Project 2 Report on:

Student Marks Histograms

By

RATNESH RANJAN

Roll – 2312res506

Project Workflow

Overview

This document outlines the workflow for the “Student Marks Histogram” project, a Python program I built to collect student marks and show their distribution in a histogram. I tried to keep things simple and clear. This workflow explains the project setup, steps I followed, deliverables, and what I learned.

Project Framework

1. **Objective**: Let users enter student marks (0-100), check if they're valid, create a histogram to show how marks are spread across ranges (0-10, 10-20, ..., 90-100), and save it as a PNG file.
2. **Tools**: Python with matplotlib.pyplot for making the histogram, Jupiter Notebook for writing code, and GitHub to share my work.
3. **Input**: Marks typed by the user in the console.
4. **Output**: A histogram saved as student_marks_histogram.png, shown on-screen.

Workflow Steps

1. Data Input Handling

Purpose: Collect marks from the user.

Approach:

1. Ask the user to type marks (0-100) one by one and enter 'done' when finished.
2. Use a while loop with input() to gather marks and store them in a list called marks.
3. Give feedback so the user knows if the input worked or if they need to try again.

Example: User types 78, 95, 62, and then 'done' to move on.

```
Enter student marks (0-100). Type 'done' when finished.
Enter a mark (or 'done' to finish): 87
Enter a mark (or 'done' to finish): 75
Enter a mark (or 'done' to finish): 89
Enter a mark (or 'done' to finish): 71
Enter a mark (or 'done' to finish): 93
Enter a mark (or 'done' to finish): 56
Enter a mark (or 'done' to finish): 67
Enter a mark (or 'done' to finish): 34
Enter a mark (or 'done' to finish): 66
```

2. Data Validation

Purpose: Make sure the marks are valid numbers and between 0 and 100.

Approach:

1. Use a nested while loop inside a try-except block to check each input.
2. Convert inputs to floats and ensure they're between 0 and 100. If not, ask again until a valid mark or 'done' is entered.
3. If the user types 'done' with no valid marks, show an error and stop the program.

Example: Typing "xyz" shows "Invalid input. Enter a number or 'done'." and 120 shows "Mark must be between 0 and 100."

```
Enter a mark (or 'done' to finish): 61
Enter a mark (or 'done' to finish): 68
Enter a mark (or 'done' to finish): 105
Mark must be between 0 and 100.
Enter a mark (or 'done' to finish): 125
Mark must be between 0 and 100.
Enter a mark (or 'done' to finish): 25
Enter a mark (or 'done' to finish): 35
```

3. Data Processing

Purpose: Got the marks ready for the histogram.

Approach:

- Set up bins for the histogram: [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100].
- Keep valid marks in a list for plotting.

Example: Marks like [78, 95, 62] are grouped into bins (78 in 70-80, 95 in 90-100, 62 in 60-70).

```
# Bin Ranges (0-10, 10-20, ..., 90-100)
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

4. Data Visualization

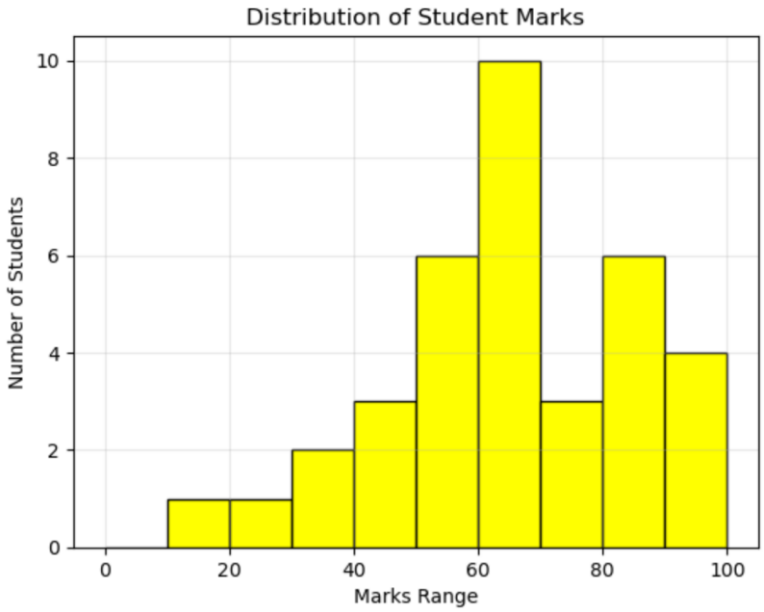
Purpose: Showed the marks distribution in a clear histogram.

Approach:

1. Use matplotlib.pyplot.hist() to create a histogram with the bins, yellow bars, and black edges.
2. Add labels ("Marks Range" for x-axis, "Number of Students" for y-axis), a title ("Distribution of Student Marks"), and a light grid (alpha=0.3).

3. Save the histogram as student_marks_histogram.png and show it on-screen with plt.show().

Example:



5. Error Handling and Code Structure

Purpose: Made the program reliable and easy to follow.

Approach:

I. Use a nested while loop with try-except to catch non-numeric inputs or marks outside 0-100, asking again until valid.

II. Check if the marks list is empty to avoid errors in plotting.

III. Organize the code into clear sections (input, validation, visualization, output) with simple variable names like marks and bins.

Example: If the user types "abc", the program says "Invalid input. Enter a number or 'done'." and keeps asking.

```
while True:
    mark = input("Enter a mark (or 'done' to finish): ")
    if mark.lower() == 'done':
        break
    try:
        mark_value = float(mark)
        if 0 <= mark_value <= 100:
            marks.append(mark_value)
        else:
            print("Mark must be between 0 and 100.")
            continue
    except ValueError:
        print("Invalid input. Enter a number or 'done'.")
        continue
```

6. Python Code of Project

In [6]:

```python
# GUVI HCL Project 2
# student_marks_histogram

import matplotlib.pyplot as plt

# Input from user
print("Enter student marks (0-100). Type 'done' when finished.")
marks = []
while True:
    mark = input("Enter a mark (or 'done' to finish): ")
    if mark.lower() == 'done':
        break
    try:
        mark_value = float(mark)
        if 0 <= mark_value <= 100:
            marks.append(mark_value)
        else:
            print("Mark must be between 0 and 100.")
            continue
    except ValueError:
        print("Invalid input. Enter a number or 'done'.")
        continue

# Check marks list empty or not
if not marks:
    print("Error: No valid marks provided. Please enter at least one valid mark.")
    exit()

# Bin Ranges (0-10, 10-20, ..., 90-100)
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# Histogram
plt.hist(marks, bins=bins, edgecolor='black', color='yellow')

# Labels and Title
plt.xlabel('Marks Range')
plt.ylabel('Number of Students')
plt.title('Distribution of Student Marks')

# Grid lines
plt.grid(True, alpha=0.3)

# Saving the plot
plt.savefig('student_marks_histogram.png')
print("\nHistogram saved as 'student_marks_histogram.png'")

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

Deliverables

1. **Python Script**: student_marks_histogram.ipynb with the full code.
2. **Visualization**: student_marks_histogram.png, the histogram output.
3. **Presentation**: A file (Student_Marks_Histogram_Presentation.pdf) with PDF about the project goal, workflow, code, histogram, and submission details.

Conclusion

This project was a great way for me to learn Python and Matplotlib as a beginner! It collects student marks, checks them carefully, and shows a neat histogram to see how they're spread out. I worked to handle errors, like when someone types the wrong thing, and made sure the code is clear.

Source Code GitHub Repository link :

https://github.com/ratneshranjan484/GUVI-HCL-Project-2-Student-Marks-Histogram

Note :

Source file (student_marks_histogram.ipynb) is in Jupiter NoteBook file. So, Use Jupiter NoteBook run the program.