Sahakar Maharshi Bhausaheb Santuji Thorat
College Sangamner

DEPARTMENT OF COMPUTER SCIENCE

Sub : Mathematics

Remark
Demonstrator's

Signature

Date:- / /20

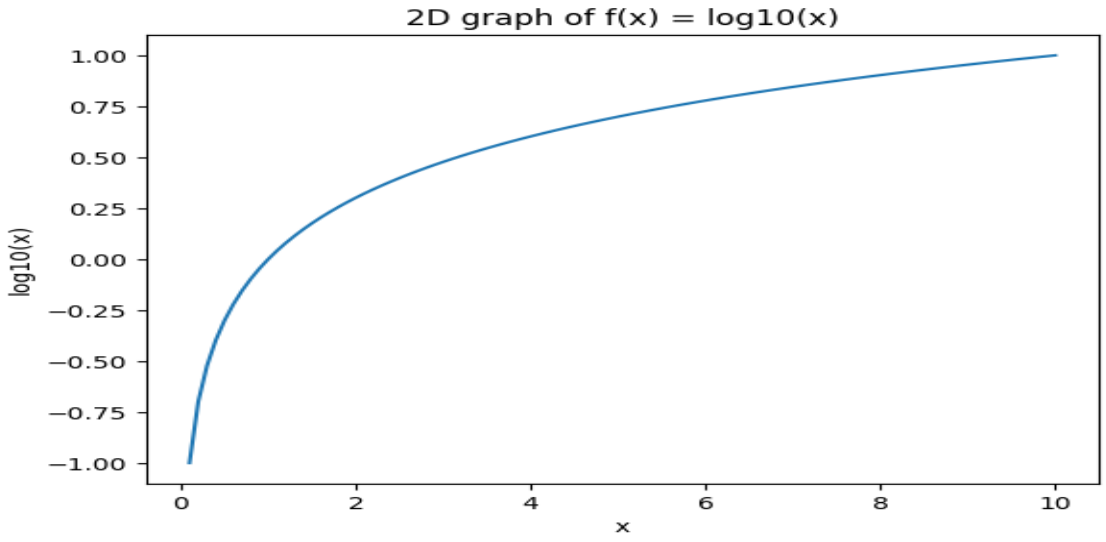
Name:- Gorde Yash Somnath **Roll.No:-21 Date:-**
Title of the expt:- **Page.no:-** **Class:-BCS**

Q1) Attempt any Two Of the following

a) Write a python program to plot 2D graph of the function $f(x)=\log_{10}(x)$ in the interval [0,10]

->

```
import matplotlib.pyplot as plt
import numpy as np
def f(x):
return np.log10(x)
x = np.linspace(0.1, 10, 100)
y = f(x)
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('log10(x)')
plt.title('2D graph of f(x) = log10(x)')
plt.show()
```

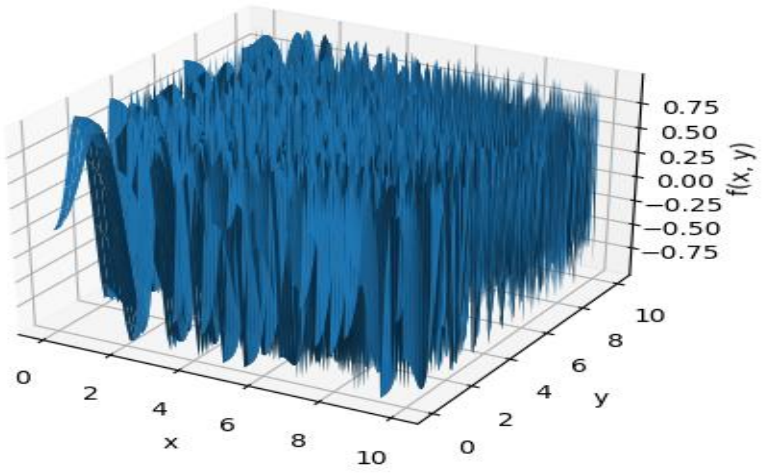


B) Using python ,generate 3D surface plot for the function $f(x)=\sin(x^2+y^2)$ in the interval [0,10]

->

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
def f(x, y):
    return np.sin(x**2 + y**2)
x = np.linspace(0, 10, 100)
y = np.linspace(0, 10, 100)
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_xlabel('x')
        ax.set_ylabel('y')
        ax.set_zlabel('f(x, y)')
        plt.show()
```

output :



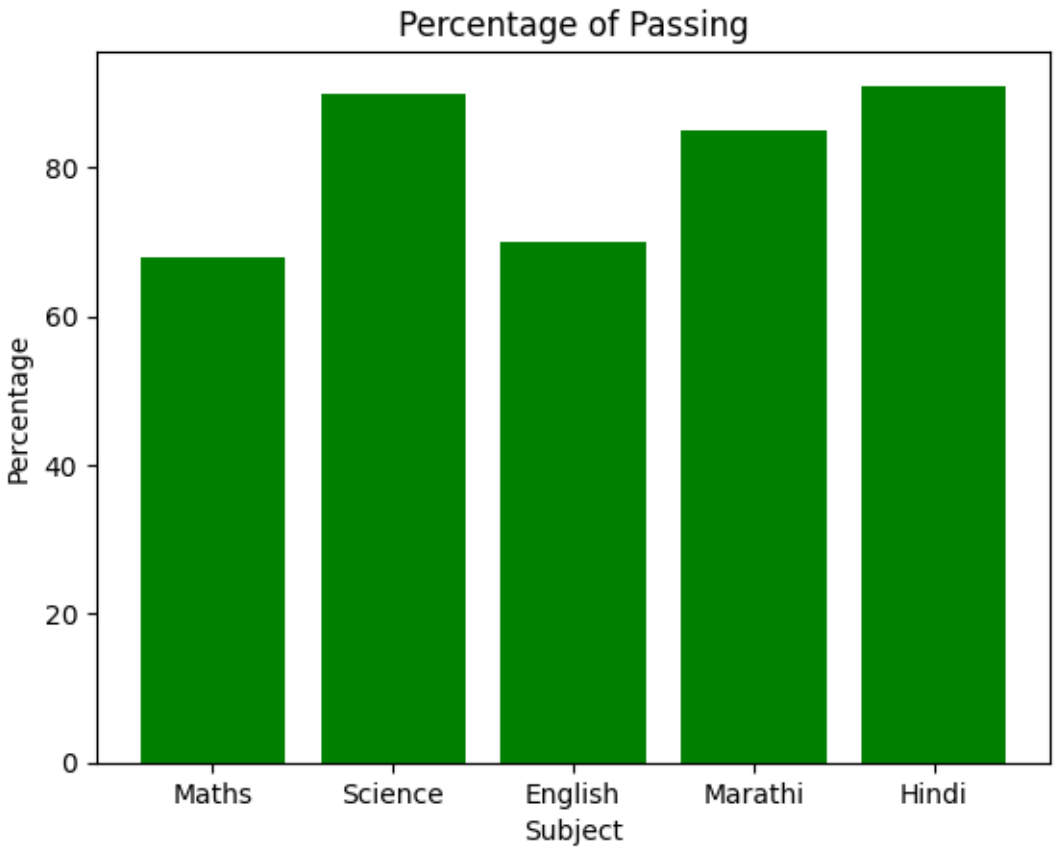
C **) Using Python draw a bar graph in Green color to represent the data below**

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

->

```
import matplotlib.pyplot as plt
subjects = ['Maths', 'Science', 'English', 'Marathi', 'Hindi']
percentage = [68, 90, 70, 85, 91]
plt.bar(subjects, percentage, color='green')
plt.title('Percentage of Passing')
plt.xlabel('Subject')
plt.ylabel('Percentage')
plt.show()
```

output :



Q2) Attempt any Two of the following

a) Using sympy declare the points A(0,2) ,B(5,2),C(3,0) check whether these points are colliner .Declare the line passing through the point A and B , find the distance of this from point C.

->

```
from sympy import Point, Line
A = Point(0, 2)
B = Point(5, 2)
C = Point(3, 0)
if A.is_collinear(B, C):
   print("The points A, B, and C are collinear")
else:
   print("The points A, B, and C are not collinear")

AB = Line(A, B)
distance = AB.distance(C)
print("The distance of the line passing through A and B from C is:", distance)
```

output-:

```
The points A, B, and C are not collinear
The distance of the line passing through A and B from C is: 2
```

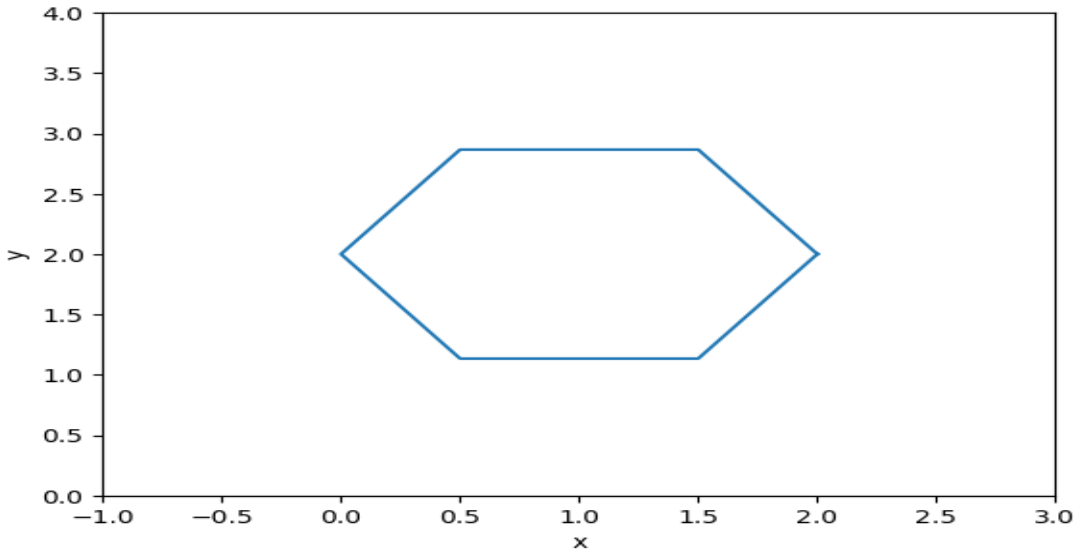
b) Using python drawn a regular polygon with 6 sides and radius 1 centerd at (1,2) and find its area

->

```
import matplotlib.pyplot as plt
import math
center = (1, 2)

num_sides = 6
radius = 1

vertices = []
for i in range(num_sides):
   x = center[0] + radius * math.cos(2*math.pi*i/num_sides)
   y = center[1] + radius * math.sin(2*math.pi*i/num_sides)
   vertices.append((x, y))

vertices.append(vertices[0])

x_coords, y_coords = zip(*vertices)
plt.plot(x_coords, y_coords)

plt.xlim(center[0] - radius - 1, center[0] + radius + 1)
plt.ylim(center[1] - radius - 1, center[1] + radius + 1)
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.show()
```

output :



c) Write a Python program to find the area and perimeter of the Δ ABC , where A[0,0],B[6,0],C[4,4]

->

```
import math
A = [0, 0]
B = [6, 0]
C = [4, 4]

AB = math.sqrt((B[0]-A[0])**2 + (B[1]-A[1])**2)
BC = math.sqrt((C[0]-B[0])**2 + (C[1]-B[1])**2)
CA = math.sqrt((A[0]-C[0])**2 + (A[1]-C[1])**2)
perimeter = AB + BC + CA
print("The perimeter of the triangle is:", perimeter)

s = perimeter / 2

area = math.sqrt(s*(s-AB)*(s-BC)*(s-CA))
print("The area of the triangle is:", area)
```

output :

```
The perimeter of the triangle is: 16.12899020449196
The area of the triangle is: 11.999999999999998
```

Q3) Attempt any ONE of the following

A)Write a python program to solve the following LPP:

Max Z=5x+3y

Subject to x+y≤7

2x+5y≤1

x≥0,y≥0

->

```
from scipy.optimize import linprog
c = [-5, -3]
A = [[1, 1], [2, 5]]
b = [7, 1]

x_bounds = (0, None)
y_bounds = (0, None)

res = linprog(c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds], method='simplex')
print("Optimal solution:")
print("x =", res.x[0])
print("y =", res.x[1])
print("Optimal value of Z:")
print("Z =", -res.fun)
```

output:

```
Optimal solution:
x = 0.5
y = 0.0
Optimal value of Z:
Z = 2.5
```

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

max Z=3x+2y+5z

subject to x+2y+z≤430

3x+2z≤460

x+4y≤120

x,y,z≥0

->

```
from pulp import *
prob = LpProblem("LPP", LpMaximize)
x = LpVariable("x", lowBound=0, cat='Continuous')
y = LpVariable("y", lowBound=0, cat='Continuous')
z = LpVariable("z", lowBound=0, cat='Continuous')
prob += 3*x + 2*y + 5*z
prob += x + 2*y + z <= 430
```

```
prob += 3*x + 2*z <= 460
prob += x + 4*y <= 120

status = prob.solve()

if status == 1:
    print("Optimal solution:")
    print("x =", value(x))
    print("y =", value(y))
    print("z =", value(z))
    print("Optimal value of Z:")
    print("Z =", value(prob.objective))
else:
    print("The problem is infeasible.")
```

B) Attempt any one of the following

1) Apply python program in each of the following transformation of the point P[4,-2]

i) Reflection through Y-axis

ii) Scaling in X-cordinate by factor 3

iii) Scaling in Y-cordinate by factor 2.5

iv) Reflection through the line y=-x

->

```
    # Define the point P
P = (4, -2)
# Reflect P through Y-axis
P_reflected = (-P[0], P[1])
# Print the result
print("Reflection through Y-axis:")
print("P' =", P_reflected)

# Define the point P
P = (4, -2)
# Scale P in X-coordinate by factor 3
P_scaled = (3*P[0], P[1])
# Print the result
print("Scaling in X-coordinate by factor 3:")
print("P' =", P_scaled)

# Define the point P
```

```
P = (4, -2)
# Scale P in Y-coordinate by factor 2.5
P_scaled = (P[0], 2.5*P[1])
# Print the result
print("Scaling in Y-coordinate by factor 2.5:")
print("P' =", P_scaled)


# Define the point P
P = (4, -2)
# Reflect P through the line y=-x
P_reflected = (-P[1], -P[0])
print("Reflection through the line y=-x:")
print("P' =", P_reflected)
```

output :

```
    Reflection through Y-axis:
P' = (-4, -2)
Scaling in X-coordinate by factor 3:
P' = (12, -2)
Scaling in Y-coordinate by factor 2.5:
P' = (4, -5.0)
Reflection through the line y=-x:
P' = (2, -4)
```

2) Find the combined transformation of the line segment between the points A[4,-1] & B[3,0] by using the python
program for the following sequence of transformation
Rotation about origin through an angle π
ii) Shering in Y direction by 4.5 units
iii) Scaling in X-cordinate by 3 units
iv) Reflection through the line y=x

->

```
 import math
# Define the points A and B
A = (4, -1)
B = (3, 0)

# i) Rotate about origin through an angle pi
def rotate(point, angle):
    x = point[0]*math.cos(angle) - point[1]*math.sin(angle)
    y = point[0]*math.sin(angle) + point[1]*math.cos(angle)
    return (x, y)
A = rotate(A, math.pi)
B = rotate(B, math.pi)

# ii) Shear in Y direction by 4.5 units
```

```
def shear(point, factor):
    x = point[0] + factor*point[1]
    y = point[1]
    return (x, y)
A = shear(A, 4.5)
B = shear(B, 4.5)

# iii) Scale in X-coordinate by factor 3
def scale(point, factor):
    x = factor*point[0]
    y = point[1]
    return (x, y)
A = scale(A, 3)
B = scale(B, 3)

# iv) Reflect through the line y=x
def reflect(point):
    x = point[1]
    y = point[0]
    return (x, y)
A = reflect(A)
B = reflect(B)
# Print the final transformed line segment
print("The final transformed line segment:")
print("A' =", A)
print("B' =", B)
```

output :

```
The final transformed line segment:
A' = (1.0000000000000004, 1.5000000000000053)
B' = (3.6739403974420594e-16, -8.999999999999995)
```