

---

# System Requirements Specification Index

For

Python Basics and NumPy, Pandas  
Use case No 8  
1.0

IIHT Pvt. Ltd.

[fullstack@iiht.com](mailto:fullstack@iiht.com)

## Usecase No 1 Use Case: Student Attendance Management System (student.py)

Dataset

students = {

```
101: {'name': 'Alice Johnson', 'class': '10th Grade'},
102: {'name': 'Bob Smith', 'class': '10th Grade'},
103: {'name': 'Charlie Brown', 'class': '10th Grade'},
104: {'name': 'David Lee', 'class': '10th Grade'},
105: {'name': 'Eve Miller', 'class': '10th Grade'}
```

}

Attendance Records (student\_id: list of (date, status) tuples):

attendance = {

```
101: [('2025-02-25', 'Absent'), ('2025-02-26', 'Absent'), ('2025-02-27', 'Absent')],
102: [('2025-02-25', 'Absent'), ('2025-02-26', 'Present'), ('2025-02-27', 'Present')],
103: [('2025-02-25', 'Present'), ('2025-02-26', 'Absent'), ('2025-02-27', 'Present')],
104: [('2025-02-25', 'Present'), ('2025-02-26', 'Present'), ('2025-02-27', 'Present')],
105: [('2025-02-25', 'Absent'), ('2025-02-26', 'Present'), ('2025-02-27', 'Absent')]
```

}

1 Write a Python function to count the number of unique attendance days.

Define: count\_number\_of\_days\_using\_dates()

The function should:

- Identify the unique dates in the attendance dataset.
- **return** the total number of unique dates.

2 Write a Python function to find the student with the most absent days.

Define: find\_most\_absent\_student()

The function should:

- Count the number of absences for each student.
- Identify and return the following details of the student with the highest number of absent days in form of tuple:
  - Student name
  - No of days he/she was absent

## Use Case No 2: Flight Management System (FlightReservationSystem.py)

Dataset

flights\_data = {

```
"Flight Number": ["AI101", "BA202", "DL303"],
"Airline": ["Air India", "British Airways", "Delta Airlines"],
"Total Seats": [150, 180, 200],
"Booked Seats": [120, 160, 190],
"Ticket Price": [8000, 12000, 10000] # Price per ticket in ₹
```

}

A **static dataset** is defined using a **Python dictionary** named flights\_data.

It contains information about 3 flights, including:

"Flight Number"

"Airline"

"Total Seats"

"Booked Seats"

"Ticket Price" (in ₹)

This dictionary is converted into a **Pandas DataFrame (flights\_df)** for all subsequent operations.

1. Write a Python function to list all flights with details.

Define: list\_all\_flights()

The function should:

- Return all the flight data as pandas data frame .
- **Return type DataFrame**

2. Write a Python function to check available seats for a given flight.

Define: available\_seats\_for\_flight(flight\_number)

The function should:

- Accepts a **flight number** (e.g., "AI101").
- Looks up the flight in the DataFrame.
- If the flight exists:
- Retrieves Total Seats and Booked Seats.
- Uses **NumPy** to compute available\_seats = Total Seats - Booked Seats.
- Returns the available seat count as an **integer**.

.

3. Write a Python function to calculate the total revenue generated from all flights.

Define: total\_revenue\_for\_all\_flights()

The function should:

- Calculate the revenue for each flight using Booked Seats \* Ticket Price.
- Compute the total revenue using Pandas operations to sum.
- Return the total revenue value as numeric value.

### Use Case No 3: Movie Ticket Booking System (MovieTicketBookingSystem.py)

Dataset is given from movie.json

```
movies = {
    "Avengers: Endgame": {
        "total_seats": 10,
        "booked_seats": 7,
        "ticket_price": 250 # Price per ticket in ₹
    },
    "Inception": {
        "total_seats": 8,
        "booked_seats": 4,
        "ticket_price": 200
    },
    "The Dark Knight": {
        "total_seats": 12,
        "booked_seats": 11,
        "ticket_price": 300
    }
}
```

Write the python code for the following

1. Use the file handling method `load_movies()`

to read the dataset value from `movie.json` and return as Dictionary. If the file does not exist return empty Dictionary

2. Write a Python function to find the total tickets sold

Define : `total_tickets_sold()`

The function should:

- Receive dictionary of movie data (retrieved from `load_movies()` method)
- Sum up all the booked seats across different movies.
- Return the total count of sold tickets.

2. Write a Python function to find the movie that has generated the highest revenue.

Define: `highest_revenue_movie()`

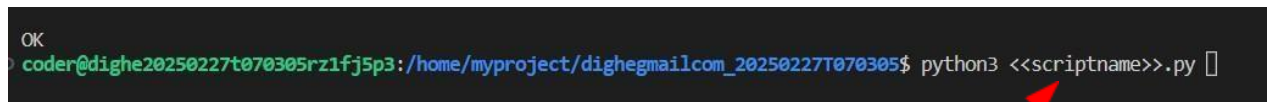
The function should:

- Receive dictionary of movie data (retrieved from `load_movies()` method)
- Compute the total revenue for each movie using `Booked Seats * Ticket Price`.
- Identify and return the following details of movie with the highest revenue as tuple :
  - Movie Name
  - Revenue

## 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) .
- 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 student.py**  
**python3 MovieTicketBookingSystem.py**  
**python3 FlightReservationSystem.py**
- To run Test cases:  
**python3 -m unittest**

## Screen shot to run the program

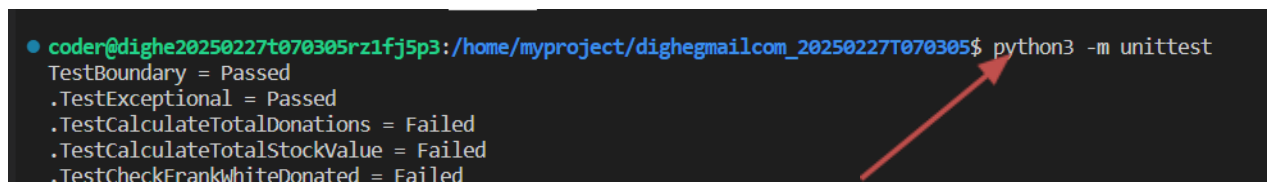


```
OK
coder@dighe20250227t070305rz1fj5p3: /home/myproject/dighegmailcom_20250227T070305$ python3 <<scriptname>>.py
```

A terminal window with a dark background. The prompt is 'coder@dighe20250227t070305rz1fj5p3: /home/myproject/dighegmailcom\_20250227T070305\$'. The command 'python3 <<scriptname>>.py' is entered. A red arrow points from the text 'To run the application' to the '<<scriptname>>' part of the command.

To run the application

```
python3 student.py
python3 MovieTicketBookingSystem.py
python3 FlightReservationSystem.py
```

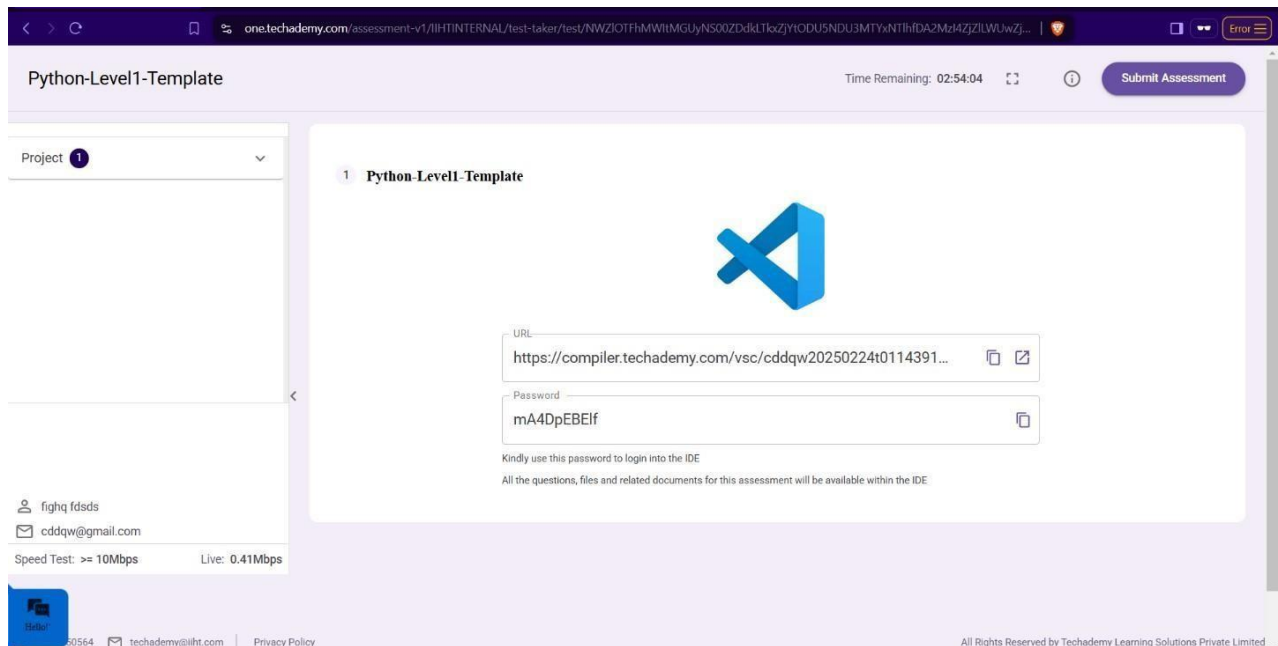


```
• coder@dighe20250227t070305rz1fj5p3: /home/myproject/dighegmailcom_20250227T070305$ python3 -m unittest
TestBoundary = Passed
.TestExceptional = Passed
.TestCalculateTotalDonations = Failed
.TestCalculateTotalStockValue = Failed
.TestCheckFrankWhiteDonated = Failed
```

A terminal window with a dark background. The prompt is '• coder@dighe20250227t070305rz1fj5p3: /home/myproject/dighegmailcom\_20250227T070305\$'. The command 'python3 -m unittest' is entered. The output is 'TestBoundary = Passed', '.TestExceptional = Passed', '.TestCalculateTotalDonations = Failed', '.TestCalculateTotalStockValue = Failed', and '.TestCheckFrankWhiteDonated = Failed'. A red arrow points from the text 'To run the testcase' to the 'python3 -m unittest' command.

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.**