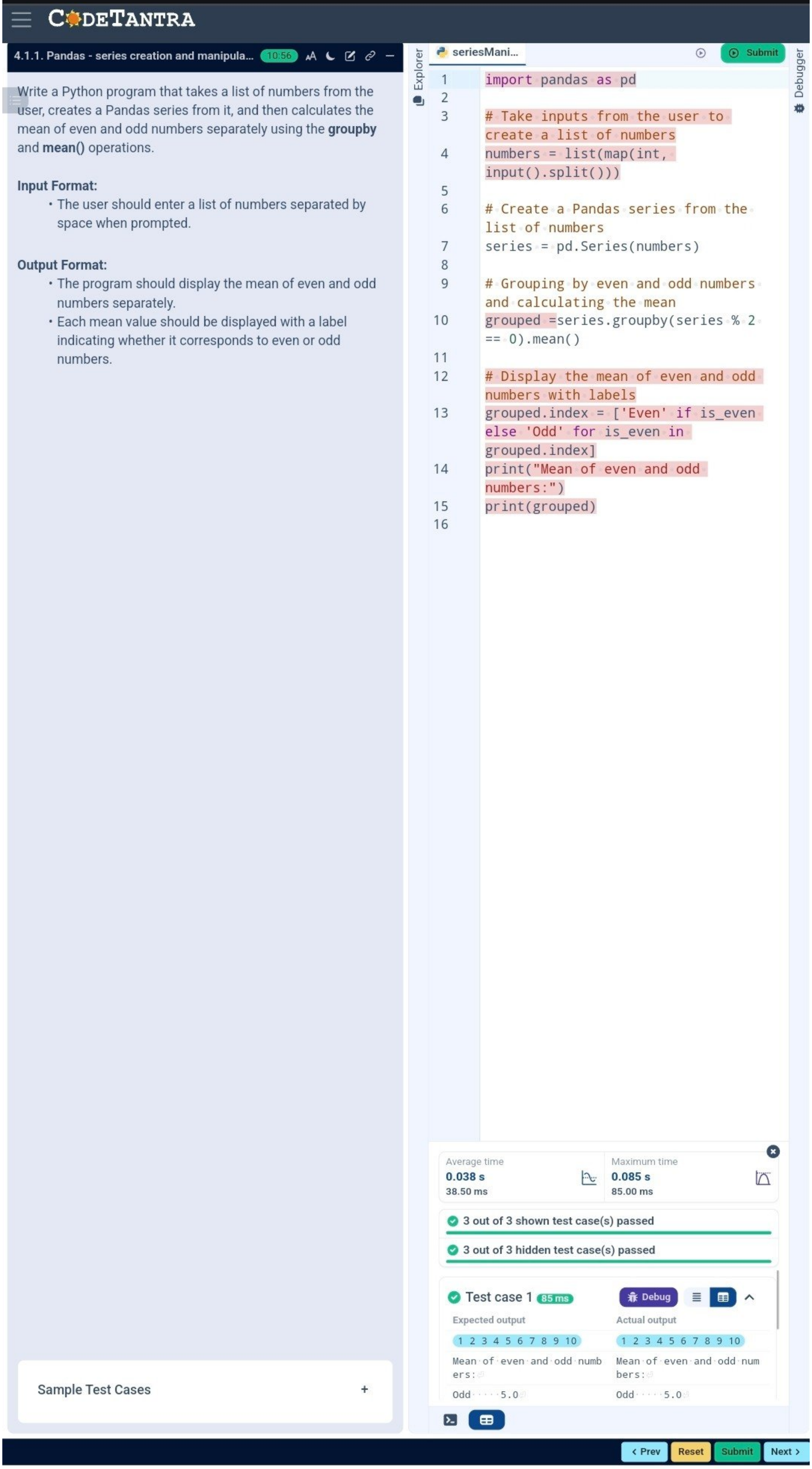


Write a Python program that takes a list of numbers from the user, creates a Pandas series from it, and then calculates the mean of even and odd numbers separately using the **groupby** and **mean()** operations.

Input Format:

- The user should enter a list of numbers separated by space when prompted.

Output Format:

- The program should display the mean of even and odd numbers separately.
- Each mean value should be displayed with a label indicating whether it corresponds to even or odd numbers.

```python
import pandas as pd

# Take inputs from the user to create a list of numbers
numbers = list(map(int, input().split()))

# Create a Pandas series from the list of numbers
series = pd.Series(numbers)

# Grouping by even and odd numbers and calculating the mean
grouped =series.groupby(series % 2 == 0).mean()

# Display the mean of even and odd numbers with labels
grouped.index = ['Even' if is_even else 'Odd' for is_even in grouped.index]
print("Mean of even and odd numbers:")
print(grouped)

```

Average time
0.038 s
38.50 ms

Maximum time
0.085 s
85.00 ms

3 out of 3 shown test case(s) passed

3 out of 3 hidden test case(s) passed

Test case 1 85 ms

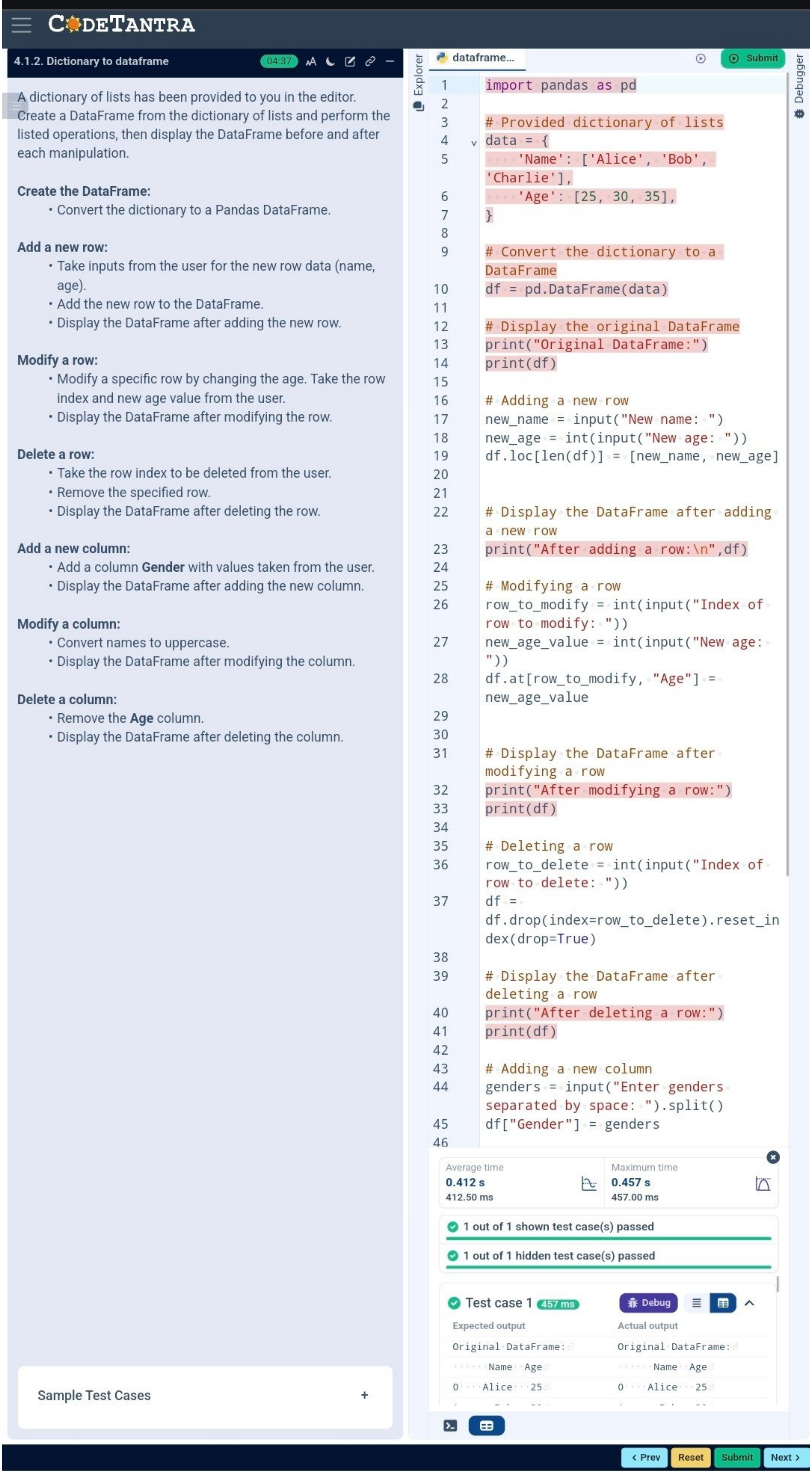
Expected output

```
1 2 3 4 5 6 7 8 9 10
Mean of even and odd numbers:
Odd    5.0
```

Actual output

```
1 2 3 4 5 6 7 8 9 10
Mean of even and odd numbers:
Odd    5.0
```

Sample Test Cases +



4.1.2. Dictionary to dataframe

A dictionary of lists has been provided to you in the editor. Create a DataFrame from the dictionary of lists and perform the listed operations, then display the DataFrame before and after each manipulation.

Create the DataFrame:

- Convert the dictionary to a Pandas DataFrame.

Add a new row:

- Take inputs from the user for the new row data (name, age).
- Add the new row to the DataFrame.
- Display the DataFrame after adding the new row.

Modify a row:

- Modify a specific row by changing the age. Take the row index and new age value from the user.
- Display the DataFrame after modifying the row.

Delete a row:

- Take the row index to be deleted from the user.
- Remove the specified row.
- Display the DataFrame after deleting the row.

Add a new column:

- Add a column **Gender** with values taken from the user.
- Display the DataFrame after adding the new column.

Modify a column:

- Convert names to uppercase.
- Display the DataFrame after modifying the column.

Delete a column:

- Remove the **Age** column.
- Display the DataFrame after deleting the column.

Sample Test Cases +

```python
import pandas as pd

# Provided dictionary of lists
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
}

# Convert the dictionary to a DataFrame
df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Adding a new row
new_name = input("New name: ")
new_age = int(input("New age: "))
df.loc[len(df)] = [new_name, new_age]


# Display the DataFrame after adding a new row
print("After adding a row:\n",df)

# Modifying a row
row_to_modify = int(input("Index of row to modify: "))
new_age_value = int(input("New age: "))
df.at[row_to_modify, "Age"] = new_age_value


# Display the DataFrame after modifying a row
print("After modifying a row:")
print(df)

# Deleting a row
row_to_delete = int(input("Index of row to delete: "))
df = df.drop(index=row_to_delete).reset_index(drop=True)

# Display the DataFrame after deleting a row
print("After deleting a row:")
print(df)

# Adding a new column
genders = input("Enter genders separated by space: ").split()
df["Gender"] = genders

```

Average time 0.412 s 412.50 ms

Maximum time 0.457 s 457.00 ms

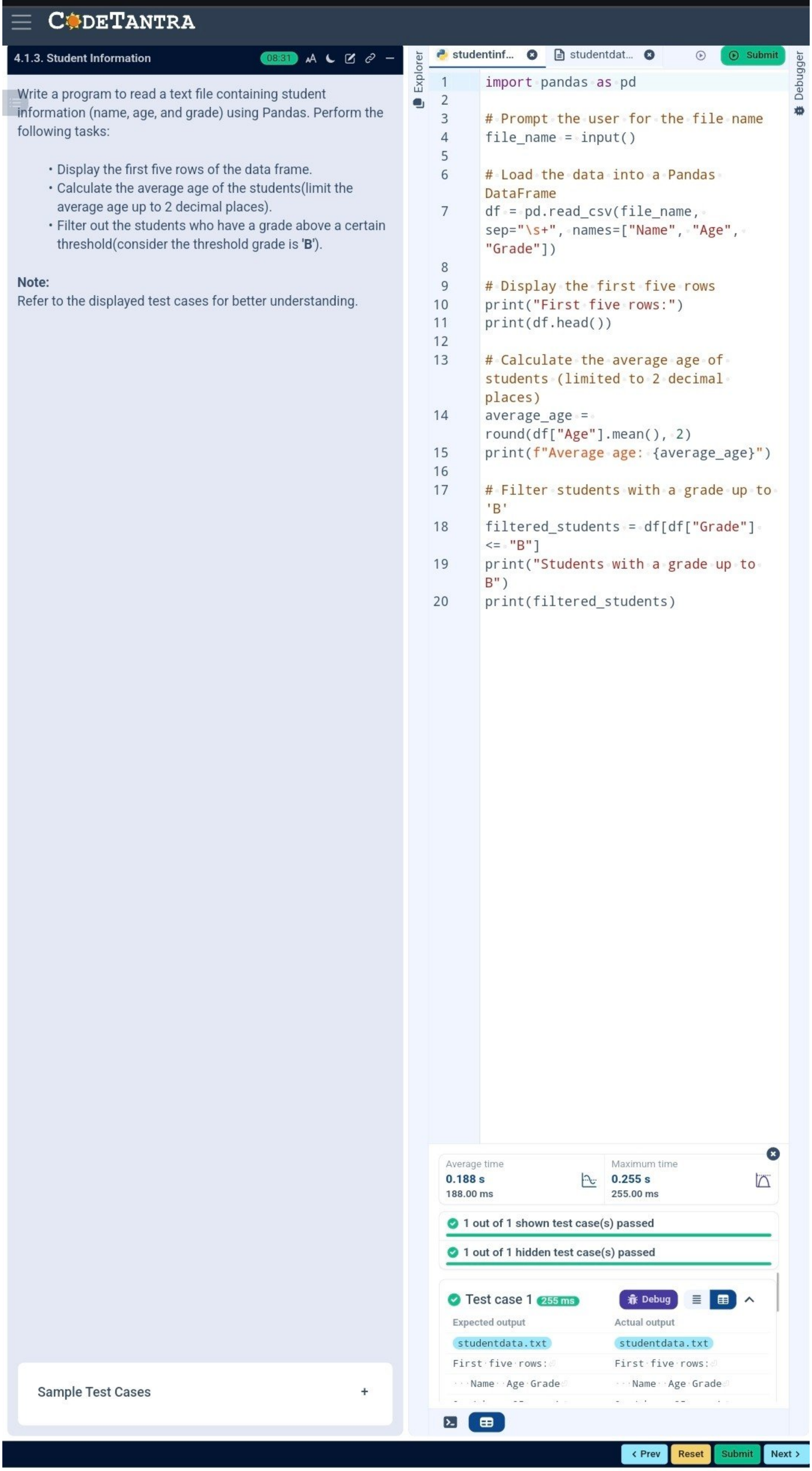
1 out of 1 shown test case(s) passed

1 out of 1 hidden test case(s) passed

Test case 1 457 ms Debug

Expected output	Actual output
Original DataFrame:	Original DataFrame:
Name Age	Name Age
0 Alice 25	0 Alice 25

< Prev Reset Submit Next >



4.1.3. Student Information

Write a program to read a text file containing student information (name, age, and grade) using Pandas. Perform the following tasks:

- Display the first five rows of the data frame.
- Calculate the average age of the students(limit the average age up to 2 decimal places).
- Filter out the students who have a grade above a certain threshold(consider the threshold grade is '**B**').

Note:
Refer to the displayed test cases for better understanding.

Sample Test Cases +

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data into a Pandas DataFrame
df = pd.read_csv(file_name, sep="\s+", names=["Name", "Age", "Grade"])

# Display the first five rows
print("First five rows:")
print(df.head())

# Calculate the average age of students (limited to 2 decimal places)
average_age = round(df["Age"].mean(), 2)
print(f"Average age: {average_age}")

# Filter students with a grade up to 'B'
filtered_students = df[df["Grade"] <= "B"]
print("Students with a grade up to B")
print(filtered_students)
```

Average time
0.188 s
188.00 ms

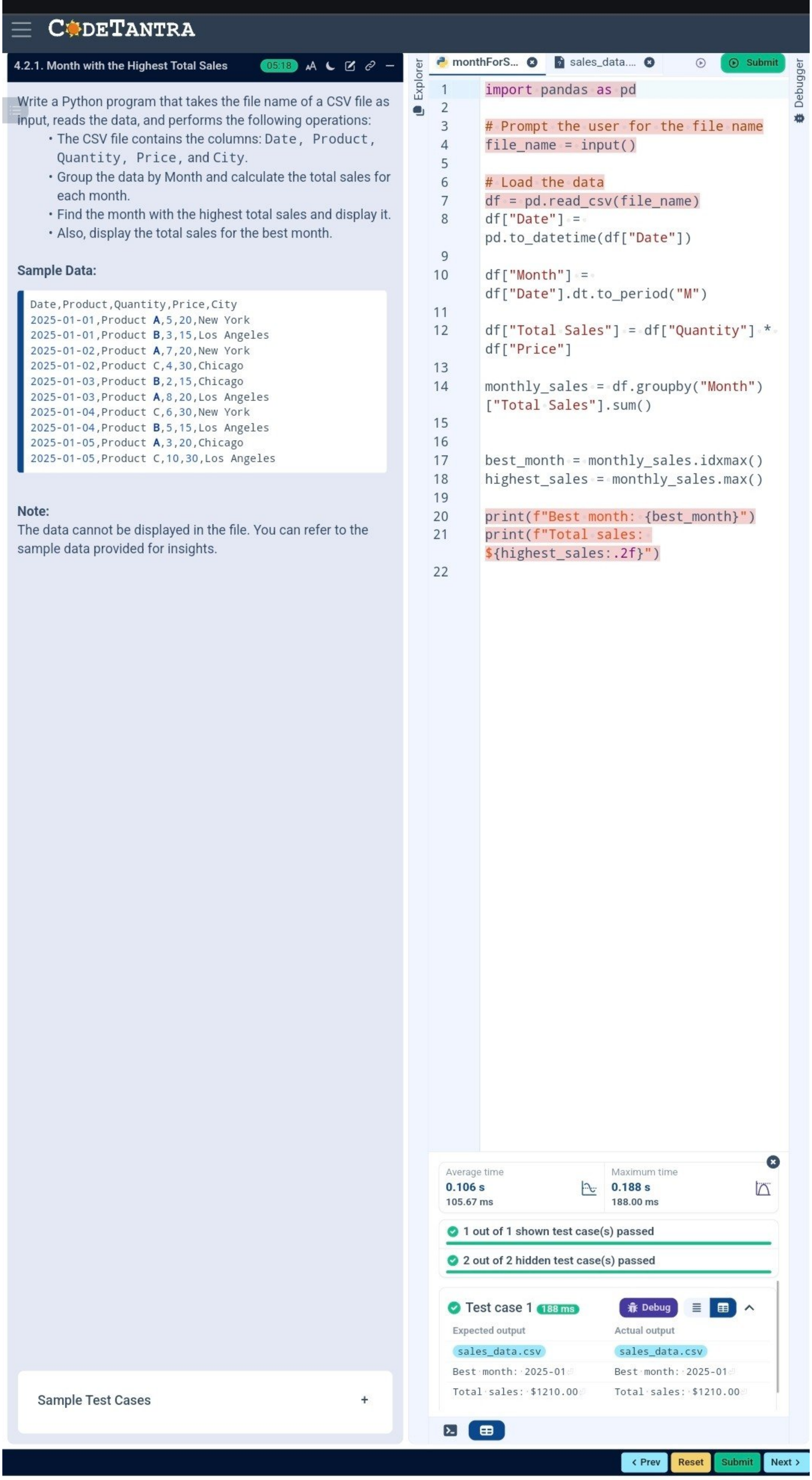
Maximum time
0.255 s
255.00 ms

1 out of 1 shown test case(s) passed

1 out of 1 hidden test case(s) passed

Test case 1 255 ms

Expected output	Actual output
studentdata.txt	studentdata.txt
First five rows:	First five rows:
Name Age Grade	Name Age Grade



4.2.1. Month with the Highest Total Sales

Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the columns: `Date, Product, Quantity, Price,` and `City`.
- Group the data by Month and calculate the total sales for each month.
- Find the month with the highest total sales and display it.
- Also, display the total sales for the best month.

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:
The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases +

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)
df["Date"] = pd.to_datetime(df["Date"])

df["Month"] = df["Date"].dt.to_period("M")

df["Total Sales"] = df["Quantity"] * df["Price"]

monthly_sales = df.groupby("Month")["Total Sales"].sum()


best_month = monthly_sales.idxmax()
highest_sales = monthly_sales.max()

print(f"Best month: {best_month}")
print(f"Total sales: ${highest_sales:.2f}")

```

Average time
0.106 s
105.67 ms

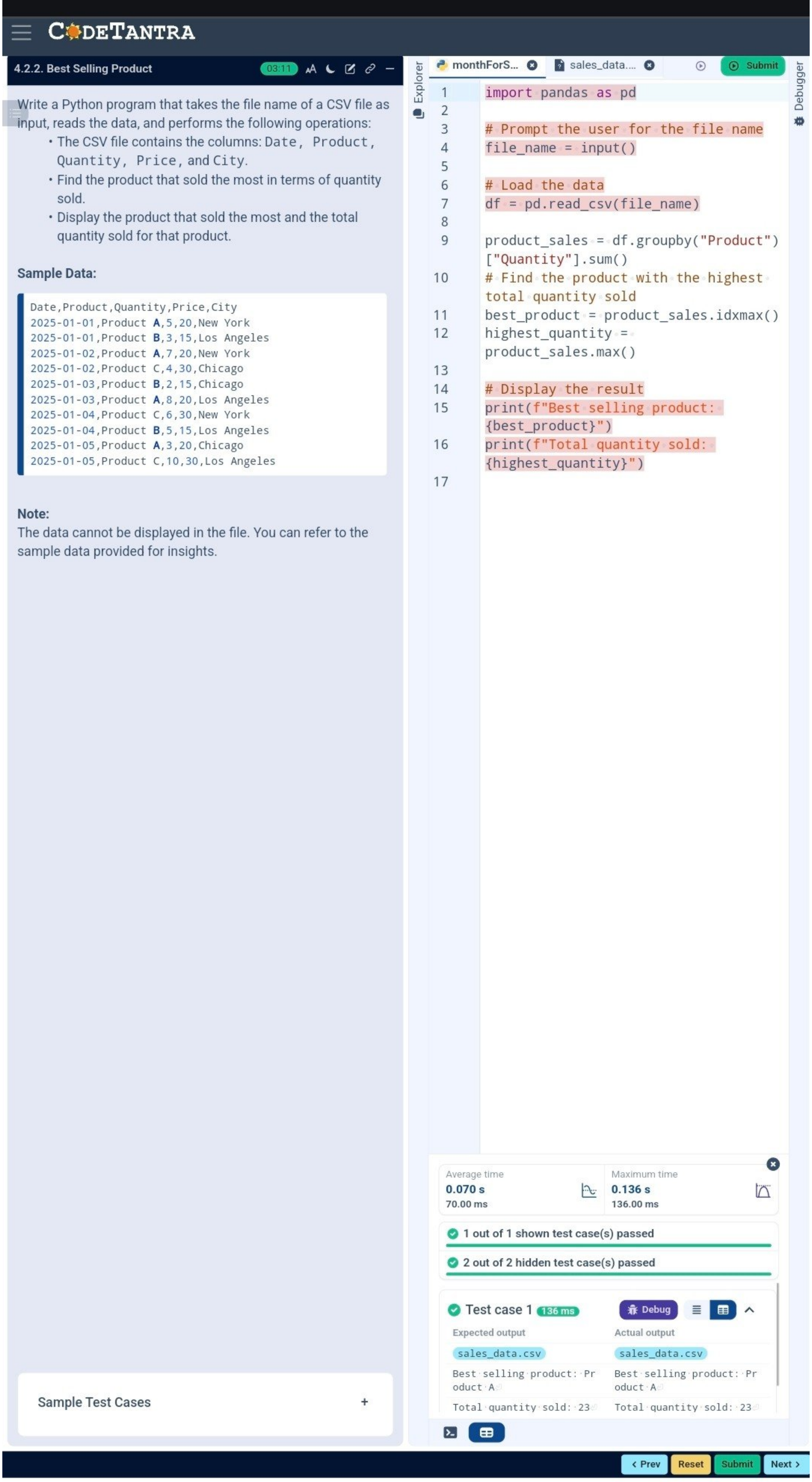
Maximum time
0.188 s
188.00 ms

1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 188 ms

Expected output	Actual output
sales_data.csv	sales_data.csv
Best month: 2025-01	Best month: 2025-01
Total sales: $1210.00	Total sales: $1210.00



Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the columns: `Date, Product, Quantity, Price,` and `City`.
- Find the product that sold the most in terms of quantity sold.
- Display the product that sold the most and the total quantity sold for that product.

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:
The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases +

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)

product_sales = df.groupby("Product")["Quantity"].sum()
# Find the product with the highest total quantity sold
best_product = product_sales.idxmax()
highest_quantity = product_sales.max()

# Display the result
print(f"Best selling product: {best_product}")
print(f"Total quantity sold: {highest_quantity}")

```

Average time
0.070 s
70.00 ms

Maximum time
0.136 s
136.00 ms

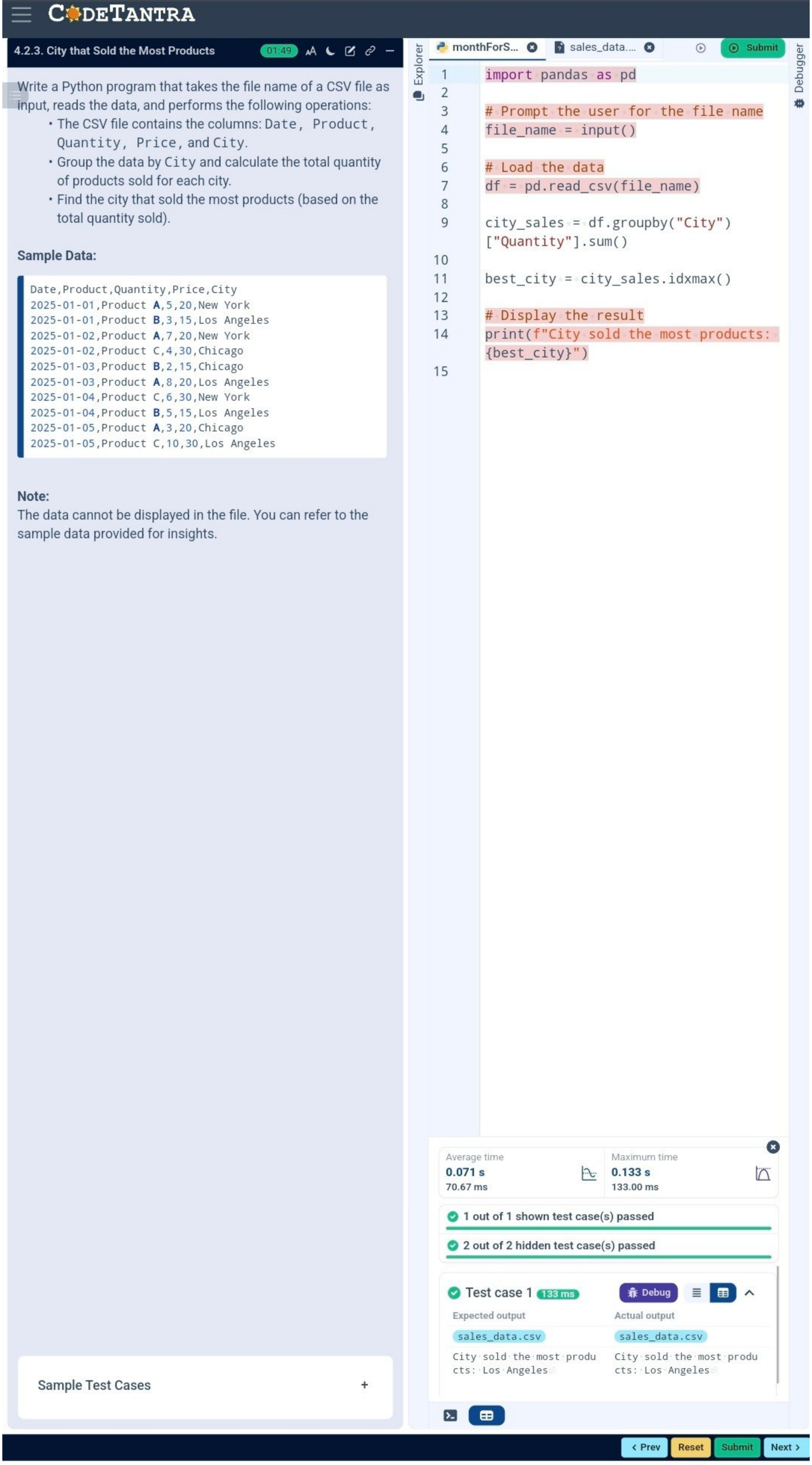
1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 136 ms Debug

Expected output	Actual output
sales_data.csv	sales_data.csv
Best selling product: Product A	Best selling product: Product A
Total quantity sold: 23	Total quantity sold: 23

< Prev Reset Submit Next >



4.2.3. City that Sold the Most Products

Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the columns: `Date, Product, Quantity, Price,` and `City`.
- Group the data by `City` and calculate the total quantity of products sold for each city.
- Find the city that sold the most products (based on the total quantity sold).

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:
The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)

city_sales = df.groupby("City")["Quantity"].sum()

best_city = city_sales.idxmax()

# Display the result
print(f"City sold the most products: {best_city}")
```

Average time 0.071 s 70.67 ms
Maximum time 0.133 s 133.00 ms

1 out of 1 shown test case(s) passed
2 out of 2 hidden test case(s) passed

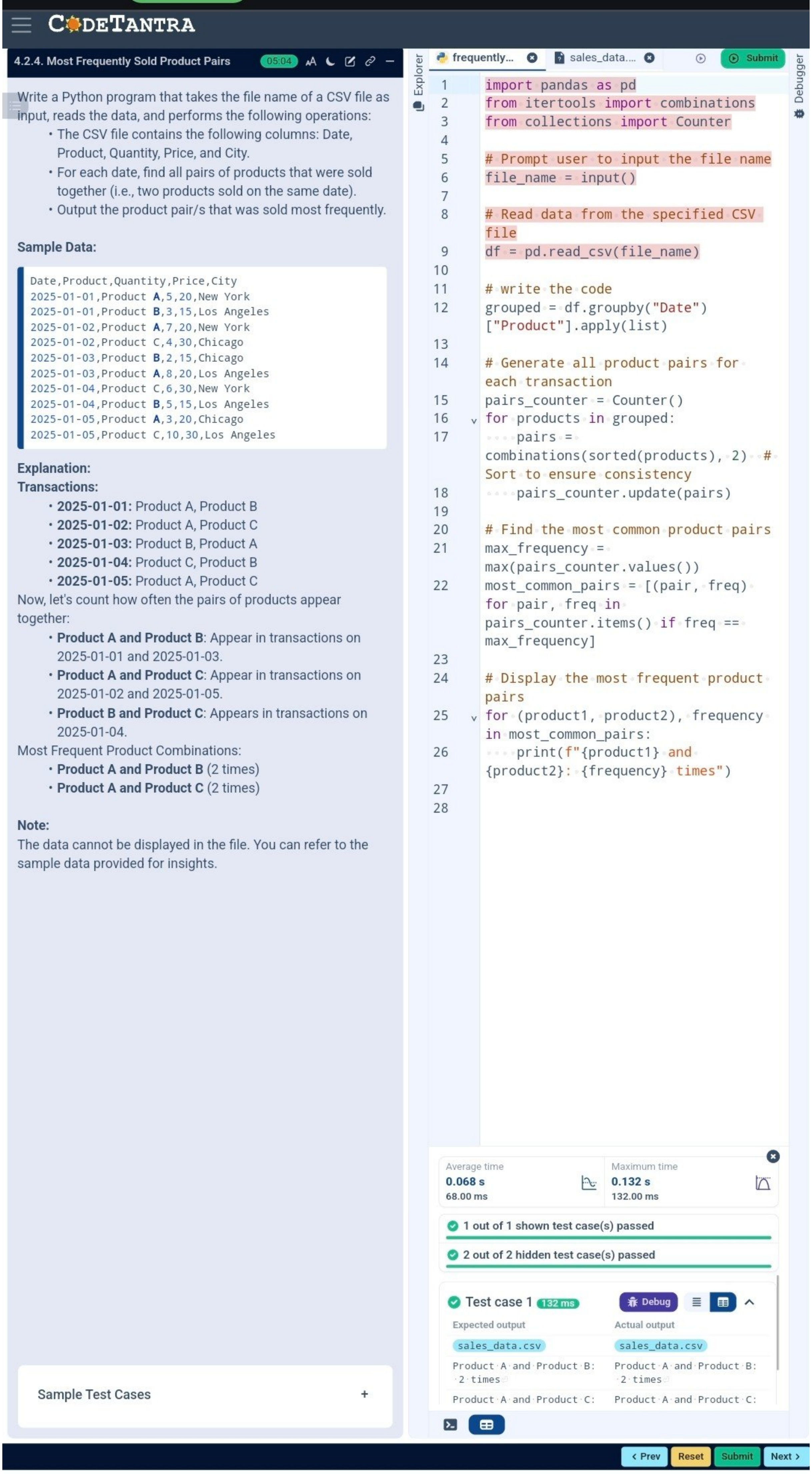
Test case 1 133 ms

Expected output

```
sales_data.csv
City sold the most products: Los Angeles
```

Actual output

```
sales_data.csv
City sold the most products: Los Angeles
```



CODETANTRA

4.2.4. Most Frequently Sold Product Pairs 05:04

Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the following columns: Date, Product, Quantity, Price, and City.
- For each date, find all pairs of products that were sold together (i.e., two products sold on the same date).
- Output the product pair/s that was sold most frequently.

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Explanation:
Transactions:

- **2025-01-01:** Product A, Product B
- **2025-01-02:** Product A, Product C
- **2025-01-03:** Product B, Product A
- **2025-01-04:** Product C, Product B
- **2025-01-05:** Product A, Product C

Now, let's count how often the pairs of products appear together:

- **Product A and Product B**: Appear in transactions on 2025-01-01 and 2025-01-03.
- **Product A and Product C**: Appear in transactions on 2025-01-02 and 2025-01-05.
- **Product B and Product C**: Appears in transactions on 2025-01-04.

Most Frequent Product Combinations:

- **Product A and Product B** (2 times)
- **Product A and Product C** (2 times)

Note:
The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases +

```python
import pandas as pd
from itertools import combinations
from collections import Counter

# Prompt user to input the file name
file_name = input()

# Read data from the specified CSV file
df = pd.read_csv(file_name)

# write the code
grouped = df.groupby("Date")["Product"].apply(list)

# Generate all product pairs for each transaction
pairs_counter = Counter()
for products in grouped:
    pairs = combinations(sorted(products), 2)  # Sort to ensure consistency
    pairs_counter.update(pairs)

# Find the most common product pairs
max_frequency = max(pairs_counter.values())
most_common_pairs = [(pair, freq) for pair, freq in pairs_counter.items() if freq == max_frequency]

# Display the most frequent product pairs
for (product1, product2), frequency in most_common_pairs:
    print(f"{product1} and {product2}: {frequency} times")
```

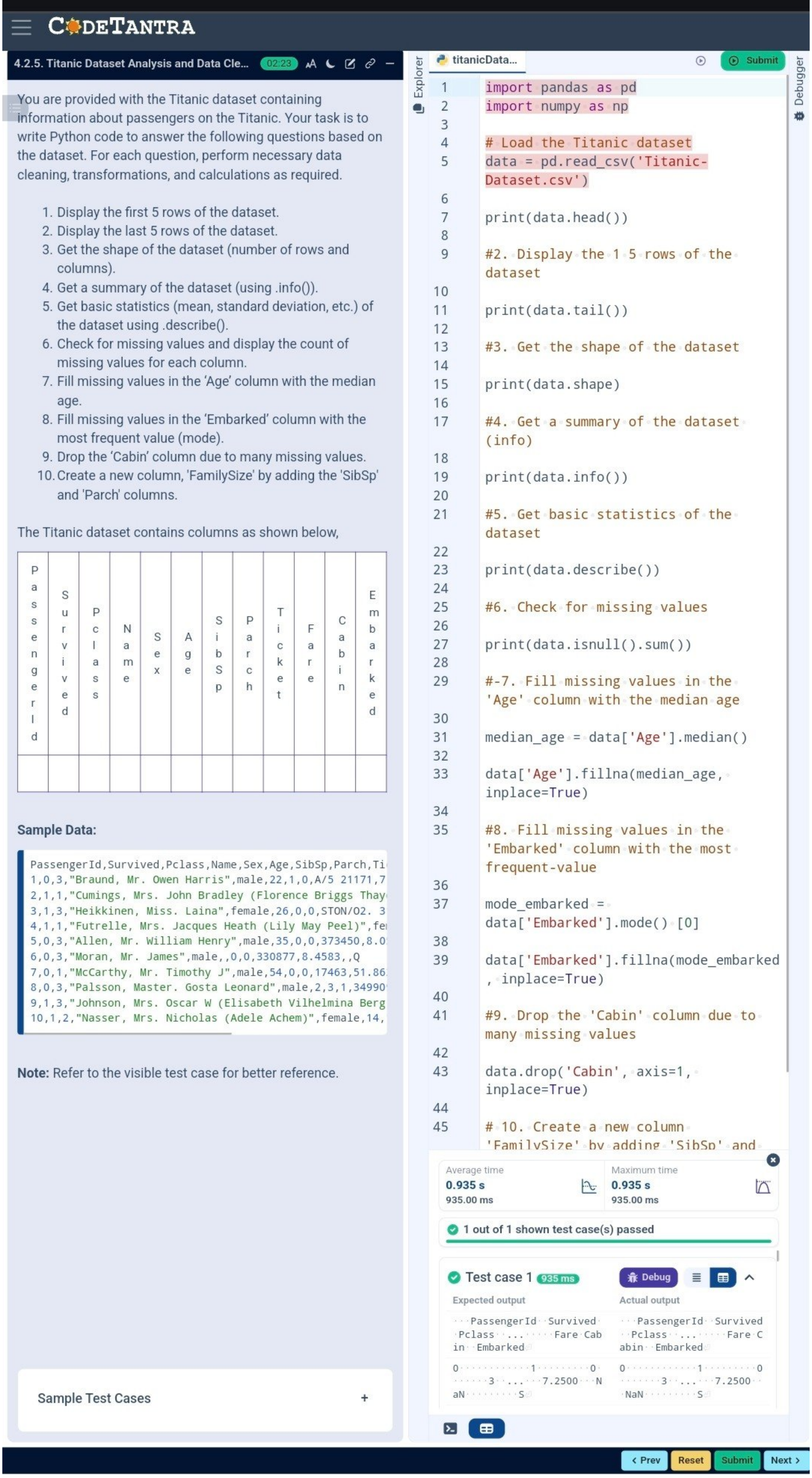
Average time 0.068 s 68.00 ms
Maximum time 0.132 s 132.00 ms

✓ 1 out of 1 shown test case(s) passed
✓ 2 out of 2 hidden test case(s) passed

✓ Test case 1 132 ms Debug

Expected output	Actual output
sales_data.csv	sales_data.csv
Product A and Product B: 2 times	Product A and Product B: 2 times
Product A and Product C:	Product A and Product C:

< Prev Reset Submit Next >



CODETANTRA

4.2.5. Titanic Dataset Analysis and Data Cle... 02:23

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset. For each question, perform necessary data cleaning, transformations, and calculations as required.

1. Display the first 5 rows of the dataset.
2. Display the last 5 rows of the dataset.
3. Get the shape of the dataset (number of rows and columns).
4. Get a summary of the dataset (using .info()).
5. Get basic statistics (mean, standard deviation, etc.) of the dataset using .describe().
6. Check for missing values and display the count of missing values for each column.
7. Fill missing values in the 'Age' column with the median age.
8. Fill missing values in the 'Embarked' column with the most frequent value (mode).
9. Drop the 'Cabin' column due to many missing values.
10. Create a new column, 'FamilySize' by adding the 'SibSp' and 'Parch' columns.

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

titanicData...   Submit

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')

print(data.head())

#2. Display the 1 5 rows of the dataset

print(data.tail())

#3. Get the shape of the dataset

print(data.shape)

#4. Get a summary of the dataset (info)

print(data.info())

#5. Get basic statistics of the dataset

print(data.describe())

#6. Check for missing values

print(data.isnull().sum())

#-7. Fill missing values in the 'Age' column with the median age

median_age = data['Age'].median()

data['Age'].fillna(median_age, inplace=True)

#8. Fill missing values in the 'Embarked' column with the most frequent-value

mode_embarked = data['Embarked'].mode() [0]

data['Embarked'].fillna(mode_embarked, inplace=True)

#9. Drop the 'Cabin' column due to many missing values

data.drop('Cabin', axis=1, inplace=True)

# 10. Create a new column 'FamilySize' by adding 'SibSp' and
```

Average time 0.935 s 935.00 ms | Maximum time 0.935 s 935.00 ms

1 out of 1 shown test case(s) passed

Test case 1 935 ms   Debug

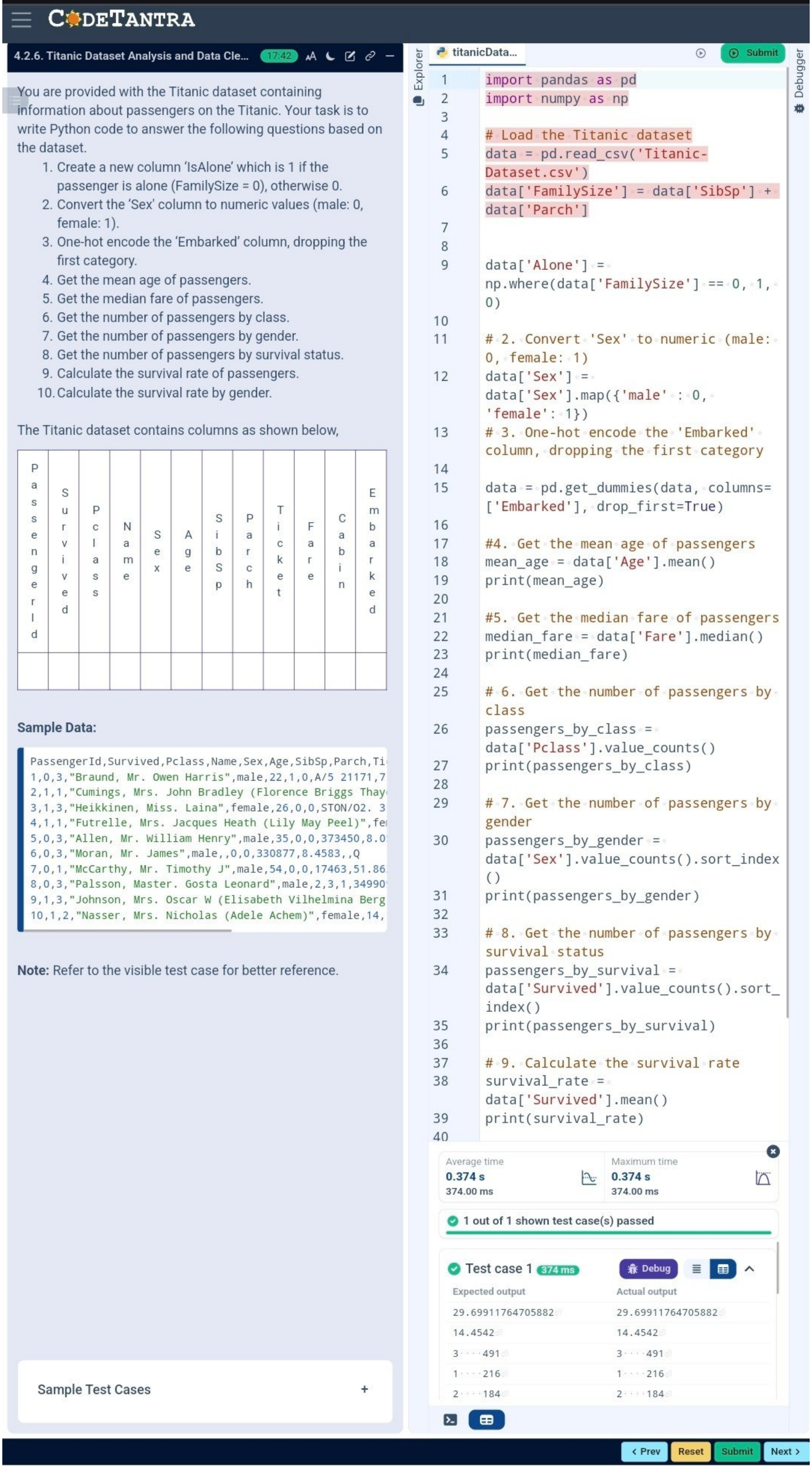
Expected output

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
```

Actual output

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
```

< Prev   Reset   Submit   Next >



CODETANTRA

4.2.6. Titanic Dataset Analysis and Data Cle... 17:42

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Create a new column 'IsAlone' which is 1 if the passenger is alone (FamilySize = 0), otherwise 0.
2. Convert the 'Sex' column to numeric values (male: 0, female: 1).
3. One-hot encode the 'Embarked' column, dropping the first category.
4. Get the mean age of passengers.
5. Get the median fare of passengers.
6. Get the number of passengers by class.
7. Get the number of passengers by gender.
8. Get the number of passengers by survival status.
9. Calculate the survival rate of passengers.
10. Calculate the survival rate by gender.

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

titanicData... Submit

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data['FamilySize'] = data['SibSp'] + data['Parch']


data['Alone'] = np.where(data['FamilySize'] == 0, 1, 0)

# 2. Convert 'Sex' to numeric (male: 0, female: 1)
data['Sex'] = data['Sex'].map({'male' : 0, 'female': 1})
# 3. One-hot encode the 'Embarked' column, dropping the first category

data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

#4. Get the mean age of passengers
mean_age = data['Age'].mean()
print(mean_age)

#5. Get the median fare of passengers
median_fare = data['Fare'].median()
print(median_fare)

# 6. Get the number of passengers by class
passengers_by_class = data['Pclass'].value_counts()
print(passengers_by_class)

# 7. Get the number of passengers by gender
passengers_by_gender = data['Sex'].value_counts().sort_index()
print(passengers_by_gender)

# 8. Get the number of passengers by survival status
passengers_by_survival = data['Survived'].value_counts().sort_index()
print(passengers_by_survival)

# 9. Calculate the survival rate
survival_rate = data['Survived'].mean()
print(survival_rate)

```

Average time
0.374 s
374.00 ms

Maximum time
0.374 s
374.00 ms

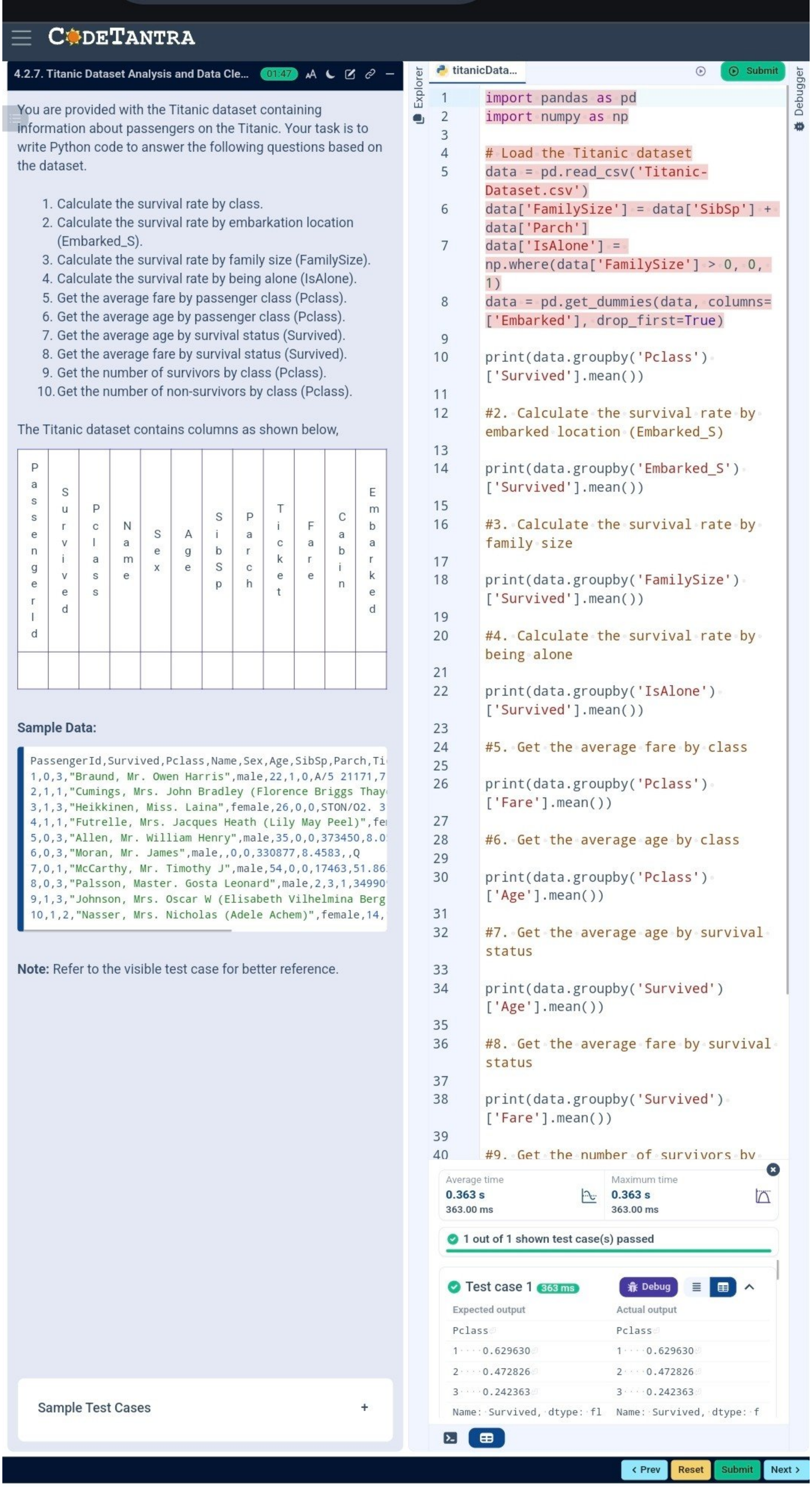
1 out of 1 shown test case(s) passed

Test case 1 374 ms Debug

Expected output
```
29.69911764705882
14.4542
3    491
1    216
2    184
```

Actual output
```
29.69911764705882
14.4542
3    491
1    216
2    184
```

‹ Prev Reset Submit Next ›



You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Calculate the survival rate by class.
2. Calculate the survival rate by embarkation location (Embarked_S).
3. Calculate the survival rate by family size (FamilySize).
4. Calculate the survival rate by being alone (IsAlone).
5. Get the average fare by passenger class (Pclass).
6. Get the average age by passenger class (Pclass).
7. Get the average age by survival status (Survived).
8. Get the average fare by survival status (Survived).
9. Get the number of survivors by class (Pclass).
10. Get the number of non-survivors by class (Pclass).

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data['FamilySize'] = data['SibSp'] + data['Parch']
data['IsAlone'] = np.where(data['FamilySize'] > 0, 0, 1)
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

print(data.groupby('Pclass')['Survived'].mean())

#2. Calculate the survival rate by embarked location (Embarked_S)

print(data.groupby('Embarked_S')['Survived'].mean())

#3. Calculate the survival rate by family size

print(data.groupby('FamilySize')['Survived'].mean())

#4. Calculate the survival rate by being alone

print(data.groupby('IsAlone')['Survived'].mean())

#5. Get the average fare by class

print(data.groupby('Pclass')['Fare'].mean())

#6. Get the average age by class

print(data.groupby('Pclass')['Age'].mean())

#7. Get the average age by survival status

print(data.groupby('Survived')['Age'].mean())

#8. Get the average fare by survival status

print(data.groupby('Survived')['Fare'].mean())

#9. Get the number of survivors by
```

Average time 0.363 s 363.00 ms | Maximum time 0.363 s 363.00 ms

1 out of 1 shown test case(s) passed

Test case 1 363 ms Debug

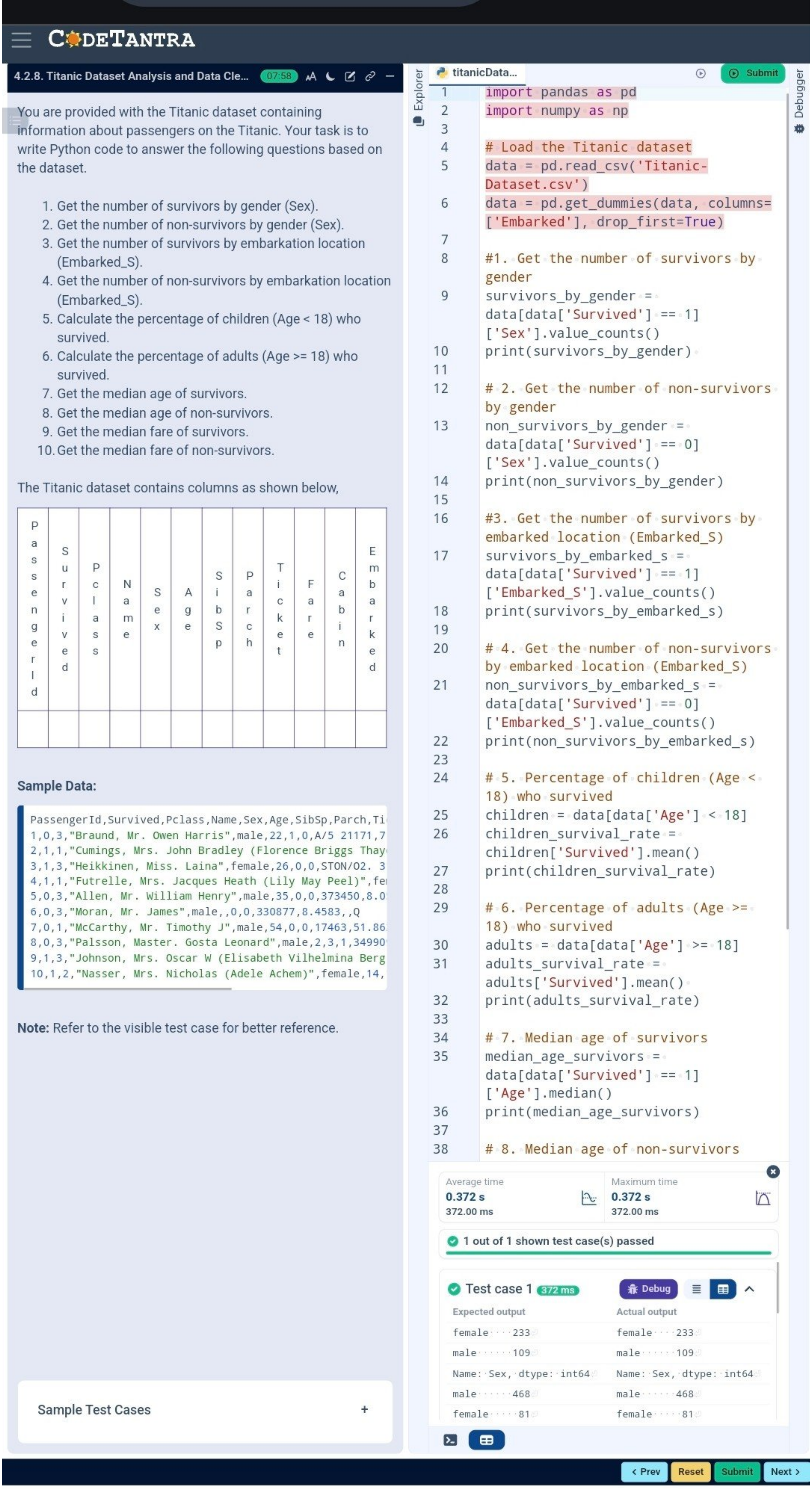
Expected output

```
Pclass
1    0.629630
2    0.472826
3    0.242363
Name: Survived, dtype: fl
```

Actual output

```
Pclass
1    0.629630
2    0.472826
3    0.242363
Name: Survived, dtype: f
```

< Prev Reset Submit Next >


CODETANTRA

4.2.8. Titanic Dataset Analysis and Data Cle... 07:58

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Get the number of survivors by gender (Sex).
2. Get the number of non-survivors by gender (Sex).
3. Get the number of survivors by embarkation location (Embarked_S).
4. Get the number of non-survivors by embarkation location (Embarked_S).
5. Calculate the percentage of children (Age < 18) who survived.
6. Calculate the percentage of adults (Age >= 18) who survived.
7. Get the median age of survivors.
8. Get the median age of non-survivors.
9. Get the median fare of survivors.
10. Get the median fare of non-survivors.

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

titanicData...

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

#1. Get the number of survivors by gender
survivors_by_gender = data[data['Survived'] == 1]['Sex'].value_counts()
print(survivors_by_gender)

# 2. Get the number of non-survivors by gender
non_survivors_by_gender = data[data['Survived'] == 0]['Sex'].value_counts()
print(non_survivors_by_gender)

#3. Get the number of survivors by embarked location (Embarked_S)
survivors_by_embarked_s = data[data['Survived'] == 1]['Embarked_S'].value_counts()
print(survivors_by_embarked_s)

# 4. Get the number of non-survivors by embarked location (Embarked_S)
non_survivors_by_embarked_s = data[data['Survived'] == 0]['Embarked_S'].value_counts()
print(non_survivors_by_embarked_s)

# 5. Percentage of children (Age < 18) who survived
children = data[data['Age'] < 18]
children_survival_rate = children['Survived'].mean()
print(children_survival_rate)

# 6. Percentage of adults (Age >= 18) who survived
adults = data[data['Age'] >= 18]
adults_survival_rate = adults['Survived'].mean()
print(adults_survival_rate)

# 7. Median age of survivors
median_age_survivors = data[data['Survived'] == 1]['Age'].median()
print(median_age_survivors)

# 8. Median age of non-survivors
```

Average time 0.372 s 372.00 ms | Maximum time 0.372 s 372.00 ms

1 out of 1 shown test case(s) passed

Test case 1 372 ms

Expected output

```
female    233
male      109
Name: Sex, dtype: int64
male      468
female     81
```

Actual output

```
female    233
male      109
Name: Sex, dtype: int64
male      468
female     81
```