

# **System Requirements**

## **Specification Index**

**For**

### **NumPy Array Operations for Sales Data**

**(Topic: Basic Array Operations )**

**Version 1.0**

# Sales Data Analysis Console

## System Requirements Specification

### Project Abstract

The Sales Data Analysis Console is a Python application developed to analyze and manipulate sales data for two products over a period of time. The application performs various statistical operations on sales data using NumPy arrays, including addition, subtraction, mean, and median calculations. The system is designed to provide insights into sales trends, helping businesses better understand product performance and optimize their sales strategies.

### Business Requirements

- **Sales Data Analysis:** The application allows businesses to analyze daily sales data for two products and provides key performance indicators such as total sales, average sales, and sales distribution.
- **Data Accuracy:** The application ensures that sales data is correctly processed and validated, raising errors if the data is empty or incorrectly formatted.
- **Efficient Computation:** The system uses NumPy arrays for fast computation of sales metrics, ensuring the system is optimized for large datasets.

### Constraints

#### Input Requirements

- **Product 1 Sales Data:** Must be a list of integers representing daily sales for product 1.
- **Product 2 Sales Data:** Must be a list of integers representing daily sales for product 2.
- **Sales Data Validation:** Both sales data arrays must not be empty to ensure accurate computations.

#### Conversion Constraints

- **Addition of Sales:** The application must add sales data for both products element-wise and ensure the sizes of the arrays match.
- **Subtraction of Sales:** The application must subtract the sales data of the second product from the first product's sales, element-wise.
- **Statistical Calculations:** The application computes mean and median for the combined sales data of both products.

### Output Constraints

1. **Display Format:**

- Show the Results for Each Calculation:
    - "Sum of Sales: {value}"
    - "Difference of Sales: {value}"
    - "Mean Sales: {value}"
    - "Median Sales: {value}"
  - 2. Required Output:
    - Display the results of all calculations with proper labeling, including:
      - Sum of the two products' daily sales.
      - Difference between the two products' daily sales.
      - Mean sales across both products.
      - Median sales across both products.
- 

### Template Code Structure:


1. Class Definition:
  - **SalesAnalysis**: Class to store and process sales data for two products.
2. Functions:
  - **add\_sales()**: Adds the sales of the two products element-wise.
  - **subtract\_sales()**: Subtracts the sales data of product 2 from product 1 element-wise.
  - **calculate\_mean()**: Computes the mean of the total sales (sum of both products).
  - **calculate\_median()**: Computes the median of the total sales.
3. Input Section:
  - Initialize product sales data for both products (as lists of integers).
  - Validate that sales data arrays are not empty.
4. Processing Section:
  - Perform addition, subtraction, and statistical calculations on the sales data.
5. Output Section:
  - Display results for the sum, difference, mean, and median calculations.

### Execution Steps to Follow:

- All actions like build, compile, running application, running test cases will be through Command Terminal.
- To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
- This editor Auto Saves the code
- If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B** -command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
- These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
- To setup environment:  
You can run the application without importing any packages
- To launch application:  
**python3 mainclass.py**
- To run Test cases:  
**python3 -m unittest**
- Before Final Submission also, you need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

### Screen shot to run the program


To run the application



```
OK
coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 <<scriptname>>.py
```

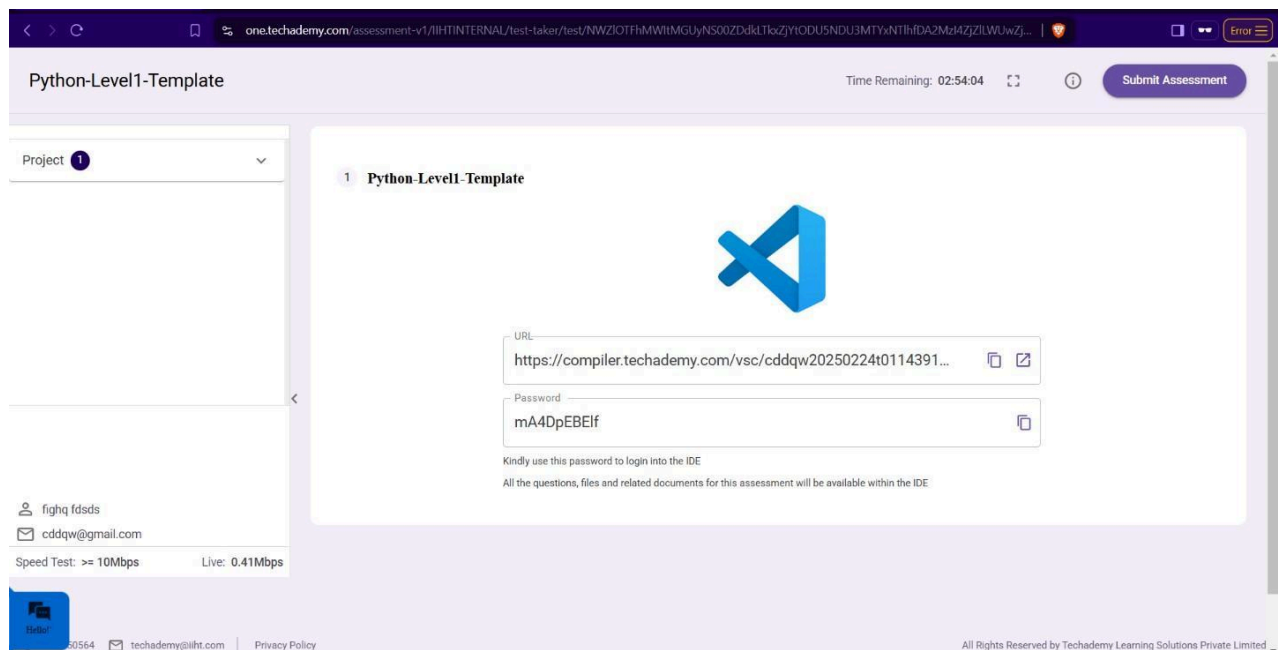
**python3 mainclass.py**

```
● coder@dighe20250227t070305rz1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 -m unittest
TestBoundary = Passed
.TestExceptional = Passed
.TestCalculateTotalDonations = Failed
.TestCalculateTotalStockValue = Failed
.TestCheckFrankWhiteDonated = Failed
```



To run the testcase

`python3 -m unittest`



- Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.