Summarizing and report about my approach

Task 1: Python Basics and Control Structures

Task 1.1: Palindrome Check

Approach:
- The program prompts the user to input a string.
- It then checks if the string reads the same forward and backward by comparing the string with its reverse (`[::-1]`).
- An appropriate message is displayed indicating whether the string is a palindrome.

Example:

.If the input is "civic," the program will output: "The string is a palindrome"

Challenges:
- Ensuring the program handles case sensitivity (e.g., "Madam" vs. "madam") and ignores spaces or special characters for phrases.

Task 1.2: Factorial Calculation Using Recursion

Approach:
- A recursive function calls itself until a base case (factorial of 1) is reached.
- The function takes a number as input, multiplies it by the factorial of the previous number, and continues until it reaches 1.
- The factorial of a number `n` is the product of all integers from 1 to `n`.

Example:
- For an input of 5, the function will return 5! = 5 × 4 × 3 × 2 × 1 = 120.

Challenges:
- Understanding recursion and ensuring a proper base case (when the number reaches 1) to avoid infinite recursion.

Task 2: Data Structures

Task 2.1: Contact Book Management

Approach:
- The user can add contacts by entering a name and phone number.
- The program allows searching for a contact by name and will display the phone number if the contact exists.
- It also displays all stored contacts in alphabetical order based on the name.
- ***Outcome***: The contact book enables efficient contact management with features for adding, searching, and listing contacts alphabetically

Challenges:-
Handling duplicate contact entries and ensuring case-insensitive search functionality.

Task 2.2: Student Grades Sorting

Approach:

- A list of tuples is created, where each tuple contains a student's name and their grade.
- The list is sorted by grade in descending order using Python's built-in `sorted()` function.
- The list is sorted by the second element in each tuple (the grade) in (descending) order (highest grades first).

- The sorted list is then printed.
  - After sorting, the program displays the list of students, now ordered by their grades from highest to lowest.

Challenges:

- Ensuring the list is correctly sorted in descending order, particularly when grades are tied or students have similar names.

Task 3: Libraries and File Handling

Task 3.1: Word Frequency Counter

Approach:

- The program reads a text file where each line contains one word.
- It counts the frequency of each word using a dictionary and writes the results to a new CSV file with two columns: "Word" and "Frequency."
- Count how many times each word appears in a text file and save the results in a CSV file.
- ***Outcome***: You'll get a CSV file that lists each word and its frequency of occurrence in the original text file.

Challenges:

- Handling case sensitivity and ensuring proper formatting of the output in the CSV file.

Task 3.2: Sales Data Analysis

- ***Objective***: Analyze sales data to calculate total sales for each product.
- ***Approach:***
- Using Pandas, the program reads a CSV file containing sales data (columns: "Product", "Quantity Sold", "Price").

- **Read the CSV file:** The program loads a file with columns: "Product", "Quantity Sold", and "Price".

  Calculate total sales: For each product, the program multiplies "Quantity Sold" by "Price" to get the total sales.

  Save the results: It saves the total sales data to a new CSV file.
- **Outcome**: The output CSV file will contain the product names and their corresponding total sales amounts.

- challenges:- Ensuring correct handling of missing or invalid data in the CSV file before calculating total sales.

Task 4: Data Analysis and Visualization

Task 4.1: Employee Salary Analysis

- ***Objective:***

Analyze and clean employee salary data and calculate the average salary by department.

Approach:

- **Load the dataset** :
  The program reads the employee data into a Pandas DataFrame.
- **Clean the data**:
  It removes any rows that have missing or incomplete values.
- **Group by department**:
  The program groups the employees by their department and calculates the average salary for each department.
- **Save to CSV**:
  The cleaned and grouped data, with the average salaries per department, are saved to a new CSV file.
- **Outcome**:
  A CSV file showing the department names and their average salaries will be generated.

Challenges:

- Handling missing data, ensuring the accuracy of salary calculations, and proper grouping by department.

Task 4.2: Visualization Dashboard

Objective:

Create a dashboard with three different types of visualizations using Matplotlib and Seaborn.

Approach:

Line Plot: Sales Trend Over the Past Year

- Explanation:
- A line plot will display how sales have changed over time across 12 months.
- The x-axis represents months, while the y-axis shows the sales figures (using made-up data).
- This plot helps to visualize trends, such as increases or decreases in sales over the year.

Bar Plot: Employee Count by Department

- **Explanation**:
- A bar plot will compare the number of employees across different departments.

- The x-axis will show the department names, and the y-axis will display the number of employees in each department.
- This plot is useful for visualizing how many employees work in each department and identifying the largest or smallest departments.

Scatter Plot: Relationship Between Employee Age and Salary
- **Explanation**:
  - A scatter plot will show the relationship between employees' ages and their salaries.
  - The x-axis represents employee ages, and the y-axis represents their salaries.
  - This plot helps to identify any potential correlation between age and salary, such as whether older employees tend to earn more.

The dashboard will give a clear overview of sales trends, department sizes, and how age relates to salary using simple yet effective visualizations.

<u>Conclusion</u>

This project covered key areas of Python programming, from basic control structures to advanced data handling.

1. Task 1-- focused on fundamental concepts like string manipulation (palindrome check) and recursion (factorial calculation), helping to reinforce Python basics.

2. Task 2-- explored data structures, emphasizing the use of dictionaries for contact management and list sorting techniques for student grades. These tasks improved skills in organizing and manipulating structured data.

3. Task 3-- involved file handling and data analysis with libraries like Pandas. Tasks such as word frequency counting and sales data analysis demonstrated the practical use of external libraries in processing and analyzing data.

4. Task 4-- required advanced data analysis, including cleaning and grouping data, to perform employee salary analysis. It highlighted the importance of Pandas in extracting insights from datasets.

Overall, these tasks provided a solid foundation in Python, enhancing your ability to manage, analyze, and visualize data efficiently.

<u>THANKYOU</u>
Summeraized and reported by : - Ezedin abdellah (candidate programmer)