Practical 3 : Designing a Python Module for Summary Statistics

Name : *Deepak Kumar Bharti*

Roll No. : 25056765013

Group : *A*

Problem Statement

Solve the following problems using Python. Do not use NumPy, pandas, or the statistics module unless explicitly mentioned.

Given Dataset

The following data represent the number of hours studied by a group of students. The dataset contains missing values.

```
study_hours = [2, 4, 5, None, 3, 6, 4, 5, 7, None, 3, 4, 6, 5, 7, 7, 3, 4, 3, 4, 5, 6]
```

Problem 1 : Data Cleaning Using Functions

(a) Store the given data in a Python variable.

(b) Write a user-defined function 'clean_data(data)' that removes missing values (None) from the dataset.

(c) Apply the function to 'study_hours' and store the cleaned data in a new variable.

In [1]:
```
# (a) Storing the given data (i.e. List containing the number of hours studied by students) in a Python variable.
study_hours = [2, 4, 5, None, 3, 6, 4, 5, 7, None, 3, 4, 6, 5, 7, 7, 3, 4, 3, 4, 5, 6]

# Displaying the original dataset
print("Original Dataset :", study_hours)
```

```
Original Dataset : [2, 4, 5, None, 3, 6, 4, 5, 7, None, 3, 4, 6, 5, 7, 7, 3, 4, 3, 4, 5, 6]
```

In [2]:
```
# (b) Writing a user-defined function 'clean_data(data)' that removes missing values (None) from the dataset.
def clean_data(data):
    cleaned_data = []  # Empty list to store valid values

    # Loop through each element in the input list
    for value in data:
        # Check if the value is not None
        if value is not None:
```

```
            cleaned_data.append(value)  # Add valid value to the list
    return cleaned_data
```

In [3]:
```
# (c) Applying the function to 'study_hours' and store the cleaned data in a new v
ariable.
cleaned_study_hours = clean_data(study_hours)

# Display the cleaned dataset
print("Cleaned Datasetv :", cleaned_study_hours)
```

```
Cleaned Datasetv : [2, 4, 5, 3, 6, 4, 5, 7, 3, 4, 6, 5, 7, 7, 3, 4, 3, 4, 5, 6]
```

Problem 2 : Using For Loop and List Comprehension

Using the cleaned dataset obtained in Problem 1 :

(a) Using a for loop, compute the square of each element and store the result in a new list.

(b) Using list comprehension, compute the square of each element in a single line of code.

(c) Display both lists.

(d) Briefly comment on the advantages of list comprehension over traditional loops.

In [4]:
```
# (a) Computing the square of each element and store the result in a new list usin
g a for loop.
# Empty list to store squares
squares_using_loop = []

# Loop through each value in cleaned data
for value in cleaned_study_hours:
    square = value ** 2          # Square of the value
    squares_using_loop.append(square)  # Add result to list
```

In [5]:
```
# (b) Computing the square of each element in a single line of code using list com
prehension.
squares_using_comprehension = [value ** 2 for value in cleaned_study_hours]
```

In [6]:
```
# (c) Displaying both lists.
print("Squares using for loop:")
print(squares_using_loop)

print("\nSquares using list comprehension:")
print(squares_using_comprehension)
```

```
Squares using for loop:
[4, 16, 25, 9, 36, 16, 25, 49, 9, 16, 36, 25, 49, 49, 9, 16, 9, 16, 25, 36]

Squares using list comprehension:
[4, 16, 25, 9, 36, 16, 25, 49, 9, 16, 36, 25, 49, 49, 9, 16, 9, 16, 25, 36]
```

In [7]:
```
#(d) The advantages of list comprehension over traditional loops.
print("""
Advantages of List Comprehension:
1. Requires fewer lines of code
2. Easier to read for simple operations
```

```
3. Faster execution compared to traditional loops
""")
```

```
Advantages of List Comprehension:
1. Requires fewer lines of code
2. Easier to read for simple operations
3. Faster execution compared to traditional loops
```

Problem 3 : Creating a User-Defined Mean Function

(a) Write a user-defined function 'mean(data)' to compute the arithmetic mean of the cleaned dataset.

(b) The function should :

- Accept a list as input
- Return the mean value

(c) Use the function to compute and display the mean study hours.

In [8]:
```
# (a) Writing a user-defined function 'mean(data)' to compute the arithmetic mean
of the cleaned dataset.

# (b) The function will :
#       * Accept a list as input
#       * Return the mean value

def mean(data):

    total = 0   # Variable to store sum of values
    count = 0   # Variable to store number of elements

    # Loop through each element in the list
    for value in data:
        total += value
        count += 1

    # Mean = Sum of values / Number of values
    return total / count
```

In [9]:
```
# (c) Using the function to compute and display the mean study hours.
mean_value = mean(cleaned_study_hours)
print("Mean Study Hours:", mean_value)
```

```
Mean Study Hours: 4.65
```

Problem 4 : Storing the Mean Function as a Module

(a) Create a Python module named : $summary\_basic.py$

(b) Move the mean(data) function into this module.

(c) Import the module and use the mean() function to compute the mean of the cleaned dataset.

In [10]:
```
# (a) Created a Python module named : 'summary_basic.py' in the same destination
# (b) Moved the mean(data) function into this module.
```

In [11]:
```
# (c) Importing the module and using the mean() function to compute the mean of th
e cleaned dataset.
import summary_basic

mean_from_module = summary_basic.mean(cleaned_study_hours)
print("Mean using summary_basic module:", mean_from_module)
```

```
Mean using summary_basic module: 4.65
```

Problem 5 : Creating Additional Summary Statistics Functions as a Module

Extend the module summary_basic.py by adding the following separate user-defined functions:

(a) 'median(data)' – returns the median of the dataset.

(b) 'mode(data)' – returns the mode of the dataset. If more than one mode exists, return any one of them.

(c) 'std_dev(data)' – returns the sample standard deviation of the dataset.

(d) Each function should :

- Work only on cleaned numeric data
- Not rely on external statistical libraries

In [12]:
```
# Extended the module 'summary_basic.py' by adding the following separate user-def
ined functions:
# (a) 'median(data)' – returns the median of the dataset.
# (b) 'mode(data)' – returns the mode of the dataset. If more than one mode exist
s, return any one of them.
# (c) 'std_dev(data)' – returns the sample standard deviation of the dataset.
# (d) Each function will :
#       * Work only on cleaned numeric data
#       * Not rely on external statistical libraries
```

Problem 6 : Using the Summary Statistics Module

(a) Import the updated summary_basic module in a new Python file.

(b) Using the functions from the module, compute and display:

- Mean
- Median
- Mode
- Standard Deviation

(c) Display the results clearly with appropriate labels.

In [13]:
```
# (a) Importing the updated summary_basic module in a new Python file.
import summary_basic
```

In [14]:
```
# (b) Using the functions from the module, compute and display:
#      * Mean
#      * Median
#      * Mode
#      * Standard Deviation

mean_val = summary_basic.mean(cleaned_study_hours)
median_val = summary_basic.median(cleaned_study_hours)
mode_val = summary_basic.mode(cleaned_study_hours)
std_dev_val = summary_basic.std_dev(cleaned_study_hours)
```

In [15]:
```
# (c) Display the results clearly with appropriate labels.
print("Summary Statistics of Study Hours")
print("---------------------------------")
print("Mean               :", round(mean_val, 4))
print("Median             :", round(median_val, 4))
print("Mode               :", round(mode_val, 4))
print("Standard Deviation :", round(std_dev_val, 4))
```

```
Summary Statistics of Study Hours
---------------------------------
Mean               : 4.65
Median             : 4.5
Mode               : 4
Standard Deviation : 1.4965
```