

---

# System Requirements Specification Index

For

Python Basics and NumPy, Pandas

Usecase No 6

1.0

IIHT Pvt. Ltd.  
fullstack@iiht.com

## Use Case: Recipe Management System (recipe.py)

### Dataset

```
recipes = {
    "Pasta": {"Flour": 200, "Eggs": 2, "Cheese": 50, "Milk": 100},
    "Pizza": {"Flour": 250, "Tomato": 100, "Cheese": 150, "Olives": 50},
    "Salad": {"Lettuce": 100, "Tomato": 150, "Cucumber": 100, "Cheese": 50},
    "Soup": {"Carrot": 200, "Potato": 150, "Onion": 100, "Garlic": 20},
    "Cake": {"Flour": 300, "Sugar": 200, "Butter": 150, "Eggs": 3}
}

ingredient_prices = {
    "Flour": 0.5, "Eggs": 5, "Cheese": 2, "Milk": 1,
    "Tomato": 1.5, "Olives": 3, "Lettuce": 1.2, "Cucumber": 2,
    "Carrot": 0.8, "Potato": 0.6, "Onion": 0.9, "Garlic": 4,
    "Sugar": 1.3, "Butter": 2.5
}
```

**1) Write a Python function to calculate the total cost of each recipe.**

**Define calculate\_total\_cost()**

The function should:

- Iterate through the recipe dictionary.
- Calculate the total cost based on ingredient quantities and their respective prices.
- Return a dictionary with the recipe name as the key and total cost as the value.

**2 ) Write a Python function to normalize ingredient quantities.**

- **Define normalize\_quantities()**

The function should:

- Convert the recipe data into a NumPy array.
- Scale ingredient quantities by a factor of 1.5.
- Return a Pandas DataFrame with updated ingredient values.

**3 ) Write a Python function to identify unique ingredients used in all recipes.**

**Define unique\_ingredients()**

The function should:

Extract unique ingredient names from all recipes.

Return a **set** of unique ingredient names.

## Use Case: Sports Score Prediction System (SportsScorePredictionSystem.py)

```
match_data = [
    {"team": "Team A", "prev_score": 65, "opp_strength": 70},
    {"team": "Team B", "prev_score": 80, "opp_strength": 60},
    {"team": "Team C", "prev_score": 55, "opp_strength": 75},
    {"team": "Team D", "prev_score": 90, "opp_strength": 50},
    {"team": "Team E", "prev_score": 70, "opp_strength": 65}
]
```

**1 ) Write a Python function to predict the match score.**

Define predict\_score(prev\_score, opp\_strength)

The function should:

- Take the previous score and opponent strength as inputs.
- Predict the score using the formula:  $\text{prev\_score} * 0.8 + \text{opp\_strength} * 0.2$ .
- Return the predicted score as an integer.

## 2 )Write a Python function to save match predictions.

Define `save_predictions(match_data, filename="predictions.csv")`

The function should:

- Iterate through the match data list.
- Calculate and add the predicted score for each match.
- Save the predictions to a CSV file.
- Return the list of predictions.

## 3 )Write a Python function to analyze predictions and find the highest predicted scorer.

Define `analyze_predictions(filename="predictions.csv")`

The function should:

- Read the predictions from the CSV file.
- Identify the team with the highest predicted score.
- Print and return the team name along with its predicted score.

## Use Case: Loan Management System (LoanManagementSystem.py)

### 1)Write a Python function to calculate the total repayment amount.

Define `calculate_total_amount(principal, rate, tenure)`

```
loan_data = np.array([
    [101, 5000, 5, 2],
    [102, 10000, 4, 5],
    [103, 7500, 6, 3],
    [104, 12000, 3, 4],
    [105, 3000, 7, 1]
])
```

The function should:

- Take principal amount, interest rate (in %), and tenure (in years) as inputs.
- Calculate the total repayment amount using the formula:  
$$\text{Total Amount} = \text{Principal} + (\text{Principal} \times \text{Rate} \times \text{Tenure} / 100)$$
  
$$\text{Total Amount} = \text{Principal} + (\text{Principal} \times \frac{\text{Rate}}{100} \times \text{Tenure})$$
- Return the total repayment amount.

### 2 )Write a Python function to display loan details.

Define `display_loans(loan_data)`

The function should:

- Iterate through the loan dataset.
- Extract loan details including loan ID, principal, interest rate, and tenure.
- Calculate the total repayment amount for each loan.
- Print and return a list of formatted loan details.

### 3 )Write a Python function to find the loan with the highest total repayment amount.

Define `find_highest_repayment(loan_data)`

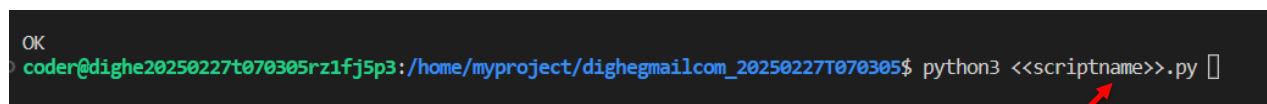
The function should:

- Iterate through the loan dataset.
- Identify the loan with the highest total repayment amount.
- Return the loan details with the highest repayment.

### **Execution Steps to Follow:**

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
3. This editor Auto Saves the code
4. 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.
5. 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.
6. To setup environment:  
You can run the application without importing any packages
7. To launch application:  
**python3 recipe.py**  
**python3 SportsScorePredictionSystem.py**  
**python3 LoanManagementSystem.py**
8. To run Test cases:  
**python3 -m unittest**
9. 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.

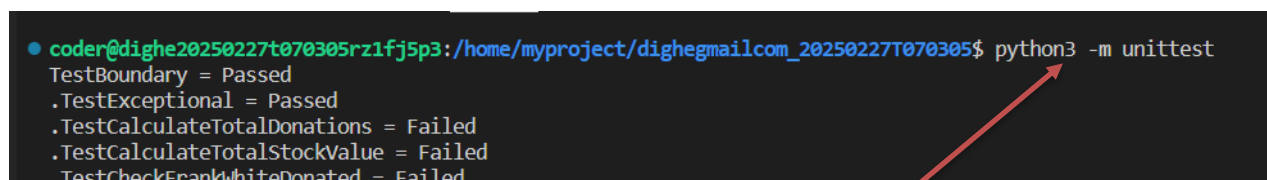
### **Screen shot to run the program**



```
OK
coder@dighe20250227t070305r1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 <<scriptname>>.py
```

To run the application

```
python3 recipe.py
python3 SportsScorePredictionSystem.py
python3 LoanManagementSystem.py
```



```
● coder@dighe20250227t070305r1fj5p3:/home/myproject/dighegmailcom_20250227T070305$ python3 -m unittest
TestBoundary = Passed
.TestExceptional = Passed
.TestCalculateTotalDonations = Failed
.TestCalculateTotalStockValue = Failed
.TestCheckFrankWhiteDonated = Failed
```

To run the test case

```
python3 -m unittest
```


Python-Level1-Template

Time Remaining: 02:54:04

Submit Assessment

Project 1

1 Python-Level1-Template



URL  
https://compiler.techademy.com/vsc/cddqw20250224t0114391...

Password  
mA4DpEBElF

Kindly use this password to login into the IDE  
All the questions, files and related documents for this assessment will be available within the IDE

Speed Test: >= 10Mbps Live: 0.41Mbps

50564 techademy@iitit.com Privacy Policy

All Rights Reserved by Techademy Learning Solutions Private Limited

**Click on the submit assessment button after you have used the command CTRL+Shift+B-command in the terminal.**