Python Programming

Lab:- 22(Matplotlib Bar plot and Histogram)

Student Id:- AF0417098

Student Name:- Ankush

Matplotlib:- It is a widely-used Python library for creating static, animated, and interactive visualizations in Python. It provides a flexible framework for creating a wide range of plots and charts.

Here are some key points about Matplotlib Bar plot and Histogram:-

Bar Plot

A bar plot (or bar chart) is used to visualize categorical data where each category is represented by a bar, and the length or height of the bar corresponds to the value of the data for that category.

1. **Usage:**
   - Bar plots are best for comparing discrete categories (e.g., sales by product, population by age group).
   - Can be plotted vertically (plt.bar()) or horizontally (plt.barh()).
2. **Key Elements:**
   - Bars: Each bar represents a category.
   - Categories on X-axis: In a vertical bar plot, categories are usually on the X-axis, and the value is on the Y-axis.
   - Bar Width: You can adjust the width of the bars using the width parameter.
   - Colors: You can customize the color of bars for better visualization (color parameter).
   - Edge Color: Adding edges around bars (edgecolor parameter) helps distinguish them.
   - Data Labels: Adding labels to the bars (e.g., counts or values) enhances interpretability.
3. **Enhancements:**

- Use different colors for each bar to represent data more clearly.
- Add data labels on top of bars to show exact values.
- Use gridlines to make the graph easier to read.
- Highlight specific bars (e.g., the highest/lowest values).

Histogram

A histogram is used to represent the distribution of a continuous variable by dividing data into bins and plotting the frequency of data points that fall into each bin.

1. Usage:

   - Histograms are used to show the distribution of data points across a range of values (e.g., age distribution, test scores).
   - Best for continuous data, where each bin represents a range of values.

2. Key Elements:

   - Bins: The range of values grouped together (can be customized using the bins parameter).
   - Frequencies on Y-axis: The height of each bar represents the number of occurrences within that bin.
   - X-axis (Value Range): Represents the range of data values split into bins.

3. Enhancements:

   - Customize bin sizes (bins parameter) to control the granularity of the histogram.
   - Use different colors to enhance the appearance (color parameter).
   - Add data labels to indicate the number of occurrences in each bin.
   - Use gridlines to make the histogram more readable.
   - Adjust the transparency (alpha parameter) if layering multiple histograms.
   - Consider using normalized histograms (density=True) to compare distributions.

Differences:

- Bar Plot:

  - Displays categorical data.
  - Spaces between bars (discrete categories).

  - Bars can represent frequency or other metrics (e.g., sales, count).
- Histogram:
  - Displays continuous data.
  - Bars touch, showing that data is continuous.
  - Bins represent ranges of values and the bar height shows frequency.

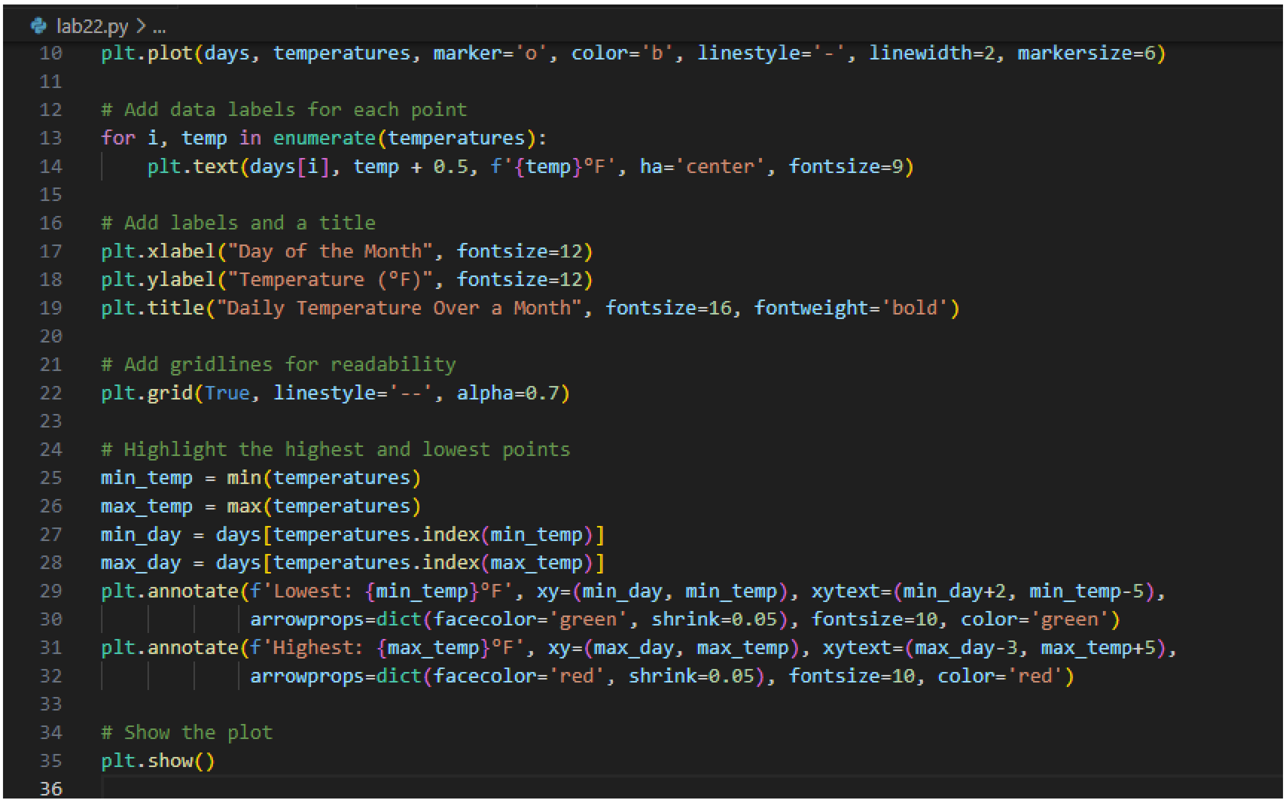
Common Customizations in Both:

1. Axis Labels: Use plt.xlabel() and plt.ylabel() to describe the data.
2. Title: Add a title with plt.title() to describe the plot.
3. Data Labels: Add numerical labels to bars or bins for clarity.
4. Gridlines: Make the plot easier to read by using plt.grid().
5. Layout: Use plt.tight_layout() to optimize spacing.

Assignment Questions:-

Ques1:- Visualize daily temperature in a specific location

Program:-



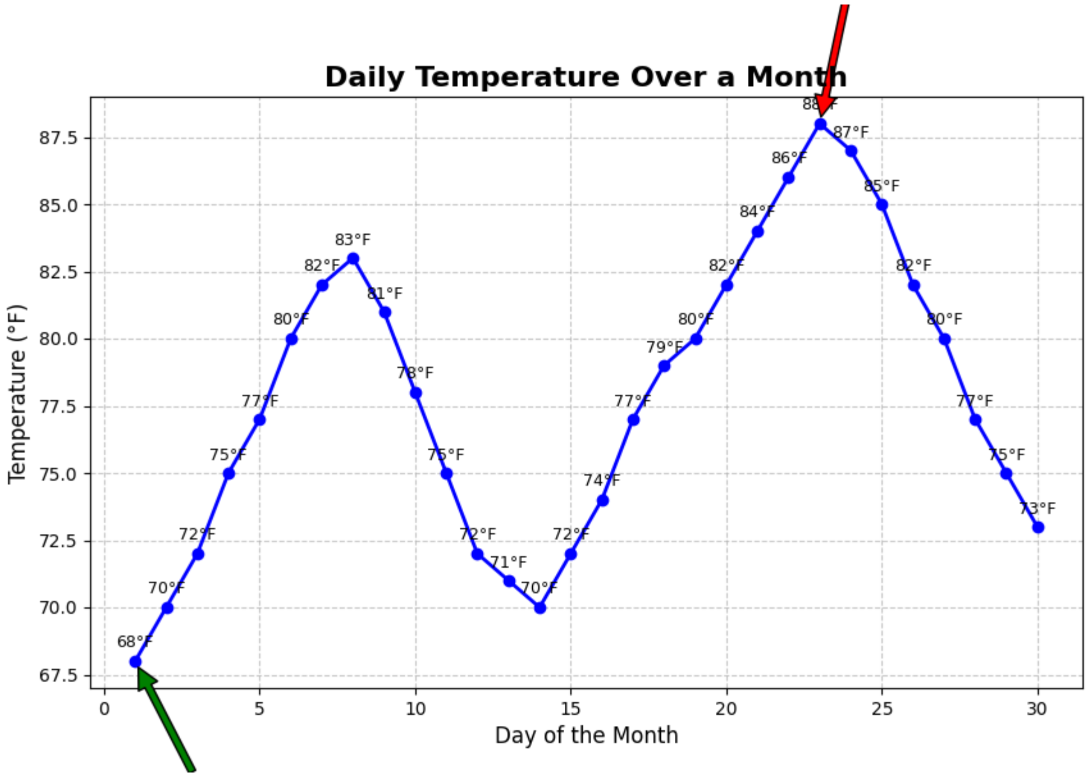
lab22.py > ...

```python
plt.plot(days, temperatures, marker='o', color='b', linestyle='-', linewidth=2, markersize=6)

# Add data labels for each point
for i, temp in enumerate(temperatures):
    plt.text(days[i], temp + 0.5, f'{temp}°F', ha='center', fontsize=9)

# Add labels and a title
plt.xlabel("Day of the Month", fontsize=12)
plt.ylabel("Temperature (°F)", fontsize=12)
plt.title("Daily Temperature Over a Month", fontsize=16, fontweight='bold')

# Add gridlines for readability
plt.grid(True, linestyle='--', alpha=0.7)

# Highlight the highest and lowest points
min_temp = min(temperatures)
max_temp = max(temperatures)
min_day = days[temperatures.index(min_temp)]
max_day = days[temperatures.index(max_temp)]
plt.annotate(f'Lowest: {min_temp}°F', xy=(min_day, min_temp), xytext=(min_day+2, min_temp-5),
             arrowprops=dict(facecolor='green', shrink=0.05), fontsize=10, color='green')
plt.annotate(f'Highest: {max_temp}°F', xy=(max_day, max_temp), xytext=(max_day-3, max_temp+5),
             arrowprops=dict(facecolor='red', shrink=0.05), fontsize=10, color='red')

# Show the plot
plt.show()
```

Key Enhancements:

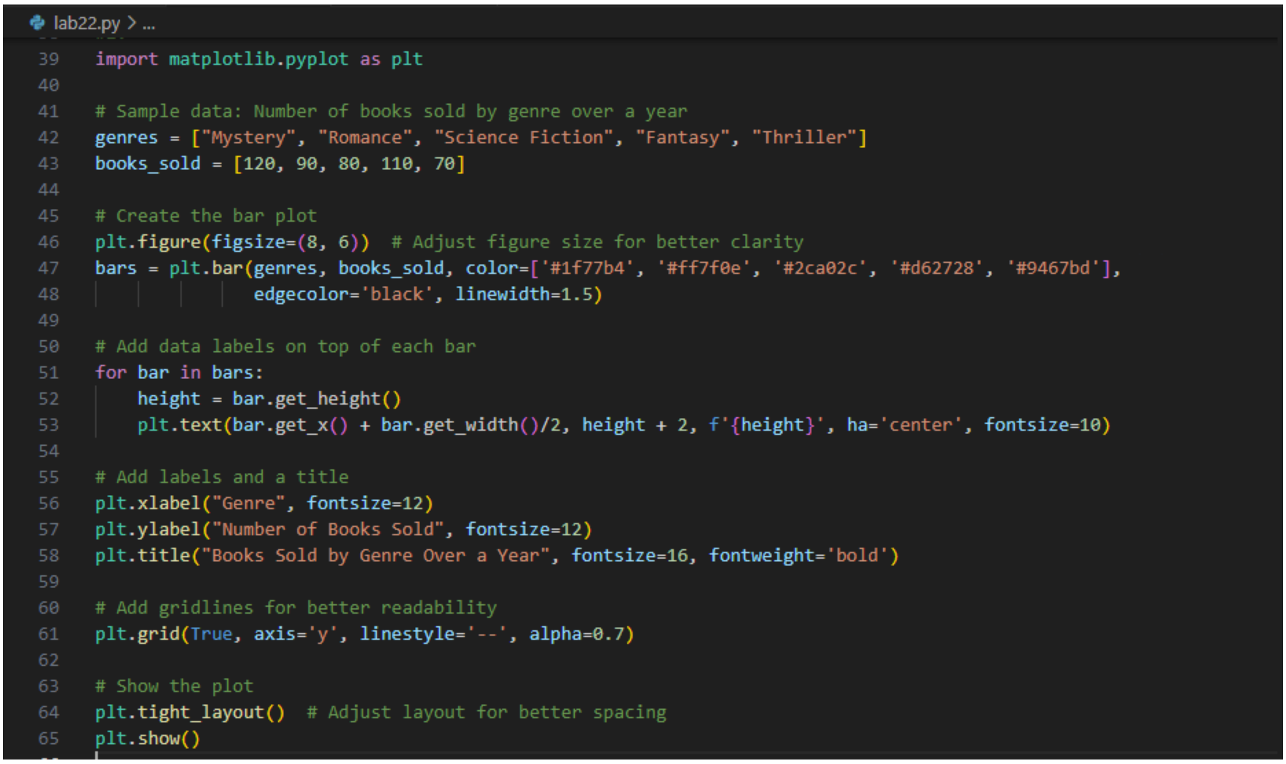
- **Data labels:** Added for each temperature point.
- **Gridlines:** Improved readability.
- **Annotations:** Highlight the highest and lowest temperatures.
- **Customization:** Adjusted line width, marker size, and font sizes for labels.

Output:-



Ques 2. Visualize the number of books sold in a bookstore by genre over a year

Program:-

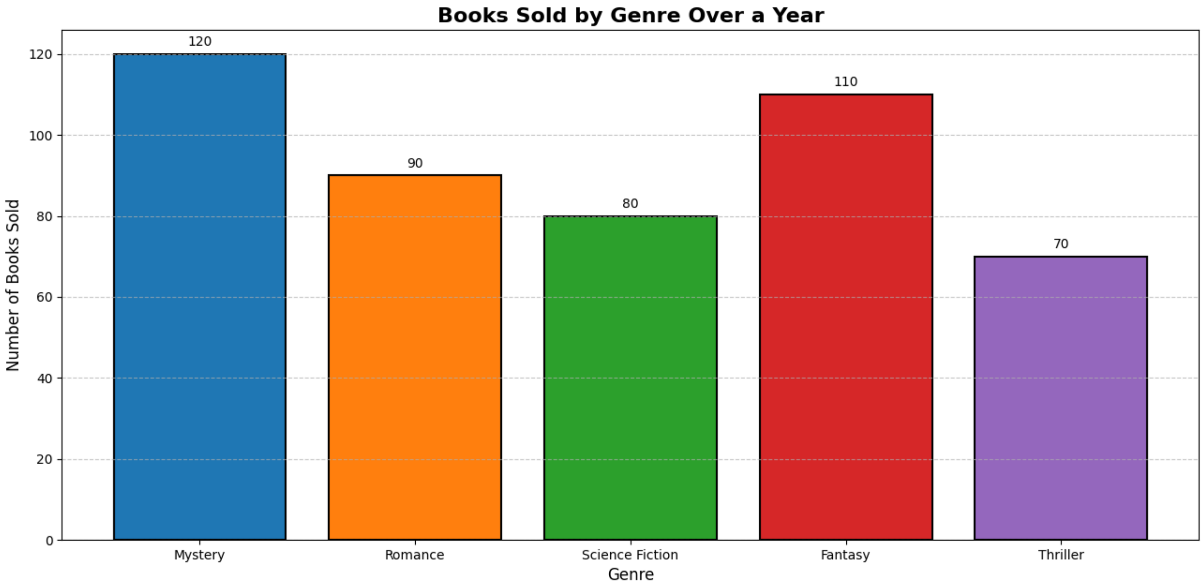


```python
import matplotlib.pyplot as plt

# Sample data: Number of books sold by genre over a year
genres = ["Mystery", "Romance", "Science Fiction", "Fantasy", "Thriller"]
books_sold = [120, 90, 80, 110, 70]

# Create the bar plot
plt.figure(figsize=(8, 6))  # Adjust figure size for better clarity
bars = plt.bar(genres, books_sold, color=['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd'],
               edgecolor='black', linewidth=1.5)

# Add data labels on top of each bar
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 2, f'{height}', ha='center', fontsize=10)

# Add labels and a title
plt.xlabel("Genre", fontsize=12)
plt.ylabel("Number of Books Sold", fontsize=12)
plt.title("Books Sold by Genre Over a Year", fontsize=16, fontweight='bold')

# Add gridlines for better readability
plt.grid(True, axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.tight_layout()  # Adjust layout for better spacing
plt.show()
```

Key Enhancements:

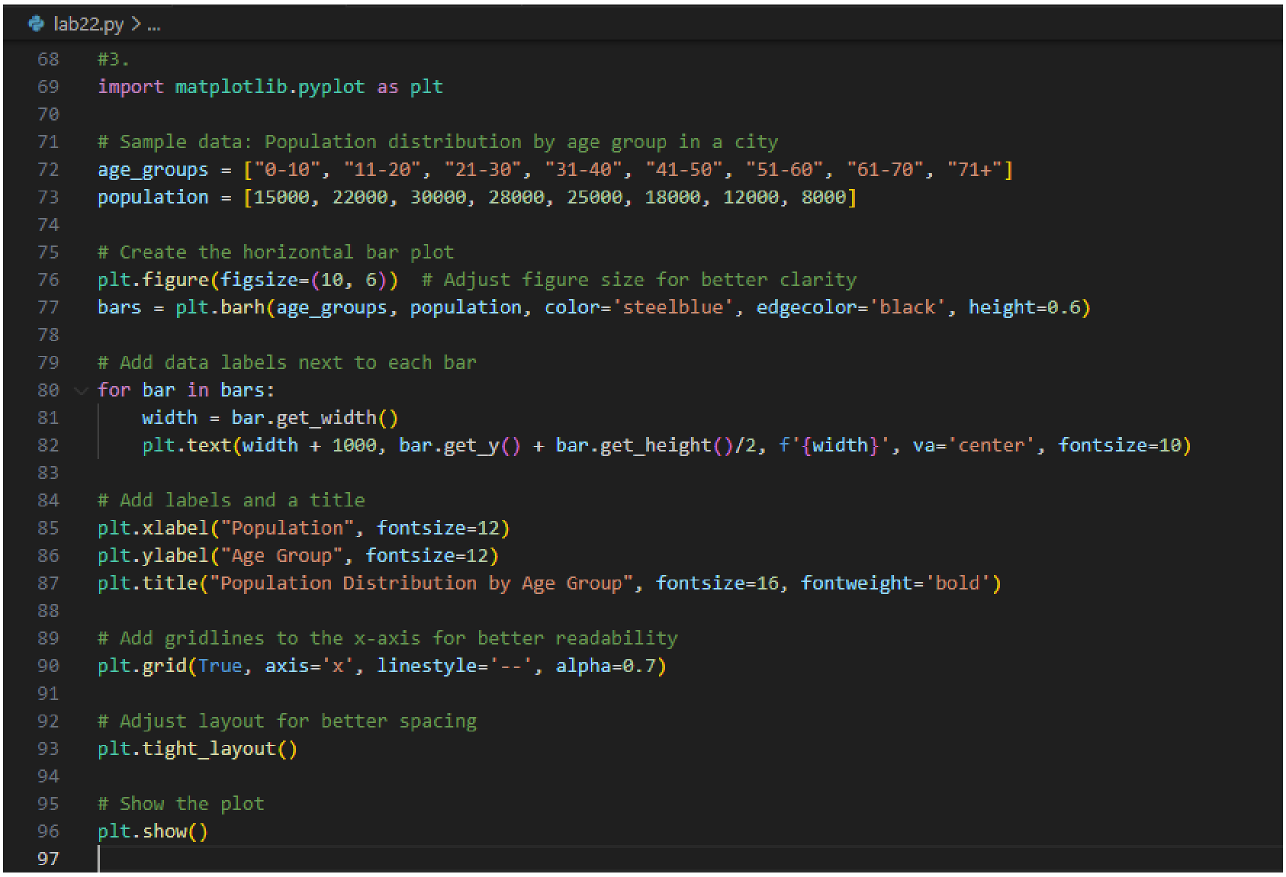
- **Data Labels: Display the number of books sold above each bar for easy reading.**
- **Color Customization: Each genre bar has a distinct color, making the chart visually appealing.**
- **Gridlines: Horizontal gridlines improve readability.**
- **Edgecolor and linewidth: Adding black edges and a thicker line around bars gives a cleaner appearance.**
- **Font Customization: Titles and labels are made clearer by adjusting the font size and style.**

Output:-



Ques 3. Horizontal Bar Plot in Python over age groups and population.

Program:-

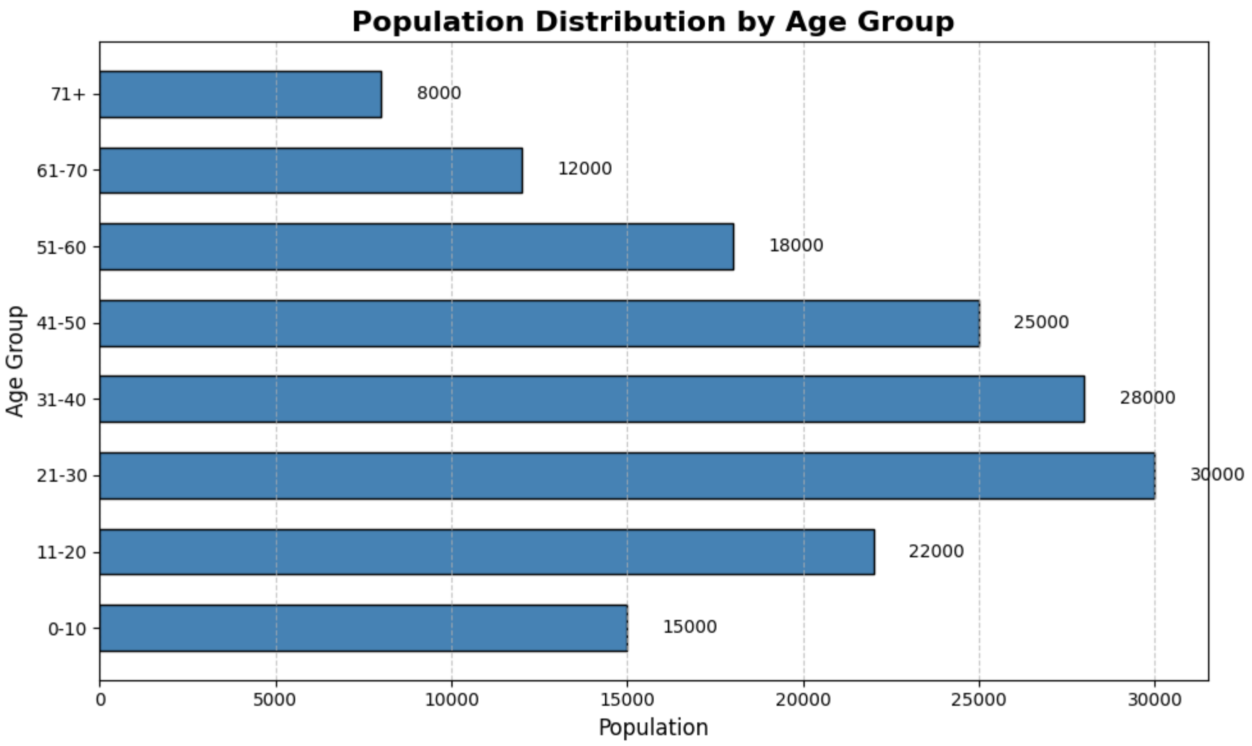


```python
#3.
import matplotlib.pyplot as plt

# Sample data: Population distribution by age group in a city
age_groups = ["0-10", "11-20", "21-30", "31-40", "41-50", "51-60", "61-70", "71+"]
population = [15000, 22000, 30000, 28000, 25000, 18000, 12000, 8000]

# Create the horizontal bar plot
plt.figure(figsize=(10, 6))  # Adjust figure size for better clarity
bars = plt.barh(age_groups, population, color='steelblue', edgecolor='black', height=0.6)

# Add data labels next to each bar
for bar in bars:
    width = bar.get_width()
    plt.text(width + 1000, bar.get_y() + bar.get_height()/2, f'{width}', va='center', fontsize=10)

# Add labels and a title
plt.xlabel("Population", fontsize=12)
plt.ylabel("Age Group", fontsize=12)
plt.title("Population Distribution by Age Group", fontsize=16, fontweight='bold')

# Add gridlines to the x-axis for better readability
plt.grid(True, axis='x', linestyle='--', alpha=0.7)

# Adjust layout for better spacing
plt.tight_layout()

# Show the plot
plt.show()

```

Key Enhancements:

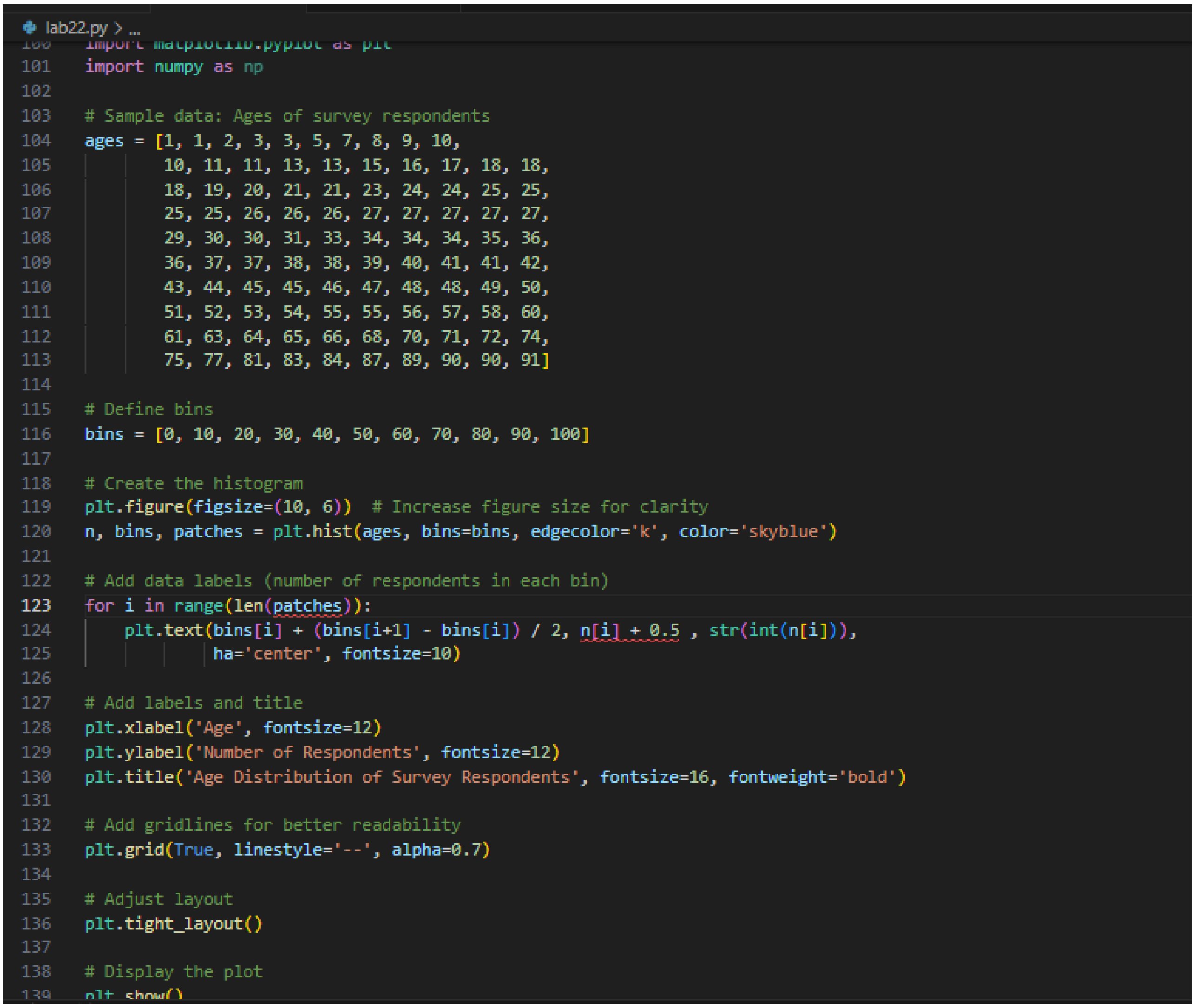
- **Data Labels:** The population values are displayed next to each bar for clear interpretation.
- **Color and Edge Customization:** A visually appealing color (steelblue) with black edges for a cleaner look.
- **Gridlines:** Horizontal gridlines enhance readability.
- **Bar Height:** The height of each bar is adjusted for visual clarity.
- **Text Positioning:** The data labels are offset slightly to avoid overlapping the bars.

Output:-



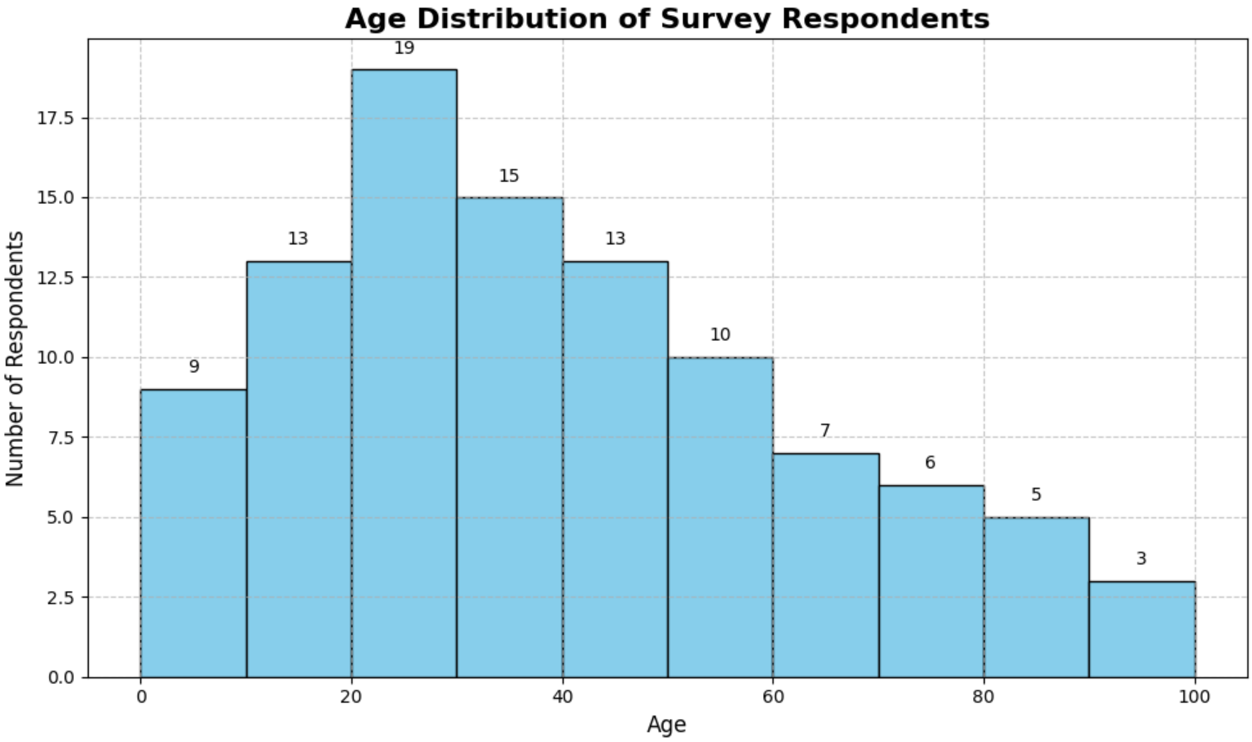
Ques 4:- Histogram of Ages of Survey Respondents.

Program:-



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

# Sample data: Ages of survey respondents
ages = [1, 1, 2, 3, 3, 5, 7, 8, 9, 10,
        10, 11, 11, 13, 13, 15, 16, 17, 18, 18,
        18, 19, 20, 21, 21, 23, 24, 24, 25, 25,
        25, 25, 26, 26, 26, 27, 27, 27, 27, 27,
        29, 30, 30, 31, 33, 34, 34, 34, 35, 36,
        36, 37, 37, 38, 38, 39, 40, 41, 41, 42,
        43, 44, 45, 45, 46, 47, 48, 48, 49, 50,
        51, 52, 53, 54, 55, 55, 56, 57, 58, 60,
        61, 63, 64, 65, 66, 68, 70, 71, 72, 74,
        75, 77, 81, 83, 84, 87, 89, 90, 90, 91]

# Define bins
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# Create the histogram
plt.figure(figsize=(10, 6))  # Increase figure size for clarity
n, bins, patches = plt.hist(ages, bins=bins, edgecolor='k', color='skyblue')

# Add data labels (number of respondents in each bin)
for i in range(len(patches)):
    plt.text(bins[i] + (bins[i+1] - bins[i]) / 2, n[i] + 0.5 , str(int(n[i])),
            ha='center', fontsize=10)

# Add labels and title
plt.xlabel('Age', fontsize=12)
plt.ylabel('Number of Respondents', fontsize=12)
plt.title('Age Distribution of Survey Respondents', fontsize=16, fontweight='bold')

# Add gridlines for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Adjust layout
plt.tight_layout()

# Display the plot
plt.show()
```

Output:-


Age Distribution of Survey Respondents
Number of Respondents
Age
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
20
40
60
80
100
9
13
19
15
13
10
7
6
5
3