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 11 **Expt. No .** 11 **Class :-** S.Y.BCS

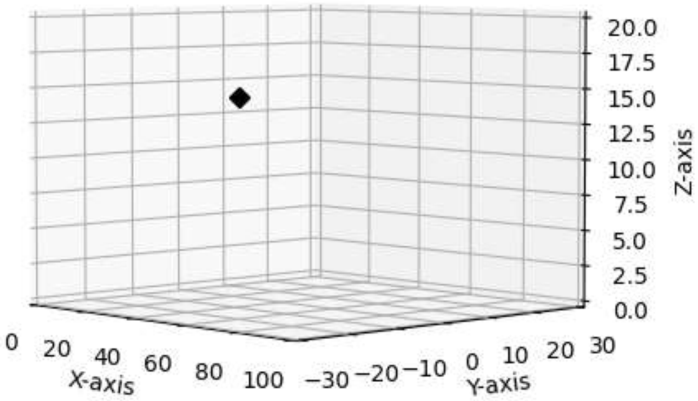
Q.1) Write a python program to plot 3D axes with labels as X – axis and Y – axis And z axis and also plot following point. With given coordinate in the same graph (70,-25,15) as a diamond in black color

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Create a 3D plot figure
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# Define the point coordinates
x = 70
y = -25
z = 15
# Plot the point as a diamond shape in black color
ax.plot([x], [y], [z], marker='D', color='black')
# Set labels for the axes
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
# Set limits for the axes
ax.set_xlim([0, 100])
ax.set_ylim([-30, 30])
ax.set_zlim([0, 20])
# Display the plot
plt.show()
```

plt.show()

OUTPUT:



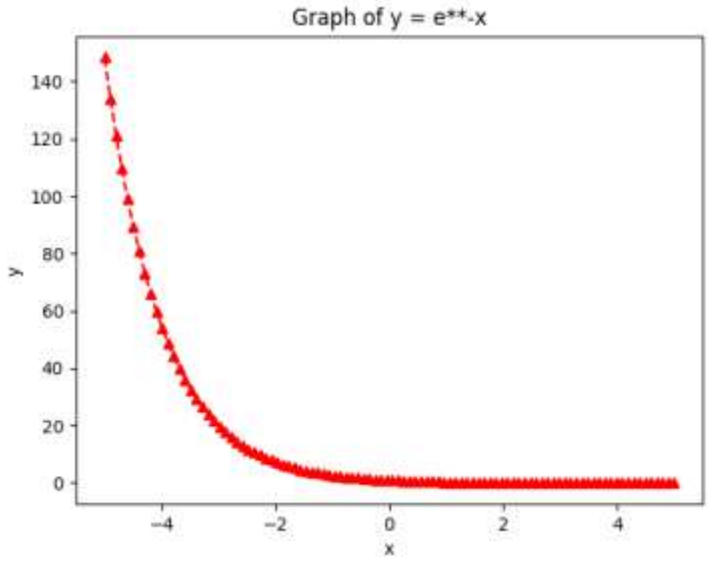
Q.2) Plot the graph of y = e**-x in [-5,5] with red dashed line with Upward pointing Triangle

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Generate x values in the range [-5,5]
x = np.linspace(-5, 5, 100)
# Compute y values using y = e**-x
y = np.exp(-x)
# Create a figure and axis
fig, ax = plt.subplots()
# Plot the graph with red dashed line and upward pointing triangles as markers
ax.plot(x, y, 'r--', marker='^')
# Set labels for the x-axis and y-axis
ax.set_xlabel('x')
ax.set_ylabel('y')
# Set title for the plot
ax.set_title('Graph of y = e**-x')
# Display the plot
```

plt.show() plt.show()

OUTPUT:



Q.3) Using python, represent the following information using a bar graph (in green color)

Subject	Maths	Science	English	Marathi	Hindi
Percentage of passing	68	90	70	85	91

Syntax:

import matplotlib.pyplot as plt

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

height = [68,90,70,85,91]

tick_label=['Maths','Science','English','Marathi',’Hindi']

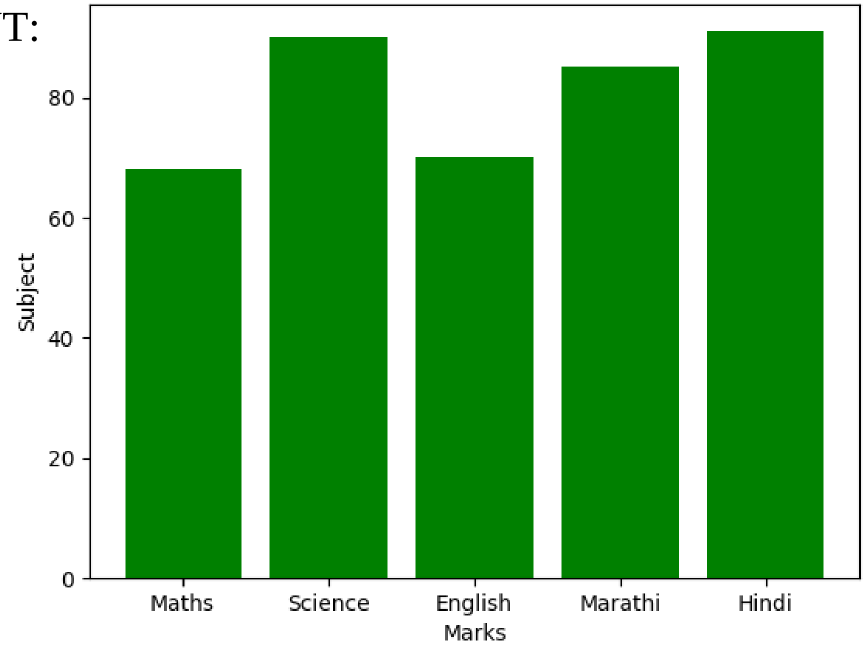
plt.bar (left,height,tick_label = tick_label,width = 0.8 ,color = ['green','green'])

plt.xlabel('Item')

plt.ylabel('Expenditure')

plt. show()

OUTPUT:



Q.4) Write a python program to rotate the ABC by 90° where A(1, 1), B(2, -2), C(1, 2).

Syntax:

```
import numpy as np
# Define the original points
A = np.array([1, 1])
B = np.array([2, -2])
C = np.array([1, 2])

# Define the rotation matrix for rotation by 90 degrees counterclockwise
angle = np.radians(90)
rotation_matrix = np.array([[np.cos(angle), -np.sin(angle)],
                            [np.sin(angle), np.cos(angle)]])
# Apply the rotation to the points
A_rotated = np.dot(rotation_matrix, A)
B_rotated = np.dot(rotation_matrix, B)
C_rotated = np.dot(rotation_matrix, C)
# Print the rotated points
print("Rotated Point A: ", A_rotated)
print("Rotated Point B: ", B_rotated)
print("Rotated Point C: ", C_rotated)
```

Output:

Rotated Point A: [-1. 1.]
Rotated Point B: [2. 2.]
Rotated Point C: [-2. 1.]

Q.5) Write a python program to reflect the ABC through the line y = 3 where A(1, 0), B(2, -2), C(-1, 2).

Syntax:

```
import numpy as np
# Define the reflection line y = 3
reflection_line = 3
# Define the points A, B, and C
A = np.array([1, 0])
B = np.array([2, -2])
C = np.array([-1, 2])
# Compute the reflected points A', B', and C'
Ap = np.array([A[0], 2 * reflection_line - A[1]])
Bp = np.array([B[0], 2 * reflection_line - B[1]])
Cp = np.array([C[0], 2 * reflection_line - C[1]])
# Print the original points and reflected points
print("Original Points:")
print("A: ", A)
print("B: ", B)
print("C: ", C)
print("Reflected Points:")
print("A':", Ap)
print("B':", Bp)
print("C':", Cp)
```

Output:

Original Points:

A: [1 0]

B: [2 -2]

C: [-1 2]

Reflected Points:

A': [1 6]

B': [2 8]

C': [-1 4]

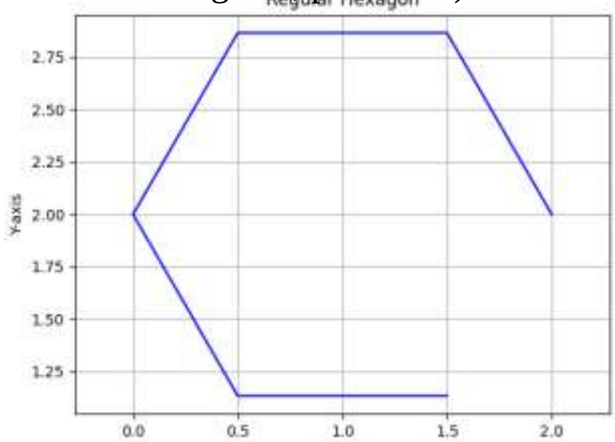
Q.6) Write a python program to draw a polygon with 6 sides and radius 1 centered at (1,2) and find its area and perimeter

Synatx:

```
import math

import matplotlib.pyplot as plt

import numpy as np

# Define the center of the hexagon

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

# Define the radius of the hexagon

radius = 1

# Calculate the coordinates of the vertices of the hexagon

angle_deg = np.linspace(0, 360, 7)[:-1]

angle_rad = np.deg2rad(angle_deg)

x_coords = center[0] + radius * np.cos(angle_rad)

y_coords = center[1] + radius * np.sin(angle_rad)

# Plot the hexagon

plt.plot(x_coords, y_coords, 'b-')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')
```

```
plt.title('Regular Hexagon')
plt.axis('equal')
plt.grid(True)
plt.show()
# Calculate the area of the hexagon
side_length = 2 * radius * np.sin(np.pi / 3)
area = (3 * np.sqrt(3) * side_length ** 2) / 2
# Calculate the perimeter of the hexagon
perimeter = 6 * side_length
# Print the area and perimeter
print("Area of the hexagon:", area)
print("Perimeter of the hexagon:", perimeter)
```

OUTPUT:



Area of the hexagon: 7.794228634059947

Perimeter of the hexagon: 10.392304845413264

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

Max Z = x + y
Subjected to
x >= 6
y >= 6
x+y >= 11
x > 0 , y > 0

Syntax:

```
from pulp import *

# Create a maximization problem

prob = LpProblem("Maximization Problem", LpMaximize)

# Define decision variables

x = LpVariable("x", lowBound=0, cat='Continuous')

y = LpVariable("y", lowBound=0, cat='Continuous')

# Define the objective function

prob += x + y, "Z"

# Define the constraints

prob += x >= 6, "Constraint 1"

prob += y >= 6, "Constraint 2"

prob += x + y >= 11, "Constraint 3"

# Solve the problem

prob.solve()

# Print the status of the problem

print("Status:", LpStatus[prob.status])

# Print the optimal solution

print("Optimal Solution:")

print("x =", value(x))

print("y =", value(y))

# Print the optimal objective value

print("Z =", value(prob.objective))
```

OUTPUT:

```
Status: Unbounded
Optimal Solution:
x = 0.0
y = 0.0
Z = 0.0
```

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 = 3x+5y + 4z
subject to
2x+ 3y <= 8
2y + 5z <= 10
3x + 2y + 4z <= 15
x>=0,y>=0,z>=0

Syntax:

```
from pulp import *
# Create a minimization problem
prob = LpProblem("Minimization Problem", LpMinimize)
# Define decision variables
x = LpVariable("x", lowBound=0, cat='Continuous')
y = LpVariable("y", lowBound=0, cat='Continuous')
z = LpVariable("z", lowBound=0, cat='Continuous')
# Define the objective function
prob += 3*x + 5*y + 4*z, "Z"
# Define the constraints
prob += 2*x + 3*y <= 8, "Constraint 1"
prob += 2*y + 5*z <= 10, "Constraint 2"
prob += 3*x + 2*y + 4*z <= 15, "Constraint 3"
# Solve the problem
prob.solve()
# Print the status of the problem
print("Status:", LpStatus[prob.status])
# Print the optimal solution
print("Optimal Solution:")
print("x =", value(x))
print("y =", value(y))
print("z =", value(z))
# Print the optimal objective value
print("Z =", value(prob.objective))
```

```
Status:  Optimal
Optimal Solution:
x =  0.0
y =  0.0
z =  0.0
Z =  0.0
```

Q.9) Write a python program lo apply the following transformation on the point (-2, 4)

(I) Reflection through x – axis

(II) Scaling in X – coordinate by 6 factor

(III) Shearing in x direction by 4 unit

(IV) Rotation About origin through an angle 30

Syntax:

```
import math
# Initial point
point = (-2, 4)
x, y = point
# Transformation 1: Reflection through x-axis
point_reflection_x_axis = (x, -y)
# Transformation 2: Scaling in X-coordinate by 6 factor
scale_factor = 6
point_scaling_x = (x * scale_factor, y)
# Transformation 3: Shearing in x-direction by 4 units
shear_factor = 4
point_shearing_x = (x + shear_factor * y, y)
# Transformation 4: Rotation about origin through an angle of 30 degrees
angle = 30
angle_rad = math.radians(angle)
point_rotation = (x * math.cos(angle_rad) - y * math.sin(angle_rad), x * math.sin(angle_rad) + y * math.cos(angle_rad))
# Print the transformed points
print("Transformation 1: Reflection through x-axis")
print("x =", point_reflection_x_axis[0])
print("y =", point_reflection_x_axis[1])
print("\nTransformation 2: Scaling in X-coordinate by 6 factor")
print("x =", point_scaling_x[0])
print("y =", point_scaling_x[1])
print("\nTransformation 3: Shearing in x-direction by 4 units")
print("x =", point_shearing_x[0])
print("y =", point_shearing_x[1])
print("\nTransformation 4: Rotation about origin through an angle of 30 degrees")
print("x =", point_rotation[0])
print("y =", point_rotation[1])
```

OUTPUT:

```
Transformation 1: Reflection through x-axis
x = -2
```

y = -4
Transformation 2: Scaling in X-coordinate by 6 factor
x = -12
y = 4
Transformation 3: Shearing in x-direction by 4 units
x = 14
y = 4
Transformation 4: Rotation about origin through an angle of 30 degrees
x = -3.732050807568877
y = 2.464101615137755

Q.10) Find the combined transformation between the point by using Python program for the following sequence of transformation:-

(I) Rotation about origin through an angle pi/2.
(II) Uniform scaling by -6.4units
(III) Scaling in x & y-Coordinate by 3 &5 units respectively.
(IV) Shearing in X – Direction by 6 unit.

Syntax:

```
import math
# Initial point
point = (3, 5)
x, y = point
# Transformation 1: Rotation about origin through an angle of pi/2
angle_rad = math.pi/2
point_rotation = (x * math.cos(angle_rad) - y * math.sin(angle_rad), x * math.sin(angle_rad) + y * math.cos(angle_rad))
# Transformation 2: Uniform scaling by -6.4 units
scale_factor_uniform = -6.4
point_uniform_scaling = (x * scale_factor_uniform, y * scale_factor_uniform)
# Transformation 3: Scaling in x & y-coordinate by 3 & 5 units respectively
scale_factor_x = 3
scale_factor_y = 5
point_scaling = (x * scale_factor_x, y * scale_factor_y)
# Transformation 4: Shearing in X-Direction by 6 units
shear_factor_x = 6
point_shearing_x = (x + shear_factor_x * y, y)
# Combined Transformation
point_combined_transformation = point_rotation
point_combined_transformation = (point_combined_transformation[0] * scale_factor_uniform, point_combined_transformation[1] * scale_factor_uniform)
```

```
point_combined_transformation = (point_combined_transformation[0] * scale_factor_x, point_combined_transformation[1] * scale_factor_y)
point_combined_transformation = (point_combined_transformation[0] + shear_factor_x * point_combined_transformation[1], point_combined_transformation[1])
# Print the transformed points
print("Transformation 1: Rotation about origin through an angle of pi/2")
print("x =", point_rotation[0])
print("y =", point_rotation[1])
print("\nTransformation 2: Uniform scaling by -6.4 units")
print("x =", point_uniform_scaling[0])
print("y =", point_uniform_scaling[1])
print("\nTransformation 3: Scaling in x & y-coordinate by 3 & 5 units respectively")
print("x =", point_scaling[0])
print("y =", point_scaling[1])
print("\nTransformation 4: Shearing in X-Direction by 6 units")
print("x =", point_shearing_x[0])
print("y =", point_shearing_x[1])
print("\nCombined Transformation:")
print("x =", point_combined_transformation[0])
print("y =", point_combined_transformation[1])
```

OUTPUT:

```
Transformation 1: Rotation about origin through an angle of pi/2
x = -5.0
y = 3.0000000000000004
Transformation 2: Uniform scaling by -6.4 units
x = -19.200000000000003
y = -32.0
Transformation 3: Scaling in x & y-coordinate by 3 & 5 units respectively
x = 9
y = 25
Transformation 4: Shearing in X-Direction by 6 units
x = 33
y = 5
Combined Transformation:
x = -480.0000000000001
y = -96.00000000000001
```