# Python & Data Tools (10 Questions)

**Instructions:**

• Solve each task using Python. Include code, outputs (tables/plots), and 1–3 sentences interpreting results where applicable.

• You may use seaborn built-in datasets (tips, iris, flights, penguins) or provide small synthetic data where requested.

• Submit a single Jupyter notebook (.ipynb) with executed cells and a PDF or Word report if requested.

**1. 1. String and List Manipulation**

Write a function `normalize\_name(name: str) -> str` that trims extra spaces, capitalizes each part of the name, and returns the cleaned full name.  
  
Then given a list of full names, produce a dictionary that maps each last name to how many times it appears.

**2. 2. Control Flow and File I/O**

Write a script that reads a text file `sentences.txt` (assume one sentence per line), counts how many sentences contain the word 'data' (case-insensitive), and writes those sentences to `data\_sentences.txt`.  
  
If the file is missing, handle the exception and print a friendly message.

**3. 3. Functions and Exception Handling**

Implement a function `parse\_int\_list(s: str) -> list[int]` that takes a comma-separated string and returns a list of integers. The function should skip non-integer tokens and log a warning for each skipped token.  
  
Demonstrate with the input: '10, 5, abc, 7, 9.5, -3'.

**4. 4. Object-Oriented Programming**

Design a class `Student` with attributes `name`, `grades` (list of floats). Implement methods:  
• `add\_grade(g)` to add a grade (0–100) with validation,  
• `average()` to return the average grade,  
• `letter\_grade()` to return A/B/C/D/F based on average.  
  
Create three Student instances and print a short report (name, average, letter).

**5. 5. Inheritance & Polymorphism**

Create an abstract base class `Vehicle` with a method `num\_wheels()` and subclass it with `Car` and `Motorbike`. Implement `num\_wheels()` appropriately.  
  
Write a function `total\_wheels(vehicles: list)` that computes the total number of wheels in a list of Vehicles and demonstrate with multiple instances.

**6. 6. NumPy — array manipulation**

Using NumPy, create an array of shape (8, 5) containing integers 1..40. Then:  
• Reshape it to (4,10) and print the result.  
• Compute the mean of each column of the original array.  
• Standardize each column (z-score) and show the first 3 rows of the standardized array.

**7. 7. NumPy — random data & eigen decomposition**

Generate a 200×3 dataset `X` from a multivariate normal distribution (choose a covariance matrix with some correlation). Compute the covariance matrix, then find eigenvalues and eigenvectors. Report the explained variance ratio of the first principal component.

**8. 8. pandas — cleaning & aggregation**

Load the `tips` dataset from seaborn. Perform the following:  
• Display the first 5 rows and count missing values per column.  
• Create a new column `tip\_pct = tip / total\_bill \* 100`.  
• Produce a pivot table of average `tip\_pct` with `day` as rows and `time` as columns and save it to `avg\_tip\_pct.csv`.

**9. 9. Matplotlib — plots & annotation**

Create a figure with two subplots (1×2):  
• Left: plot `y = sin(x)` for x in [0, 2π] and mark local maxima with red markers and annotate the first maximum with coordinates.  
• Right: generate noisy linear data `y = 3x + noise` (100 points), plot the scatter and the least-squares fit line. Add axis labels and a legend. Save the figure as `exercise\_plots.png`.

**10. 10. Seaborn — exploratory visualization**

Using the `iris` dataset create:  
• A pairplot colored by species.  
• A heatmap of the correlation matrix of numeric features.  
• A violin plot (with swarm overlay) of one selected numeric feature grouped by species.  
  
For each plot, write one sentence describing a notable observation (separation, correlation, etc.).

Submission: Provide a single Jupyter notebook with all code and outputs. Include any saved files (CSV/PNG) in a submission folder.