Practical exercise for Custom package

Exercise 1: Create a Simple Package

**Objective**: Create a custom package with modules for math operations and string manipulations.

**Instructions**:

1. **Create a Directory**: Create a directory named my\_package.
2. **Add an \_\_init\_\_.py File**: Create an empty \_\_init\_\_.py file in the my\_package directory.
3. **Create a Module for Math Operations**:
   * Create a file named math\_utils.py in the my\_package directory.
   * Define two functions in math\_utils.py: add(a, b) and subtract(a, b).

**Create a Module for String Manipulations**:

* Create a file named string\_utils.py in the my\_package directory.
* Define two functions in string\_utils.py: capitalise\_words(text) and reverse\_string(text).

**Use the Package**:

* Create a new Python script or Jupyter notebook.
* Import and use the custom package modules to perform operations.

Exercise 2: Expand the Package

**Objective**: Extend the custom package by adding more functionality and demonstrate importing specific functions.

**Instructions**:

1. **Add a New Function in math\_utils.py**:

* Add a function named multiply(a, b).

1. **Add a New Function in string\_utils.py**:

* Add a function named lowercase(text).

1. **Use the Extended Package**:

* Modify the Python script or Jupyter notebook to import specific functions and use the new functionality.

Exercise 3: Creating a Custom Package for Data Analysis

Objective

The exercise will enable you to learn about the creation, structure, and usage of a custom package named **data\_analysis**, which includes modules for data loading, data cleaning, and data visualisation.

A) Create the Package Directory and Files.

Create a custom package named **data\_analysis** with the following structure:

**data\_analysis**: The main package directory.

* \_\_init\_\_.py: To initialise the package.
* load.py: Module for loading data.
* clean.py: Module for cleaning data.
* visualise.py: Module for visualising data.

B) Implement the Modules

* Implement functions to load data from CSV files. (load.py).
* Implement functions to clean the data (clean.py).
* Implement functions to visualise the data (visualise.py).

C) Use the Package

Create a separate script (main.py) to use the **data\_analysis** package and run the script main.py

Exercise 3 solution:

A)

A screenshot of a computer screen

Description automatically generated

B) I could edit each of the files using the **nano** command. However, I have opened my .py files in Spyder to edit them (you can use any Python IDE). Remember how to access your package from the previous exercise.

A screenshot of a computer

Description automatically generated

C) **Implement functions to clean the data (clean.py).** Edit the **clean.py** file to ensure that any missing data is dropped, and data is normalised. **Normalisation** is a preprocessing step that scales the values of a column to have a mean of 0 and a standard deviation of 1. This process is often used in data analysis and machine learning to ensure that different features contribute equally to the analysis.

A screen shot of a computer code

Description automatically generated

D) **Implement functions to load data from CSV files (load.py)**. Edit **load.py** to load the data from the CSV file.

A screen shot of a computer

Description automatically generated

E) **Implement functions to visualise the data (visualise.py)**. Edit visualise.py

* import matplotlib.pyplot and seaborn.
* Create a function for histogram.
* Create a function for a scatter plot.

A computer screen shot of a program

Description automatically generated

F) Create your **main.py** file using the **touch** command in Linux or directly from your Python IDE. Make sure it is saved in the same location as the other files. In main.py:

* Add the path to the custom data\_analysis package
* Import the custom package modules
* Load the data (you can use [this file](https://gla-my.sharepoint.com/personal/mireilla_bikangaada_glasgow_ac_uk/Documents/ProgSD2024-2025/sample_data1.csv)).
* Clean the data (call clean.py).
* Visualise the data that is not normalised.
* Normalise the data (call clean.py)
* Visualise the normalised data (call visualise.py).

A screen shot of a computer program

Description automatically generated

E)Outputs

Visualise without/with normalisation

|  |  |
| --- | --- |
| Without Normalisation | With Normalisation |
| A graph of age and age  Description automatically generated |  |
|  |  |
|  |  |
|  |  |