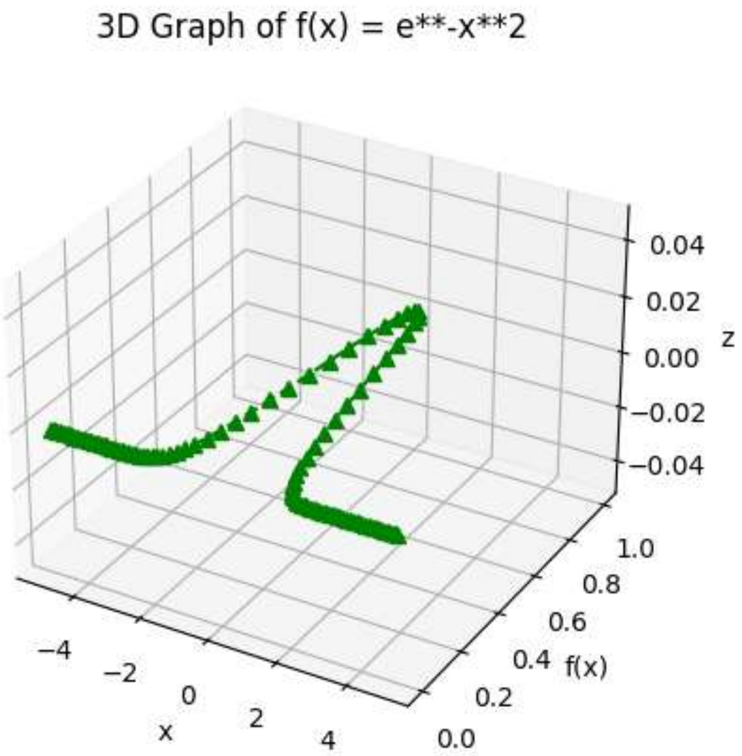
x = np.linspace(-5, 5, 100)

Compute y values using the given function

y = np.exp(-x**2)

```
# Create 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# Plot the points with green dashed line and upward-pointing triangles
ax.plot(x, y, np.zeros_like(x), linestyle='dashed', color='green', marker='^')
# Set labels for axes
ax.set_xlabel('x')
ax.set_ylabel('f(x)')
ax.set_zlabel('z')
# Set title for the plot
ax.set_title('3D Graph of f(x) = e**-x**2')
# Show the plot
plt.show()
```

OUTPUT:

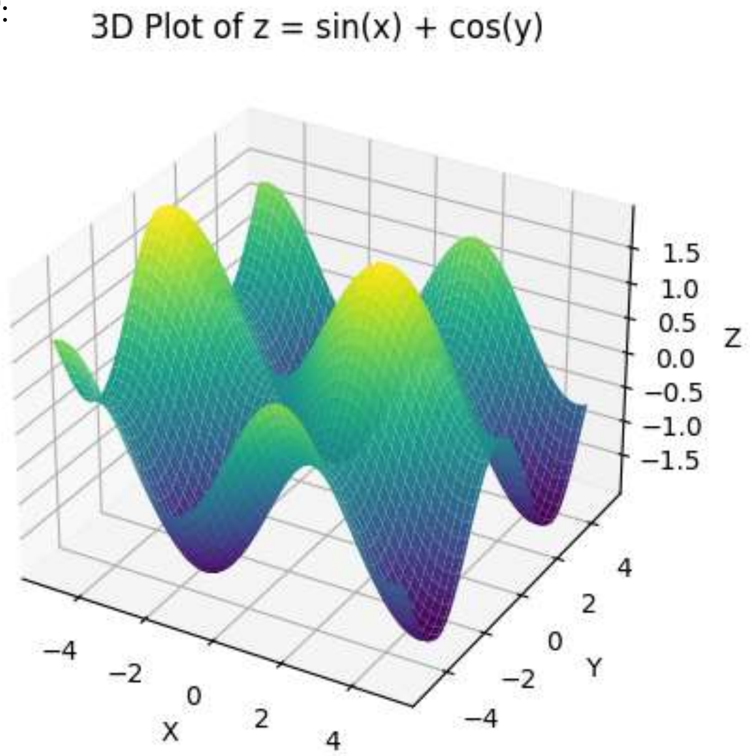


Q.3) Write a Python program to generate 3D plot of the functions z = sin x +cos y in -5< x, y < 5.

Syntax:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# Generate data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(X) + np.cos(Y)
# Create 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D Plot of z = sin(x) + cos(y)')
plt.show()
```

OUTPUT:



Q.4) write a Python program to reflect the line segment joining the points A[5, 3] and B[l, 4] through the line y = x + 1.

Syntax:

```
import numpy as np
# Define the points A and B
A = np.array([5, 3])
B = np.array([1, 4])
# Define the equation of the reflecting line
def reflect(line, point):
    m = line[0]
    c = line[1]
    x, y = point
    x_reflect = (2 * m * (y - c) + x * (m ** 2 - 1)) / (m ** 2 + 1)
    y_reflect = (2 * m * x + y * (1 - m ** 2) + 2 * c) / (m ** 2 + 1)
    return np.array([x_reflect, y_reflect])
# Define the equation of the reflecting line y = x + 1
line = np.array([1, -1])
# Reflect points A and B through the reflecting line
A_reflected = reflect(line, A)
B_reflected = reflect(line, B)
# Print the reflected points
print("Reflected Point A':", A_reflected)
print("Reflected Point B':", B_reflected)
```

Output:

```
Reflected Point A': [4. 4.]
Reflected Point B': [5. 0.]
```

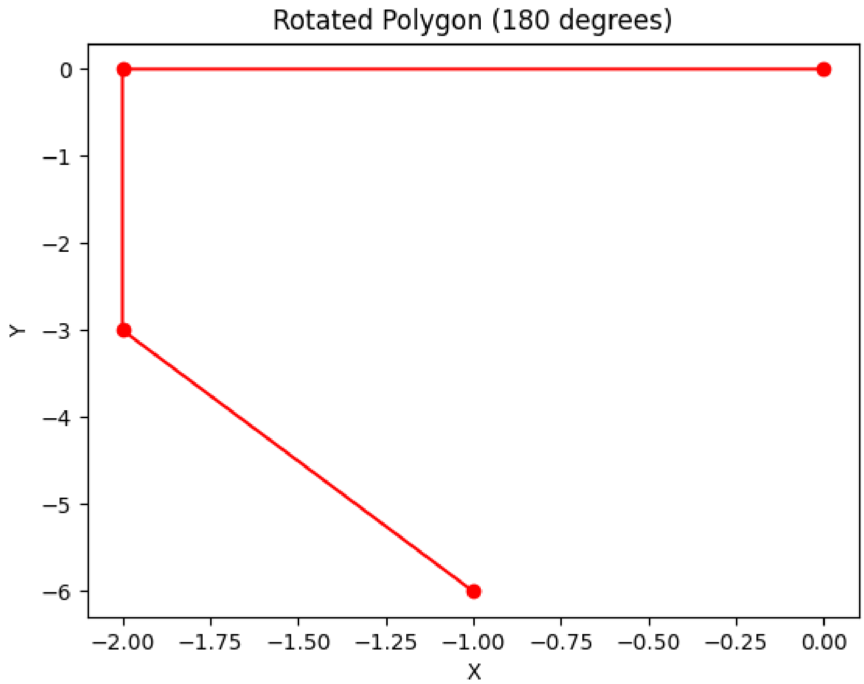
Q.5) Write a Python program to draw a polygon with vertices (0, 0), (2, 0), (2, 3) and (1, 6) and rotate it by 180° .

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Define the vertices of the polygon
vertices = np.array([[0, 0], [2, 0], [2, 3], [1, 6]])
```

```
# Plot the original polygon
plt.figure()
plt.plot(vertices[:, 0], vertices[:, 1], 'bo-')
plt.title('Original Polygon')
plt.xlabel('X')
plt.ylabel('Y')
# Define the rotation matrix for 180 degrees
theta = np.pi  # 180 degrees
rotation_matrix = np.array([[np.cos(theta), -np.sin(theta)],
                            [np.sin(theta), np.cos(theta)]])
# Apply rotation to the vertices
vertices_rotated = np.dot(vertices, rotation_matrix)
# Plot the rotated polygon
plt.figure()
plt.plot(vertices_rotated[:, 0], vertices_rotated[:, 1], 'ro-')
plt.title('Rotated Polygon (180 degrees)')
plt.xlabel('X')
plt.ylabel('Y')
# Show the plots
plt.show()
```

OUTPUT:



Q.6) Write a Python program to find the area and perimeter of the ABC, where A[0, 0] B[5, 0], C[3,3].

Synatx:

```
import numpy as np

# Define the vertices of the triangle

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

B = np.array([5, 0])

C = np.array([3, 3])

# Calculate the side lengths of the triangle

AB = np.linalg.norm(B - A)

BC = np.linalg.norm(C - B)

CA = np.linalg.norm(A - C)

# Calculate the semiperimeter

s = (AB + BC + CA) / 2

# Calculate the area using Heron's formula

area = np.sqrt(s * (s - AB) * (s - BC) * (s - CA))

# Calculate the perimeter

perimeter = AB + BC + CA

# Print the results

print("Triangle ABC:")

print("Side AB:", AB)

print("Side BC:", BC)

print("Side CA:", CA)

print("Area:", area)

print("Perimeter:", perimeter)
```

OUTPUT:

```
Triangle ABC:

Side AB: 5.0
```

Side BC: 3.605551275463989

Side CA: 4.242640687119285

Area: 7.5000000000000036

Perimeter: 12.848191962583275

Transformed Point A: [38. 10.]

Transformed Point B: [35. 8.]

Q.7) write a Python program to solve the following LPP

Max Z = 150x + 75y

Subjected to

4x + 6y <= 24

5x + 3y <= 15

x > 0 , y > 0

Syntax:

```
from pulp import *
# Create the LP problem as a maximization problem
problem = LpProblem("LPP", LpMaximize)
# Define the decision variables
x = LpVariable('x', lowBound=0, cat='Continuous')
y = LpVariable('y', lowBound=0, cat='Continuous')
# Define the objective function
problem += 150 * x + 75 * y, "Z"
# Define the constraints
problem += 4 * x + 6 * y <= 24, "Constraint1"
problem += 5 * x + 3 * y <= 15, "Constraint2"
# Solve the LP problem
problem.solve()
```

```
# Print the status of the solution
print("Status:", LpStatus[problem.status])
# Print the optimal values of x and y
print("Optimal x =", value(x))
print("Optimal y =", value(y))
# Print the optimal value of the objective function
print("Optimal Z =", value(problem.objective
```

OUTPUT:

Status: Optimal

Optimal x = 3.0

Optimal y = 0.0

Optimal Z = 450.0

Q.8) Write a python program to display the following LPP by using pulp module and simplex method. Find its optimal solution if exist.

Min Z = x+y
subject to
x => 6
y => 6
x + y <= 11
x=>0, y=>0

Syntax:

```
from pulp import *
# Create the LP problem as a minimization problem
problem = LpProblem("LPP", LpMinimize)
# Define the decision variables
x = LpVariable('x', lowBound=0, cat='Continuous')
y = LpVariable('y', lowBound=0, cat='Continuous')
# Define the objective function
problem += x + y, "Z"
# Define the constraints
problem += x >= 6, "Constraint1"
problem += y >= 6, "Constraint2"
problem += x + y <= 11, "Constraint3"
# Solve the LP problem using the simplex method
```

```
problem.solve(PULP_CBC_CMD(msg=False))
# Print the status of the solution
print("Status:", LpStatus[problem.status])
# If the problem has an optimal solution
if problem.status == LpStatusOptimal:
    # Print the optimal values of x and y
    print("Optimal x =", value(x))
    print("Optimal y =", value(y))
    # Print the optimal value of the objective function
    print("Optimal Z =", value(problem.objective))
```

OUTPUT:

Status: Optimal

Status: Infeasible

Q.9) Apply each of the following Transformation on the point P[2, -3].

(I)Refection through X-axis.

(II)Scaling in X-co-ordinate by factor 2.

(III) Scaling in Y-co-ordinate by factor 1.5.

(IV) Reflection through the line y = x

Syntax:

```
import numpy as np
# Point P
P = np.array([2, -3])
# Transformation 1: Reflection through X-axis
T1 = np.array([[1, 0], [0, -1]])
P_T1 = np.dot(T1, P)
# Transformation 2: Scaling in X-coordinate by factor 2
T2 = np.array([[2, 0], [0, 1]])
P_T2 = np.dot(T2, P)
# Transformation 3: Scaling in Y-coordinate by factor 1.5
T3 = np.array([[1, 0], [0, 1.5]])
P_T3 = np.dot(T3, P)
# Transformation 4: Reflection through the line y = x
T4 = np.array([[0, 1], [1, 0]])
P_T4 = np.dot(T4, P)
# Displaying the results
print("Original Point P: ", P)
print("Transformation 1: Reflection through X-axis: ", P_T1)
print("Transformation 2: Scaling in X-coordinate by factor 2: ", P_T2)
print("Transformation 3: Scaling in Y-coordinate by factor 1.5: ", P_T3)
```

print("Transformation 4: Reflection through the line y = x: ", P_T4)
OUTPUT:
Original Point P: [2 -3]
Transformation 1: Reflection through X-axis: [2 3]
Transformation 2: Scaling in X-coordinate by factor 2: [4 -3]
Transformation 3: Scaling in Y-coordinate by factor 1.5: [2. -4.5]
Transformation 4: Reflection through the line y = x: [-3 2]

Q.10) Apply each of the following Transformation on the point P[3, -1].
(I) Shearing in Y direction by 2 units.
(II) Scaling in X and Y direction by 1/2 and 3 units respectively.
(III) Shearing in both X and Y direction by -2 and 4 units respectively.
(IV) Rotation about origin by an angle 30 degrees.
Syntax:
import numpy as np
import math
# Point P
P = np.array([3, -1])
# Transformation 1: Shearing in Y direction by 2 units
T1 = np.array([[1, 0], [2, 1]])
P_T1 = np.dot(T1, P)
# Transformation 2: Scaling in X and Y direction by 1/2 and 3 units respectively
T2 = np.array([[1/2, 0], [0, 3]])
P_T2 = np.dot(T2, P)
# Transformation 3: Shearing in both X and Y direction by -2 and 4 units respectively
T3 = np.array([[1, -2], [4, 1]])
P_T3 = np.dot(T3, P)
# Transformation 4: Rotation about origin by an angle 30 degrees
theta = np.deg2rad(30) # Convert angle to radians
T4 = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
P_T4 = np.dot(T4, P)
# Displaying the results
print("Original Point P: ", P)
print("Transformation 1: Shearing in Y direction by 2 units: ", P_T1)
print("Transformation 2: Scaling in X and Y direction by 1/2 and 3 units respectively: ", P_T2)
print("Transformation 3: Shearing in both X and Y direction by -2 and 4 units respectively: ", P_T3)
print("Transformation 4: Rotation about origin by an angle 30 degrees: ", P_T4)

OUTPUT:
Original Point P: [3 -1]
Transformation 1: Shearing in Y direction by 2 units: [3 5]
Transformation 2: Scaling in X and Y direction by 1/2 and 3 units respectively: [1.5 -3.]
Transformation 3: Shearing in both X and Y direction by -2 and 4 units respectively: [5 11]
Transformation 4: Rotation about origin by an angle 30 degrees: [3.09807621 0.6339746]