**Week 1: Python Data Structures**

**Session 1: Tuples & Sets**

📌 **Topics Covered:**

* Tuples: Characteristics, immutability, use cases.
* Sets: Uniqueness property, operations (union, intersection, difference).
* Practical examples and real-world applications.

📖 **Tasks (10 Questions):**

1. Create a tuple with 5 elements and try modifying it. What happens?
2. Convert a tuple into a list and modify an element.
3. Create two sets and perform union, intersection, and difference operations.
4. Remove duplicate values from a list using a set.
5. Find the common elements in two sets.
6. Use a set to store unique words from a sentence.
7. Write a program to check if a given value exists in a tuple.
8. Merge two tuples and sort the result.
9. Find the length of a tuple without using len().
10. Explain when to use a tuple instead of a list.

**Session 2: Dictionaries**

📌 **Topics Covered:**

* Dictionary creation, accessing values.
* Dictionary methods (keys, values, items, get, pop, update).
* Nested dictionaries and real-world applications.

📖 **Tasks (10 Questions):**

1. Create a dictionary with student names as keys and marks as values.
2. Retrieve a value using a key and handle missing keys with .get().
3. Iterate over keys, values, and items of a dictionary.
4. Create a nested dictionary for an employee database.
5. Merge two dictionaries into one.
6. Sort a dictionary by keys and values.
7. Remove an item from a dictionary safely.
8. Find the most frequently occurring element in a dictionary.
9. Convert two lists into a dictionary (one as keys, another as values).
10. Explain a real-world use case for dictionaries.

**Project for Week 1:**

* **Goal:** Implement all data structures (list, tuple, set, dictionary) in a single project.
* **Task:** Build a **Contact Book** where users can add, update, delete, and search contacts using different data structures.

**Week 2: File Handling & Object-Oriented Programming**

**Session 3: File Handling (Reading/Writing, JSON Handling)**

📌 **Topics Covered:**

* Reading & writing text files (open(), read(), write()).
* Working with CSV files.
* JSON handling (json.dumps(), json.loads()).

📖 **Tasks (10 Questions):**

1. Read a text file and count the number of words.
2. Write a list of names to a file.
3. Append new data to an existing file.
4. Read a CSV file and extract a specific column.
5. Convert a Python dictionary to JSON format and save it.
6. Read a JSON file and extract specific keys.
7. Count the occurrences of each word in a file.
8. Search for a specific word in a file and display the line number.
9. Copy the contents of one file to another.
10. Implement exception handling in file operations.

**Session 4: OOP - Classes & Objects**

📌 **Topics Covered:**

* Creating classes and objects.
* Instance and class variables.
* Methods (\_\_init\_\_ constructor).

📖 **Tasks (10 Questions):**

1. Create a Car class with attributes brand, model, and year.
2. Implement a method in the Car class to display details.
3. Create multiple objects of a class and print their details.
4. Use the \_\_init\_\_ method to initialize objects dynamically.
5. Implement getter and setter methods.
6. Create a BankAccount class with deposit and withdrawal methods.
7. Create a Rectangle class with a method to calculate the area.
8. Implement a method to check whether an object belongs to a particular class.
9. Explain the difference between instance and class variables.
10. Create a class with a default value for an attribute and modify it later.

**Session 5: OOP - Inheritance & Polymorphism**

📌 **Topics Covered:**

* Inheritance: Single and multiple.
* Method Overriding & Polymorphism.
* Abstract classes & interfaces.

📖 **Tasks (10 Questions):**

1. Create a Vehicle class and extend it into Car and Bike.
2. Implement method overriding in a subclass.
3. Create a base class with an abstract method and implement it in a subclass.
4. Use super() to call a parent class method from a subclass.
5. Implement multiple inheritance with a practical example.
6. Create a class with a method that can accept multiple data types.
7. Implement method overloading (using default parameters).
8. Explain the difference between overriding and overloading.
9. Demonstrate how polymorphism works with different classes.
10. Implement a class factory method that returns different objects based on input.

**Project for Week 2:**

* **Goal:** Implement a Student Management System using OOP.
* **Task:** Create classes for Student, Course, and Teacher and manage enrollments using file handling.

**Week 3: NumPy & Pandas + Data Cleaning**

📌 **Objective:** Learn how to manipulate and clean datasets using NumPy and Pandas efficiently.

**Session 6: Introduction to NumPy**

📌 **Topics Covered:**

* What is NumPy and why use it?
* Creating NumPy arrays (1D, 2D, 3D).
* Indexing, slicing, and reshaping arrays.
* Mathematical operations on arrays.

📖 **Tasks (10 Questions):**

1. Create a 1D and 2D NumPy array and print its shape and size.
2. Convert a Python list into a NumPy array.
3. Extract the first three elements from a NumPy array.
4. Reshape a 1D array into a 2D array.
5. Perform element-wise addition and multiplication on two arrays.
6. Find the maximum, minimum, and mean of an array.
7. Create an identity matrix using NumPy.
8. Generate an array of random numbers between 0 and 1.
9. Stack two arrays horizontally and vertically.
10. Replace all even numbers in an array with -1.

**Session 7: Advanced NumPy Operations**

📌 **Topics Covered:**

* Broadcasting and vectorized operations.
* Statistical functions (mean, median, standard deviation).
* Boolean indexing and filtering.
* Performance optimization using NumPy.

📖 **Tasks (10 Questions):**

1. Use broadcasting to multiply a 1D array with a 2D array.
2. Compute the standard deviation and variance of an array.
3. Use Boolean indexing to filter out negative values from an array.
4. Find the index of the maximum value in an array.
5. Generate an array of 10 random integers and sort them.
6. Compute the cumulative sum of an array.
7. Find unique elements in an array.
8. Check if any element in an array is greater than 50.
9. Normalize an array to have values between 0 and 1.
10. Compare NumPy operations with traditional Python loops (performance test).

**Session 8: Introduction to Pandas**

📌 **Topics Covered:**

* Introduction to Pandas Series & DataFrames.
* Reading/Writing CSV and Excel files.
* Selecting, filtering, and modifying DataFrames.

📖 **Tasks (10 Questions):**

1. Create a Pandas Series from a Python list.
2. Convert a dictionary into a Pandas DataFrame.
3. Read a CSV file into a DataFrame and display the first 5 rows.
4. Select a specific column from a DataFrame.
5. Filter rows where a column value is greater than 50.
6. Add a new column to an existing DataFrame.
7. Delete a column from a DataFrame.
8. Find the number of missing values in each column.
9. Rename column names in a DataFrame.
10. Save a modified DataFrame back to a CSV file.

**Session 9: Data Cleaning Techniques**

📌 **Topics Covered:**

* Handling missing values (fill, drop).
* Removing duplicates.
* Outlier detection and handling.
* Data transformation (grouping, merging, reshaping).

📖 **Tasks (10 Questions):**

1. Fill missing values in a dataset with the column mean.
2. Drop rows where any column has missing data.
3. Remove duplicate rows from a dataset.
4. Replace null values with a placeholder value.
5. Group data by a categorical column and compute aggregate statistics.
6. Merge two DataFrames on a common column.
7. Convert a categorical column into numerical format.
8. Find the percentage of missing data in each column.
9. Identify and remove outliers from a dataset.
10. Create a pivot table from a DataFrame.

**Project for Week 3:**

📌 **Goal:** Work on a **real-world dataset** and apply data cleaning techniques.  
📌 **Task:**

* Download a dataset (e.g., **customer sales data**).
* Identify and handle missing values, duplicates, and outliers.
* Transform the dataset into a clean format suitable for visualization.

**Week 4: Data Visualization with Matplotlib & Seaborn**

📌 **Objective:** Learn to visualize data effectively for better insights.

**Session 10: Introduction to Matplotlib**

📌 **Topics Covered:**

* Basic plots (line, bar, scatter, histogram).
* Customizing plots (labels, titles, colors).
* Multiple plots in one figure.

📖 **Tasks (10 Questions):**

1. Plot a simple line chart using Matplotlib.
2. Create a bar chart for a dataset containing monthly sales.
3. Generate a scatter plot with random data points.
4. Customize a plot by adding labels and a title.
5. Change the color and style of a line plot.
6. Display multiple plots in a single figure.
7. Save a plot as an image file.
8. Add gridlines to a chart.
9. Change the size of a figure.
10. Create a histogram to display age distribution.

**Session 11: Seaborn for Advanced Visualizations**

📌 **Topics Covered:**

* Introduction to Seaborn.
* Heatmaps, pair plots, and violin plots.
* Styling and customizing plots.

📖 **Tasks (10 Questions):**

1. Load a dataset and display a pair plot.
2. Create a heatmap using correlation data.
3. Plot a violin plot to visualize distribution.
4. Generate a box plot for a dataset.
5. Change Seaborn’s default theme.
6. Use a swarm plot to display categorical data.
7. Combine Matplotlib and Seaborn in a single figure.
8. Plot a regression line between two variables.
9. Display data distribution using a KDE plot.
10. Add annotations to a Seaborn plot.

**Session 12: Customizing and Optimizing Data Visualizations**

📌 **Topics Covered:**

* Combining multiple visualizations.
* Interactive visualizations with Plotly.
* Best practices for data storytelling.

📖 **Tasks (10 Questions):**

1. Combine a bar chart and a line chart in one figure.
2. Create a subplot layout with different charts.
3. Add interactivity to a Matplotlib chart.
4. Customize fonts, colors, and grid lines in a visualization.
5. Add a legend to a Seaborn plot.
6. Optimize a large dataset for faster visualization.
7. Implement real-time data visualization.
8. Save a Seaborn plot with high resolution.
9. Choose the best visualization type for a given dataset.
10. Explain a real-world use case of data visualization.

**Project for Week 4:**

📌 **Goal:** Create meaningful **interactive visualizations**.  
📌 **Task:**

* Choose a dataset (e.g., **global temperature trends, stock market data, or COVID-19 cases**).
* Create at least **5 different types of charts** using Matplotlib and Seaborn.
* Optimize the charts for better readability and presentation.

.