LAB 3

TASK 1

Write code for line chart to generate 70 numbers with X- axis and Y-axis is mx+c, were m=2 and c=0.3. Draw the four different line charts using matplotlib library. (Take Y1=X, Y2=X**2, Y3=X**3 and

Y4=sqrt(X)).

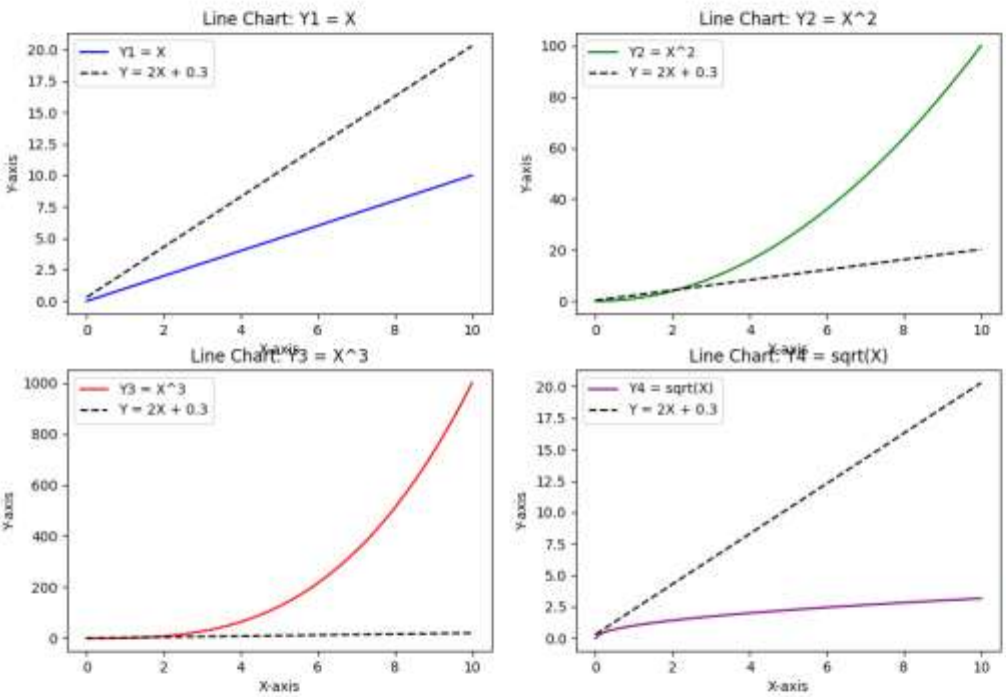
INPUT

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
m = 2
c = 0.3

# Generate 70 numbers for X-axis
X = np.linspace(0, 10, 70)   # 70 evenly spaced numbers between 0 and 10

# Equation Y = mX + c
Y = m * X + c

# Define the four different functions
Y1 = X
Y2 = X**2
Y3 = X**3
Y4 = np.sqrt(X)

# Plotting
plt.figure(figsize=(12, 8))
```

```
# Chart 1: Y vs Y1
plt.subplot(2, 2, 1)
plt.plot(X, Y1, color='blue', label='Y1 = X')
plt.plot(X, Y, '--', color='black', label='Y = 2X + 0.3')
plt.title('Line Chart: Y1 = X')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()

# Chart 2: Y vs Y2
plt.subplot(2, 2, 2)
plt.plot(X, Y2, color='green', label='Y2 = X^2')
plt.plot(X, Y, '--', color='black', label='Y = 2X + 0.3')
plt.title('Line Chart: Y2 = X^2')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()

# Chart 3: Y vs Y3
plt.subplot(2, 2, 3)
plt.plot(X, Y3, color='red', label='Y3 = X^3')
plt.plot(X, Y, '--', color='black', label='Y = 2X + 0.3')
plt.title('Line Chart: Y3 = X^3')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()

# Chart 4: Y vs Y4
```

```
plt.subplot(2, 2, 4)
plt.plot(X, Y4, color='purple', label='Y4 = sqrt(X)')
plt.plot(X, Y, '--', color='black', label='Y = 2X + 0.3')
plt.title('Line Chart: Y4 = sqrt(X)')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.savefig("Task1.png")
plt.tight_layout()
plt.show()
```

OUTPUT

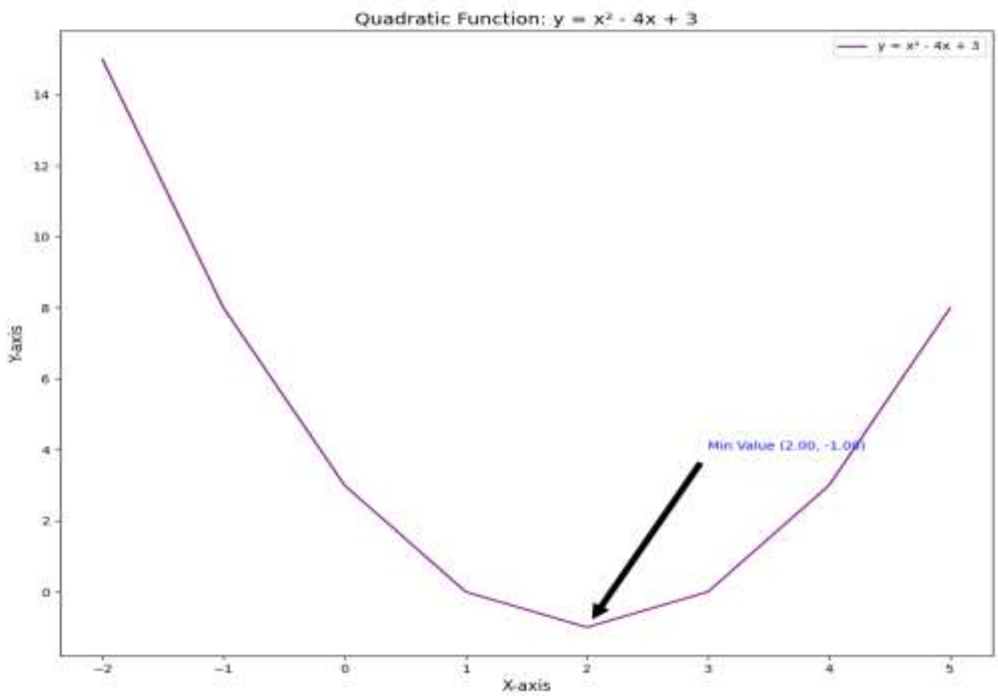


TASK 2

Plot the function y = x^2- 4x + 3 for x ranging from-2 to 6. Identify and annotate the minimum pointof the graph. Add appropriate labels, title, and grid

INPUT

```
x = np.arange(-2, 6)
Y = x**2 - 4*x + 3
a, b, c = 1, -4, 3
x_min_point = -b / (2* a)
y_min_point = x_min_point **2 - 4 * x_min_point + 3
plt.figure(figsize = (12, 10))
plt.plot(x, Y, label= "y = x² - 4x + 3", color = 'purple')
plt.annotate( f"Min Value ({x_min_point:.2f}, {y_min_point:.2f})", xy=(x_min_point, y_min_point), xytext=(x_min_point+1, y_min_point+5), arrowprops=dict(facecolor='black', shrink=0.05),  fontsize=10, color='blue' )
plt.title("Quadratic Function: y = x² - 4x + 3", fontsize=14)
plt.xlabel("X-axis", fontsize=12)
plt.ylabel("Y-axis", fontsize=12)
plt.legend()
plt.savefig("Task2.png")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```



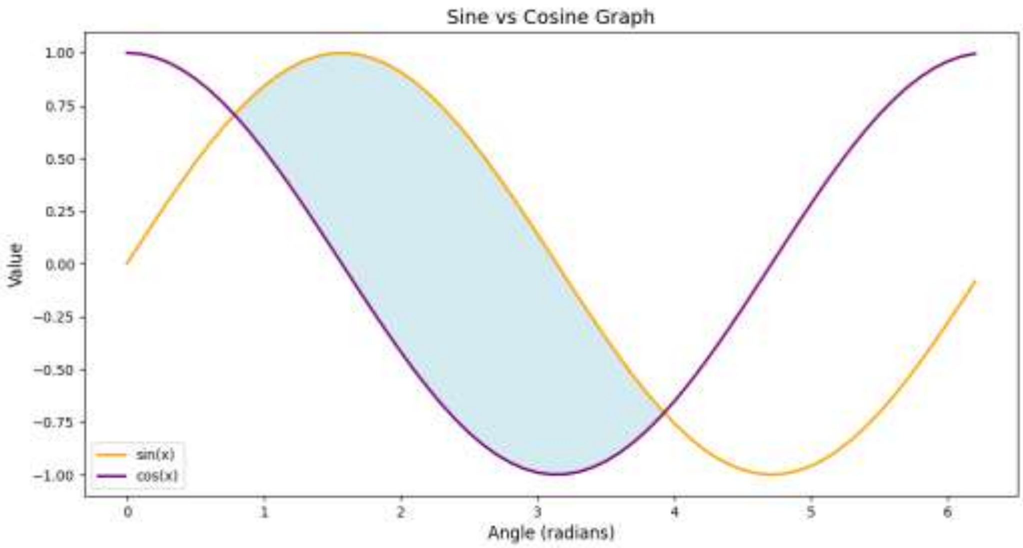
TASK 3

Plot sin(x) and cos(x) from 0 to 2π on the same graph. Use fill_between() to shade the area between the two curves where sin(x) is greater than cos(x). Add labels and a title.

INPUT

```
# Define x values from 0 to 2π
x = np.arange(0, 2 * np.pi, 0.1)


# Compute sine and cosine
Y1 = np.sin(x)
Y2 = np.cos(x)


# Create figure
plt.figure(figsize=(12, 6))


# Plot sine and cosine
plt.plot(x, Y1, label='sin(x)', color='orange', linewidth=2)
plt.plot(x, Y2, label='cos(x)', color='purple', linewidth=2)


# Fill region where sin(x) > cos(x)
plt.fill_between(x, Y1, Y2, where=(Y1 > Y2), color='lightblue', alpha=0.5)
plt.title("Sine vs Cosine Graph", fontsize=14)
plt.xlabel("Angle (radians)", fontsize=12)
plt.ylabel("Value", fontsize=12)
plt.legend()
plt.savefig("Task3.png")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

OUTPUT



TASK 4

Consider monthly temperature data for the year 2023 and plot it using a time-series line graph.

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

temperature = [12, 14, 18, 24, 30, 34, 32, 31, 28, 22, 16, 13]

Label the x-axis as Months

INPUT

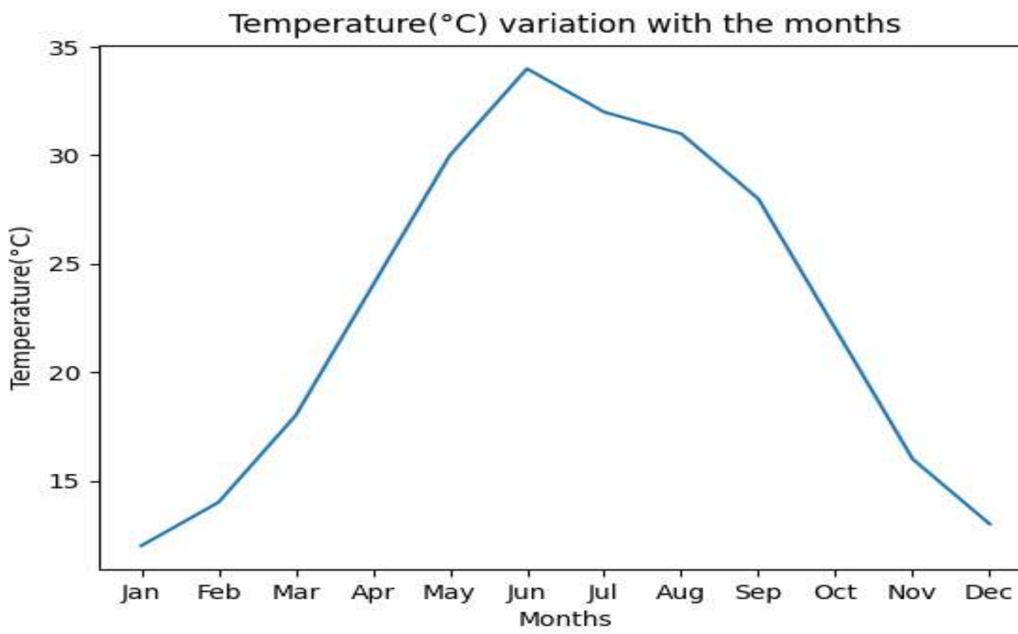
```
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

temperatures = [12, 14, 18, 24, 30, 34, 32, 31, 28, 22, 16, 13]


x= months

y = temperatures


plt.plot(x, y)

plt.title("Temperature(°C) variation with the months")

plt.xlabel("Months")
```

plt.ylabel("Temperature(°C)")

plt.savefig("Task4.png")

plt.show()

OUTPUT



TASK 5

Plot three mathematical functions on the same graph:

- y=x²(Red,DashedLine)
- y=2x+1(Blue,Solid Line)
- y=sqrt(x) (Green, Dotted Line)

Add a legend, labels, and title.

INPUT

Define the value of x with equal spaces

import numpy as np

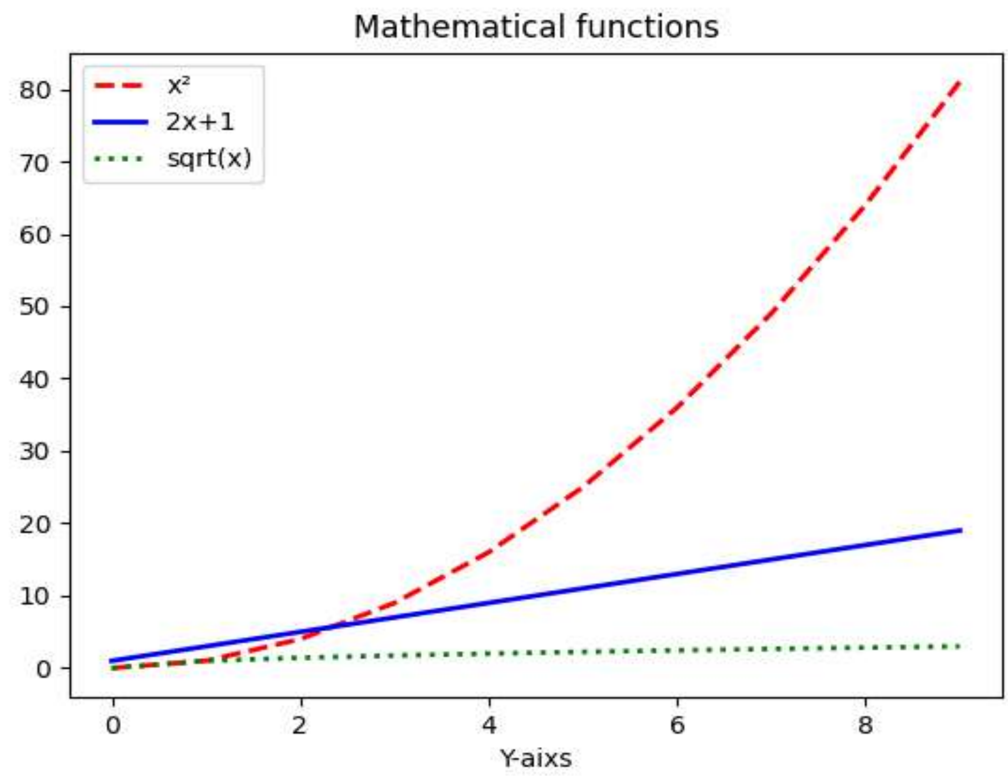
x = np.arange(0, 10, 1)

Compute the functions

Y1 = x**2

```
Y2 = 2*x + 1
Y3 = np.sqrt(x)

plt.plot(x, Y1, label='x²', color='red', linestyle = 'dashed', linewidth=2)
plt.plot(x, Y2, label='2x+1', color='blue', linestyle = 'solid', linewidth=2)
plt.plot(x, Y3, label='sqrt(x)', color='green', linestyle= 'dotted', linewidth=2)
plt.title("Mathematical functions")
plt.legend()
plt.xlabel("X-aixs")
plt.xlabel("Y-aixs")
plt.savefig("Task51.png")
plt.show()
```

OUTPUT



TASK 6

Create a dual-axis plot where:

- Theprimaryy-axis represents the monthly rainfall (in mm) in 2023 as a bar chart.
- Thesecondary y-axis represents the average temperature (in °C) as a line graph on the same plot.

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

rainfall = [78, 62, 55, 41, 32, 140, 300, 280, 150, 90, 85, 80]

temperature = [12, 14, 18, 24, 30, 34, 32, 31, 28, 22, 16, 13]

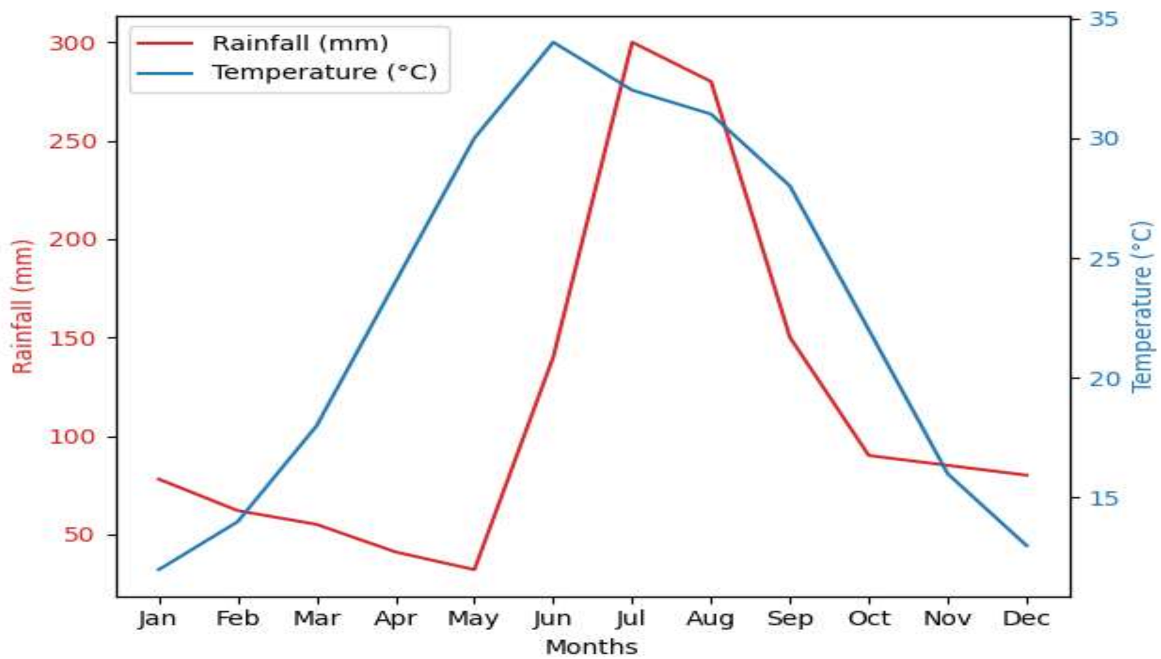
Label both y-axes and use different colors for bars and the line

INPUT

```
import matplotlib.pyplot as plt

import numpy as np


# Define the data

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

rainfall = [78, 62, 55, 41, 32, 140, 300, 280, 150, 90, 85, 80]

temperature = [12, 14, 18, 24, 30, 34, 32, 31, 28, 22, 16, 13]


x = months

Y1 = rainfall

Y2 = temperature


fig, ax1 = plt.subplots()

# Left Y-axis plot

color = 'tab:red'

ax1.set_xlabel('Months')

ax1.set_ylabel('Rainfall (mm)', color=color)

line1, = ax1.plot(x, Y1, color=color, label="Rainfall (mm)")  # unpack with comma

ax1.tick_params(axis='y', labelcolor=color)
```

```
# Right Y-axis plot
ax2 = ax1.twinx()  # Shares the same x-axis
color = 'tab:blue'
ax2.set_ylabel('Temperature (°C)', color=color)
line2, = ax2.plot(x, Y2, color=color, label="Temperature (°C)")  # unpack with comma
ax2.tick_params(axis='y', labelcolor=color)

# Combine legends from both axes
lines = [line1, line2]
labels = [l.get_label() for l in lines]
ax1.legend(lines, labels, loc="upper left")
fig.tight_layout()
plt.savefig("Task6.png")
plt.show()
```

OUTPUT



TAK 7

Draw a house using matplotlib line charts

INPUT

```
# Walls
x_walls = [1, 1, 5, 5, 1]
y_walls = [1, 5, 5, 1, 1]
plt.plot(x_walls, y_walls, color='saddlebrown', label='Walls')

# Roof
x_roof = [1, 3, 5]
y_roof = [5, 7, 5]
plt.plot(x_roof, y_roof, color='saddlebrown', label='Roof')

# Door
x_door = [1.5, 1.5, 2.5, 2.5, 1.5]
y_door = [1, 3, 3, 1, 1]
plt.plot(x_door, y_door, color='blue', label='Door')

# Window1
x_win1 = [4, 4, 4.5, 4.5, 4]
y_win1 = [1.5, 2, 2, 1.5, 1.5]
plt.plot(x_win1, y_win1, color='teal', label='Window 1')

# Window2
x_win2 = [4, 4, 4.5, 4.5, 4]
y_win2 = [3, 3.5, 3.5, 3, 3]
plt.plot(x_win2, y_win2, color='teal', label='Window 2')

plt.title("House Using Line Plots with Multiple Colors")
plt.xlabel("X-axis")
```

```
plt.ylabel("Y-axis")
plt.legend(loc='upper right')
plt.grid(True)
plt.axis('equal')  # Keep aspect ratio square
plt.savefig("Task7.png")
plt.show()
```

OUTPUT