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:-__Slip no 15_____Page.no:-_____Class:-_____BCS_____

Q1) Attempt any ONE of the following

A) Write a python program to find the area of tringle ABC ,where A[0,0],B[5,0],C[3,3]
-→

```
import numpy as np
A = np.array([0, 0])
B = np.array([5, 0])
C = np.array([3, 3])

area = 0.5 * np.abs(np.cross(B - A, C - A))
print("The area of triangle ABC is:", area)
```

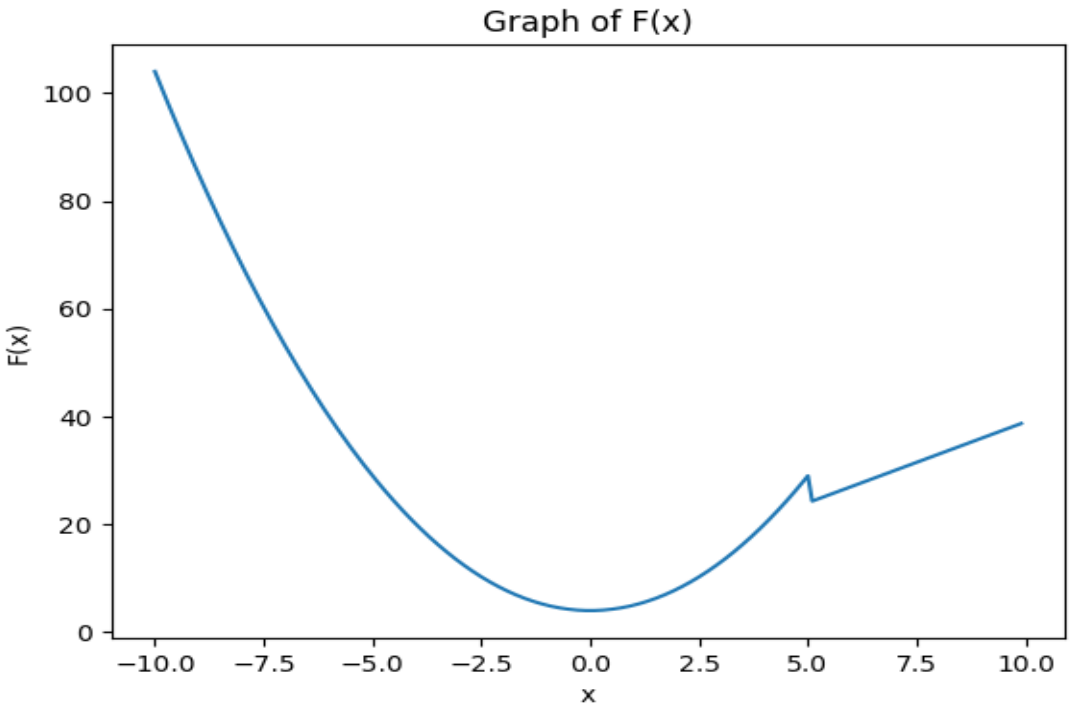
output :

The area of triangle ABC is: 7.5

B) Write the python program to plot the graph of the function using def ()
F(x)={ x^2 +4 if -10≤x<5 ,3x+9 if 5≤x<10}
-→

```
import matplotlib.pyplot as plt
import numpy as np
def F(x):
    if -10 <= x < 5:
        return x**2 + 4
    elif 5 <= x < 10:
        return 3*x + 9

x = np.arange(-10, 10, 0.1)
y = [F(i) for i in x]

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('F(x)')
plt.title('Graph of F(x)')
```

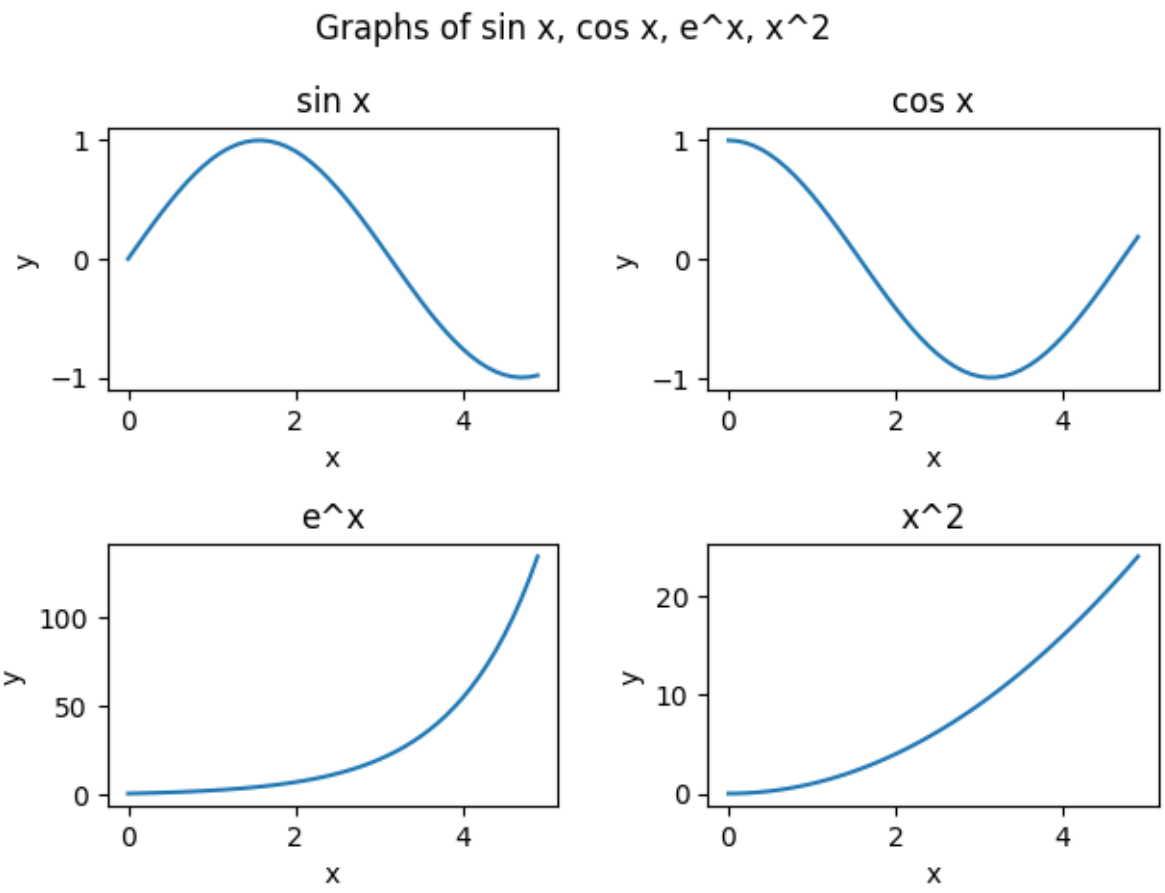
plt.show()



C) Write the python program to plot the graphs of sin x,cos x,e^x ,x^2 in[0,5] in one figure with 2x2 subplots

-→

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0, 5, 0.1)
fig, axs = plt.subplots(2, 2)

axs[0, 0].plot(x, np.sin(x))
axs[0, 0].set_title('sin x')
axs[0, 1].plot(x, np.cos(x))
axs[0, 1].set_title('cos x')
axs[1, 0].plot(x, np.exp(x))
axs[1, 0].set_title('e^x')
axs[1, 1].plot(x, x**2)
axs[1, 1].set_title('x^2')

for ax in axs.flat:
    ax.set(xlabel='x', ylabel='y')

fig.suptitle('Graphs of sin x, cos x, e^x, x^2')
fig.tight_layout()
plt.show()
```



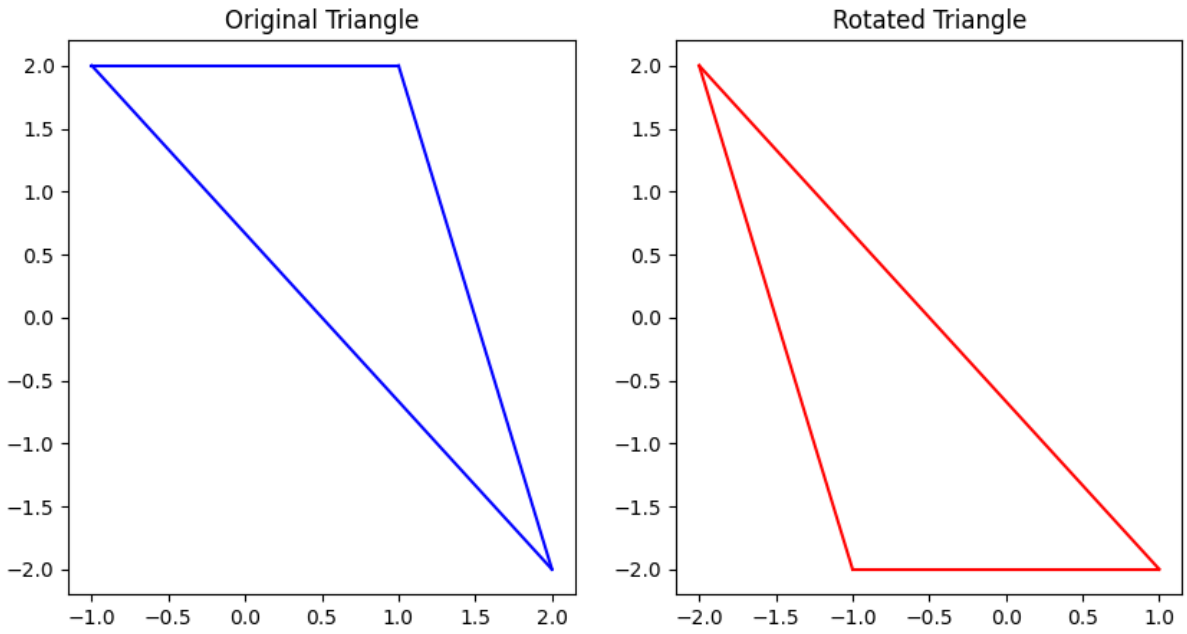
Q 2) Attempt any TWO of the following

A) Write the python program to rotate the triangle ABC by 180 degree, where a A[1,2],B[2,-2] & C[-1,2]

-→

```
import matplotlib.pyplot as plt
import numpy as np
A = np.array([1, 2])
B = np.array([2, -2])
C = np.array([-1, 2])
theta = np.pi
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])

A_rotated = R.dot(A)
B_rotated = R.dot(B)
C_rotated = R.dot(C)
```

```
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

ax[0].set_title("Original Triangle")
ax[0].plot([A[0], B[0]], [A[1], B[1]], 'b')
ax[0].plot([B[0], C[0]], [B[1], C[1]], 'b')
ax[0].plot([C[0], A[0]], [C[1], A[1]], 'b')

ax[1].set_title("Rotated Triangle")
ax[1].plot([A_rotated[0], B_rotated[0]], [A_rotated[1], B_rotated[1]], 'r')
ax[1].plot([B_rotated[0], C_rotated[0]], [B_rotated[1], C_rotated[1]], 'r')
ax[1].plot([C_rotated[0], A_rotated[0]], [C_rotated[1], A_rotated[1]], 'r')

plt.show()
```

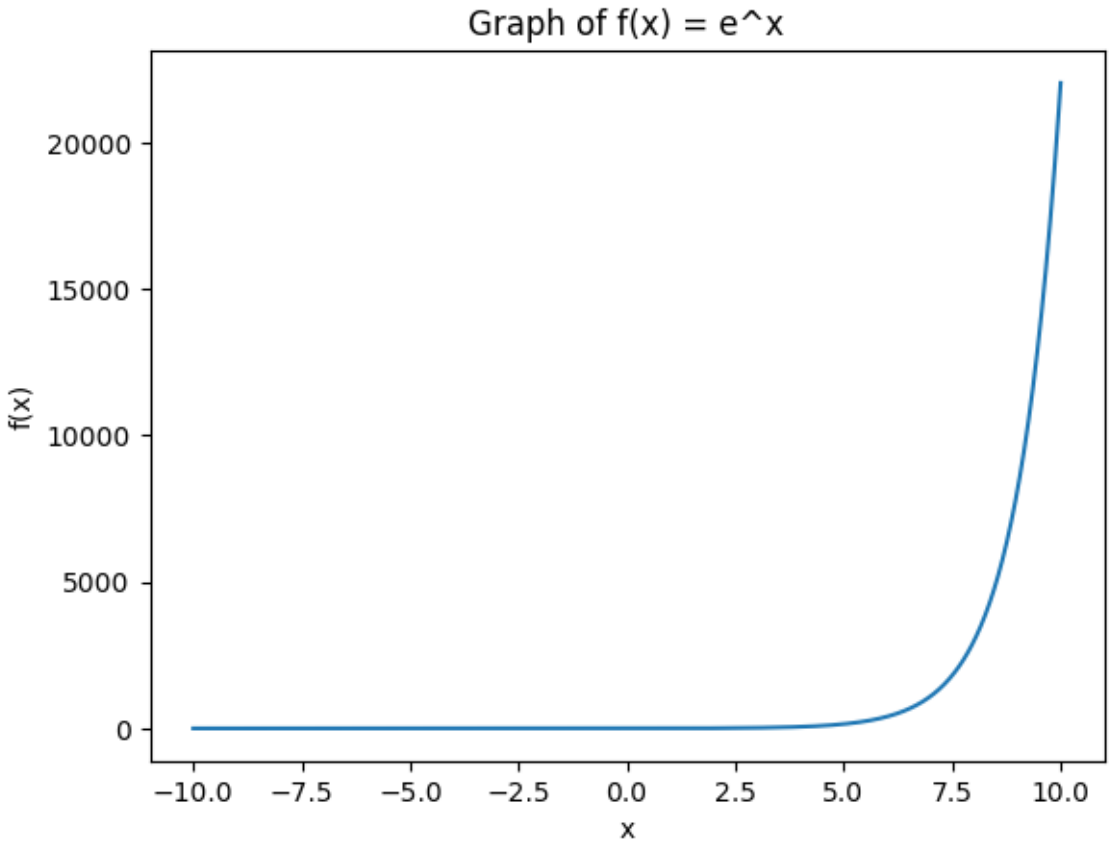


B) Write the python program to plot the graph of function f(x)=e^x in the interval [-10,10]

-→

```
  import numpy as np
import matplotlib.pyplot as plt
def f(x):
   return np.exp(x)

x = np.linspace(-10, 10, 1000)
y = f(x)

plt.plot(x, y)
plt.xlabel('x')
```

```
plt.ylabel('f(x)')
plt.title('Graph of f(x) = e^x')
plt.show()
```

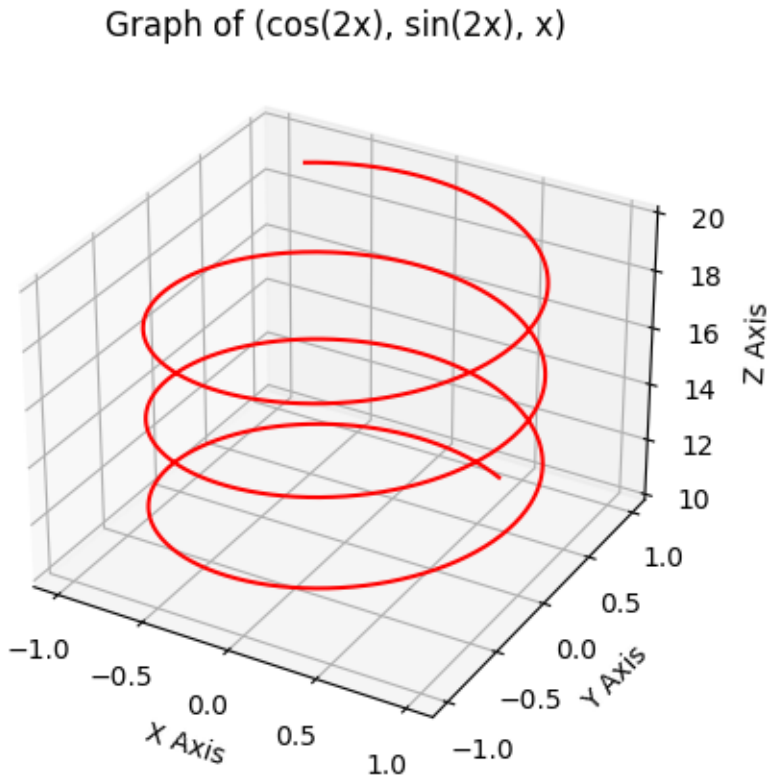


C) Write a python program to plot the 3D graph whose parametric equation in (cos(2x),sin(2x),x) for 10≤x≤20 (in red color), With title to the graph

-→

```
 import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
def f(x):
    return np.cos(2*x), np.sin(2*x), x

x = np.linspace(10, 20, 1000)
y1, y2, y3 = f(x)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(y1, y2, y3, color='red')
```

```
ax.set_xlabel('X Axis')
ax.set_ylabel('Y Axis')
ax.set_zlabel('Z Axis')
plt.title('Graph of (cos(2x), sin(2x), x)')
   plt.show()
```



Q3) Attempt the following

A) Attempt any ONE of the following

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

Min Z=3.5x+2y
Subject to x+y≥5
x≥4
y≤2
x,y≥0

-→

```
from scipy.optimize import linprog
c = [3.5, 2]
A = [[1, 1],
    [-1, 0],
```

```
    [0, 1]]
b = [5, -4, 2]

x_bounds = (0, None)
y_bounds = (0, None)
result = linprog(c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds])

print("Optimal value:", round(result.fun, 2))
print("Optimal point:", tuple(round(x, 2) for x in result.x))
```

output :

```
Optimal value: 14.0
Optimal point: (4.0, 0.0)
```

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

Min Z=x+y

Subject to x≥6

y≥6

x+y≤11

x,y≥0

-→

```
from scipy.optimize import linprog
obj = [1, 1]
lhs = [[-1, 0], [0, -1], [1, 1]]
rhs = [-6, -6, 11]
bounds = [(0, None), (0, None)]
result = linprog(c=obj, A_ub=lhs, b_ub=rhs, bounds=bounds, method='simplex')

print("Status:", result.message)
print("Optimal Solution:", result.fun)
print("Optimal Decision Variables:", result.x)
```

output :

Optimal Solution: 12.0
Optimal Decision Variables: [6. 6.]

B) Attempt any ONE of the following

I) Write a python program to find the combined transformation of the line segment between the points A[5,3] and B[1,4] for the following sequence of transformation

A) First rotation about origin through an angle π/2
B) Followed by scaling in X-coordinate by 5 units
C) Followed by reflection through the line y=x

-→

```
import numpy as np
A = np.array([5, 3])
B = np.array([1, 4])
R = np.array([[0, -1], [1, 0]])  # Rotation matrix
S = np.array([[5, 0], [0, 1]])  # Scaling matrix
F = np.array([[0, 1], [1, 0]])  # Reflection matrix

def apply_transform(matrix, point):
    return np.dot(matrix, point)

AB = B - A  # Vector representing the line segment AB
AB_rot = apply_transform(R, AB)  # Rotate the vector
AB_rot_scale = apply_transform(S, AB_rot)  # Scale the vector
AB_rot_scale_reflect = apply_transform(F, AB_rot_scale)  # Reflect the vector

B_prime = A + AB_rot_scale_reflect

  print(f"The coordinates of the new point B' are: {B_prime}")
```

output :

The coordinates of the new point B' are: [1 -2]

II) write a python program to apply each of the following transformation on the point P[-2,4]

A)Reflection through the line y=x+1
B) Scaling in Y-coordinate by factor 1.5
C) Shering in X direction by 2 units
D) Rotation about origin by an angle 45 degree

-→

```
Import numpy as np
P = np.array([-2, 4])
```

```
F = np.array([[-1/2**0.5, 1/2**0.5], [1/2**0.5, 1/2**0.5]])
P_reflect = np.dot(F, P)
print("Point P after reflection through the line y=x+1:", P_reflect)

S = np.array([[1, 0], [0, 1.5]])
P_scale = np.dot(S, P)
print("Point P after scaling in Y-coordinate by factor 1.5:", P_scale)

SH = np.array([[1, 2], [0, 1]])
P_shear = np.dot(SH, P)
print("Point P after shearing in X direction by 2 units:", P_shear)

theta = np.radians(45)
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
P_rotate = np.dot(R, P)
print("Point P after rotation about origin by 45 degrees:", P_rotate)
```

Output :

```
Point P after reflection through the line y=x+1: [4.24264069 1.41421356]
Point P after scaling in Y-coordinate by factor 1.5: [-2.  6.]
Point P after shearing in X direction by 2 units: [6 4]
Point P after rotation about origin by 45 degrees: [-4.24264069  1.41421356]
```