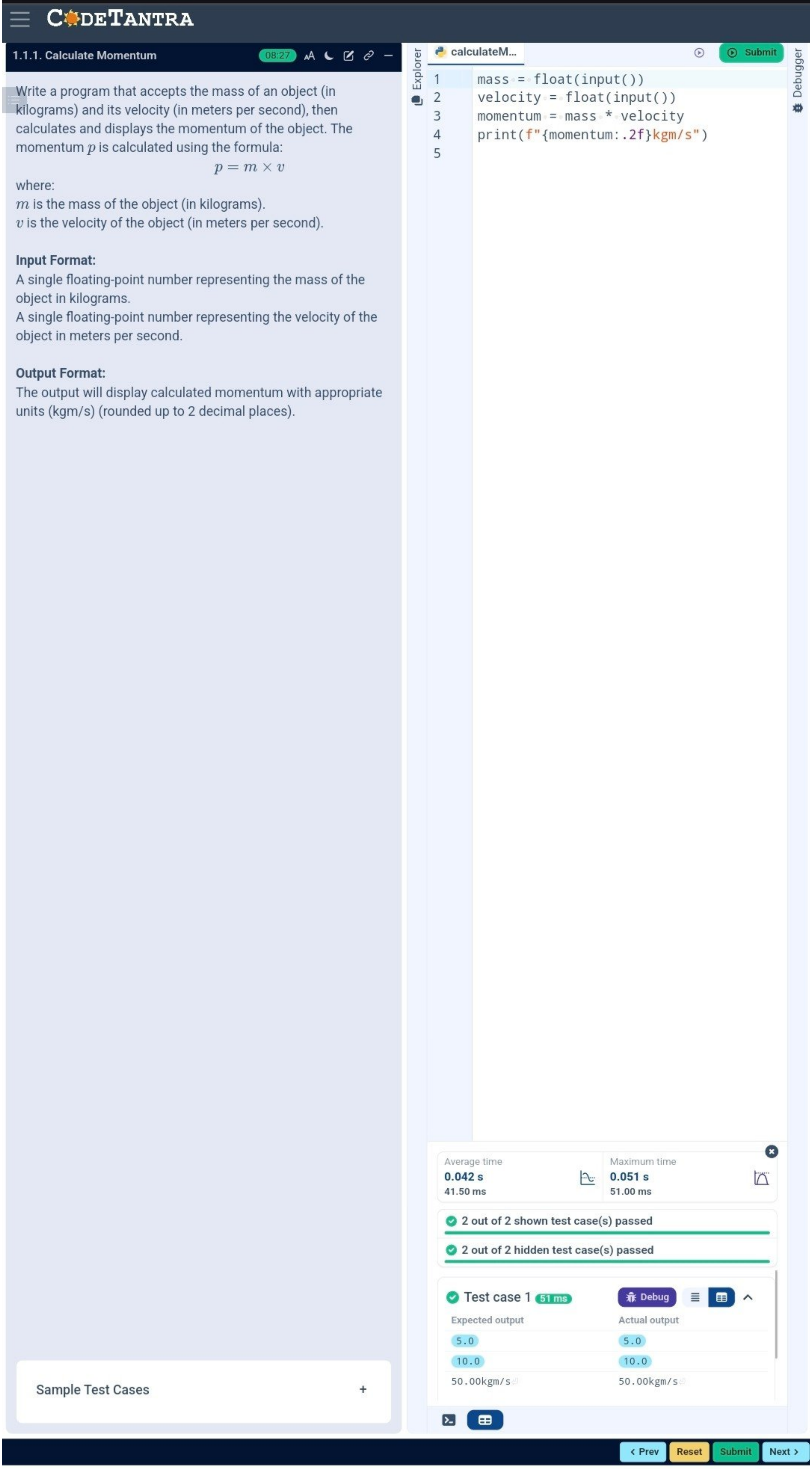


1.1.1. Calculate Momentum

Write a program that accepts the mass of an object (in kilograms) and its velocity (in meters per second), then calculates and displays the momentum of the object. The momentum p is calculated using the formula:

$$p = m \times v$$

where:
m is the mass of the object (in kilograms).
v is the velocity of the object (in meters per second).

Input Format:
A single floating-point number representing the mass of the object in kilograms.
A single floating-point number representing the velocity of the object in meters per second.

Output Format:
The output will display calculated momentum with appropriate units (kgm/s) (rounded up to 2 decimal places).

Sample Test Cases +

```python
mass = float(input())
velocity = float(input())
momentum = mass * velocity
print(f"{momentum:.2f}kgm/s")
```

Average time
0.042 s
41.50 ms

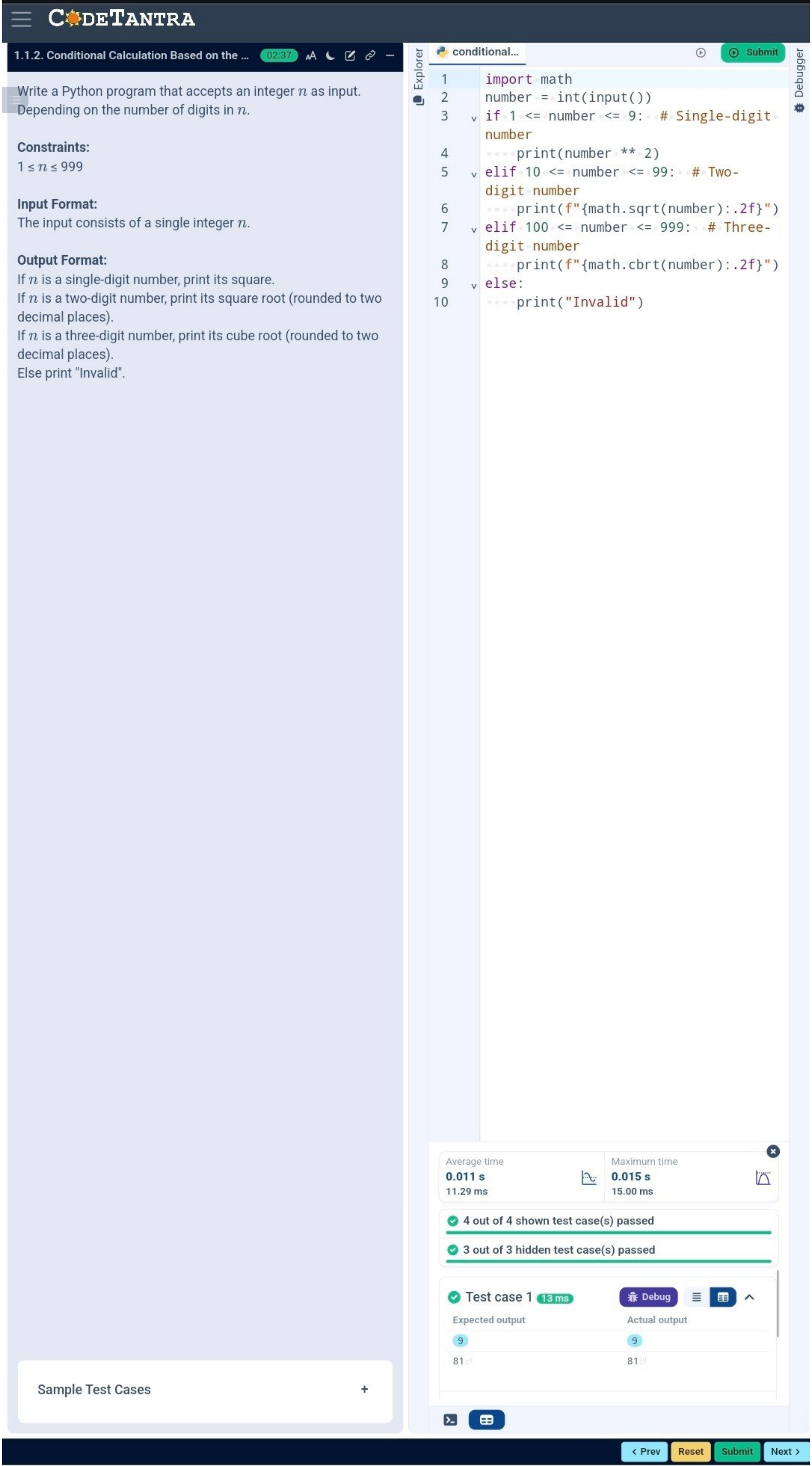
Maximum time
0.051 s
51.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 51 ms Debug

Expected output	Actual output
5.0	5.0
10.0	10.0
50.00kgm/s	50.00kgm/s



Write a Python program that accepts an integer n as input. Depending on the number of digits in n.

Constraints:
$1 \le n \le 999$

Input Format:
The input consists of a single integer n.

Output Format:
If n is a single-digit number, print its square.
If n is a two-digit number, print its square root (rounded to two decimal places).
If n is a three-digit number, print its cube root (rounded to two decimal places).
Else print "Invalid".

```python
import math
number = int(input())
if 1 <= number <= 9:  # Single-digit number
    print(number ** 2)
elif 10 <= number <= 99:  # Two-digit number
    print(f"{math.sqrt(number):.2f}")
elif 100 <= number <= 999:  # Three-digit number
    print(f"{math.cbrt(number):.2f}")
else:
    print("Invalid")
```

Average time
0.011 s
11.29 ms

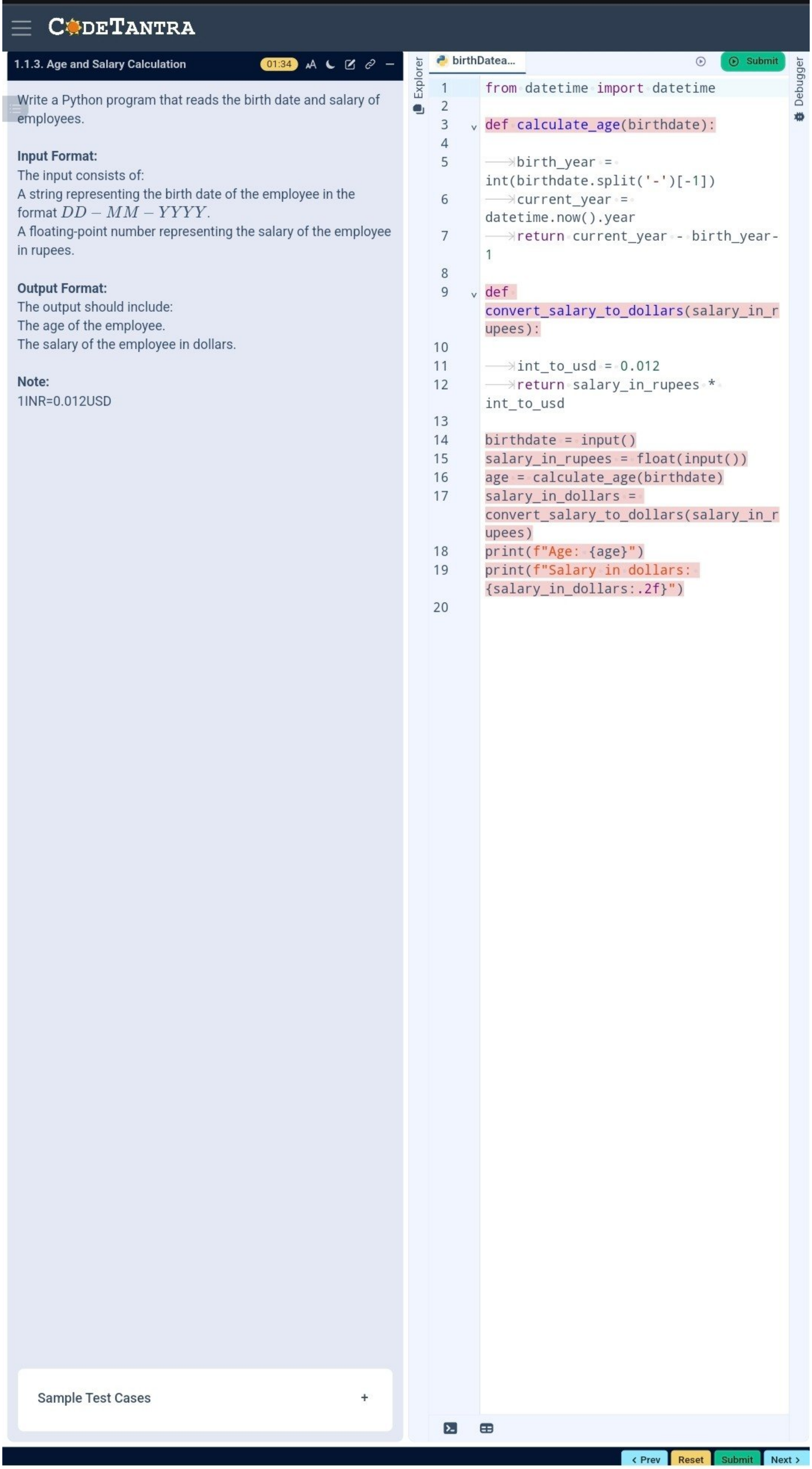
Maximum time
0.015 s
15.00 ms

4 out of 4 shown test case(s) passed

3 out of 3 hidden test case(s) passed

Test case 1 13 ms

Expected output	Actual output
9	9
81	81



1.1.3. Age and Salary Calculation

Write a Python program that reads the birth date and salary of employees.

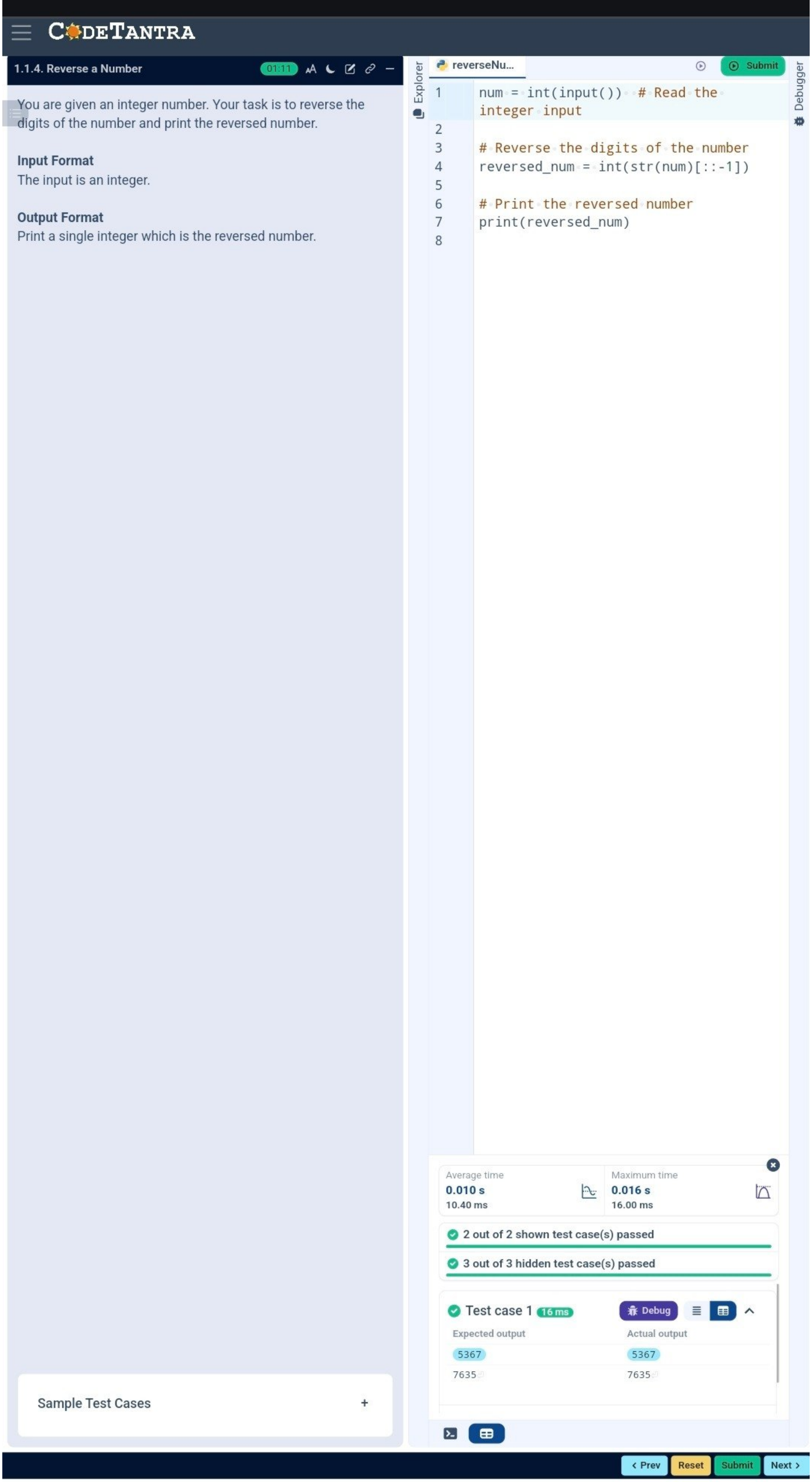
Input Format:
The input consists of:
A string representing the birth date of the employee in the format $DD - MM - YYYY$.
A floating-point number representing the salary of the employee in rupees.

Output Format:
The output should include:
The age of the employee.
The salary of the employee in dollars.

Note:
1INR=0.012USD

Sample Test Cases +

```python
from datetime import datetime

def calculate_age(birthdate):

    birth_year = int(birthdate.split('-')[-1])
    current_year = datetime.now().year
    return current_year - birth_year-1

def convert_salary_to_dollars(salary_in_rupees):

    int_to_usd = 0.012
    return salary_in_rupees * int_to_usd

birthdate = input()
salary_in_rupees = float(input())
age = calculate_age(birthdate)
salary_in_dollars = convert_salary_to_dollars(salary_in_rupees)
print(f"Age: {age}")
print(f"Salary in dollars: {salary_in_dollars:.2f}")
```



1.1.4. Reverse a Number

You are given an integer number. Your task is to reverse the digits of the number and print the reversed number.

Input Format
The input is an integer.

Output Format
Print a single integer which is the reversed number.

```python
num = int(input())  # Read the integer input

# Reverse the digits of the number
reversed_num = int(str(num)[::-1])

# Print the reversed number
print(reversed_num)
```

Average time 0.010 s 10.40 ms

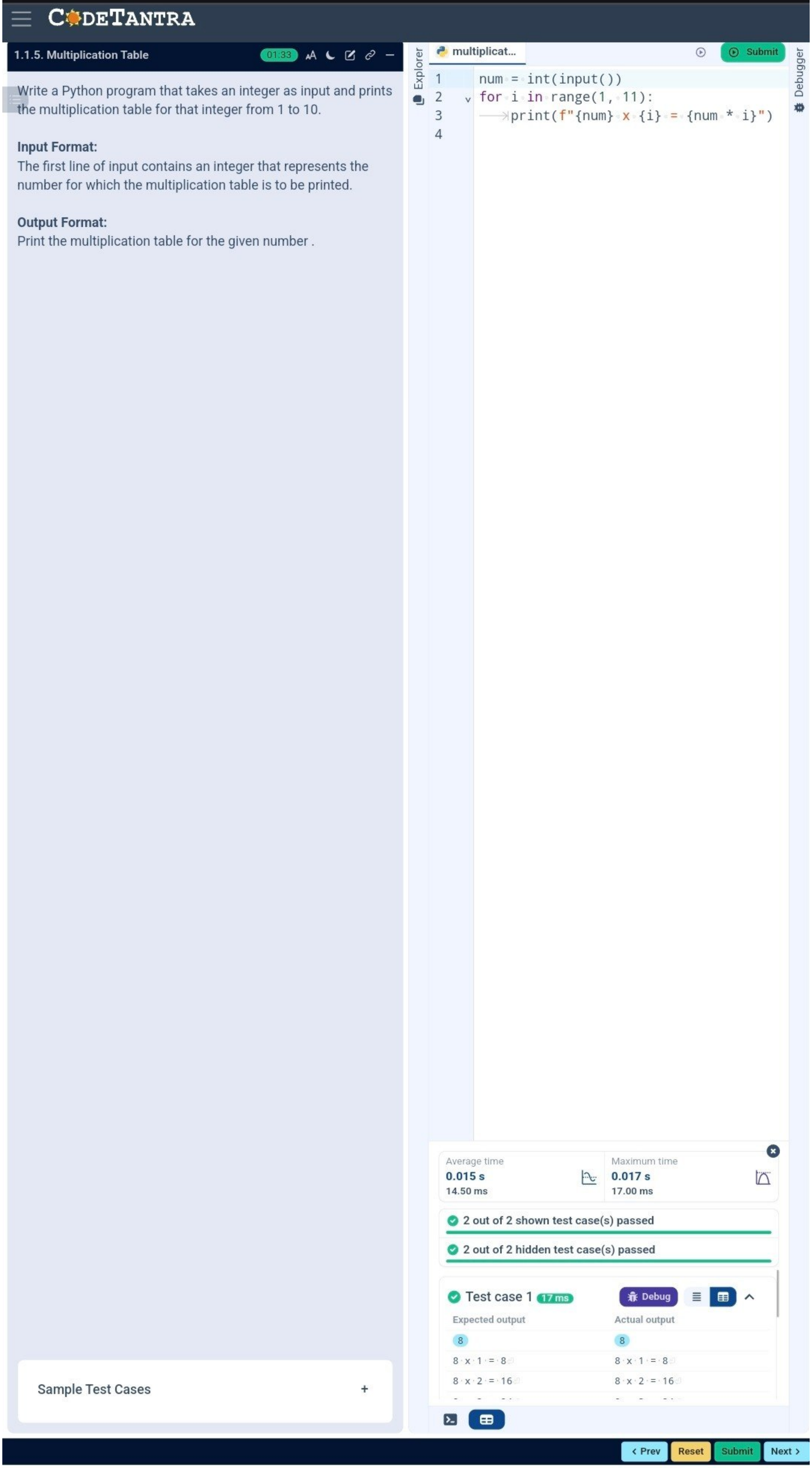
Maximum time 0.016 s 16.00 ms

2 out of 2 shown test case(s) passed

3 out of 3 hidden test case(s) passed

Test case 1 16 ms

Expected output	Actual output
5367	5367
7635	7635



1.1.5. Multiplication Table

Write a Python program that takes an integer as input and prints the multiplication table for that integer from 1 to 10.

Input Format:
The first line of input contains an integer that represents the number for which the multiplication table is to be printed.

Output Format:
Print the multiplication table for the given number .

Sample Test Cases

```python
num = int(input())
for i in range(1, 11):
	print(f"{num} x {i} = {num * i}")
```

Average time
0.015 s
14.50 ms

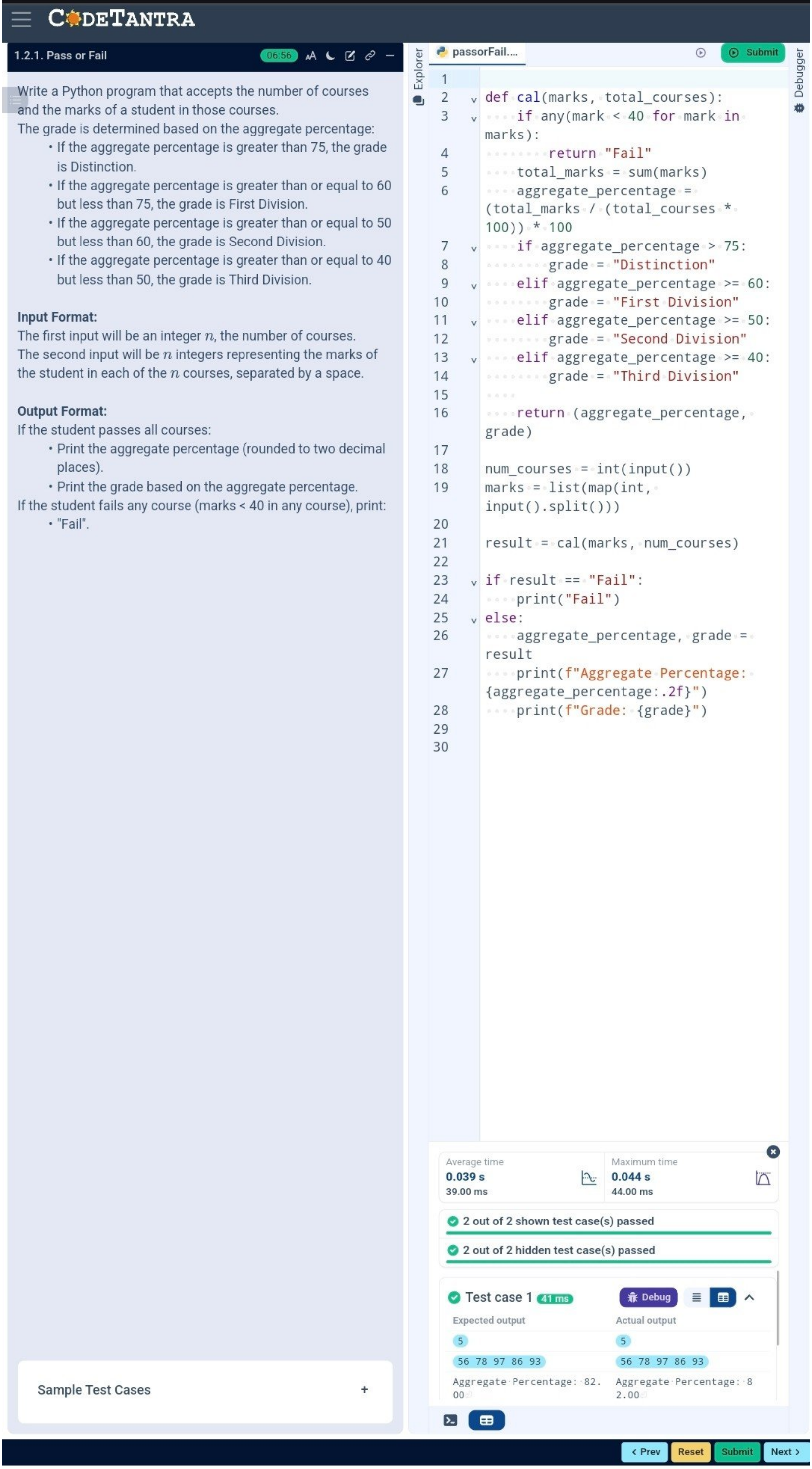
Maximum time
0.017 s
17.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 17 ms

Expected output	Actual output
8	8
8 x 1 = 8	8 x 1 = 8
8 x 2 = 16	8 x 2 = 16



Write a Python program that accepts the number of courses and the marks of a student in those courses.
The grade is determined based on the aggregate percentage:

- If the aggregate percentage is greater than 75, the grade is Distinction.
- If the aggregate percentage is greater than or equal to 60 but less than 75, the grade is First Division.
- If the aggregate percentage is greater than or equal to 50 but less than 60, the grade is Second Division.
- If the aggregate percentage is greater than or equal to 40 but less than 50, the grade is Third Division.

Input Format:
The first input will be an integer n, the number of courses.
The second input will be n integers representing the marks of the student in each of the n courses, separated by a space.

Output Format:
If the student passes all courses:

- Print the aggregate percentage (rounded to two decimal places).
- Print the grade based on the aggregate percentage.

If the student fails any course (marks < 40 in any course), print:

- "Fail".

Sample Test Cases +

```python

def cal(marks, total_courses):
    if any(mark < 40 for mark in marks):
        return "Fail"
    total_marks = sum(marks)
    aggregate_percentage = (total_marks / (total_courses * 100)) * 100
    if aggregate_percentage > 75:
        grade = "Distinction"
    elif aggregate_percentage >= 60:
        grade = "First Division"
    elif aggregate_percentage >= 50:
        grade = "Second Division"
    elif aggregate_percentage >= 40:
        grade = "Third Division"
    
    return (aggregate_percentage, grade)

num_courses = int(input())
marks = list(map(int, input().split()))

result = cal(marks, num_courses)

if result == "Fail":
    print("Fail")
else:
    aggregate_percentage, grade = result
    print(f"Aggregate Percentage: {aggregate_percentage:.2f}")
    print(f"Grade: {grade}")



```

Average time
0.039 s
39.00 ms

Maximum time
0.044 s
44.00 ms

2 out of 2 shown test case(s) passed

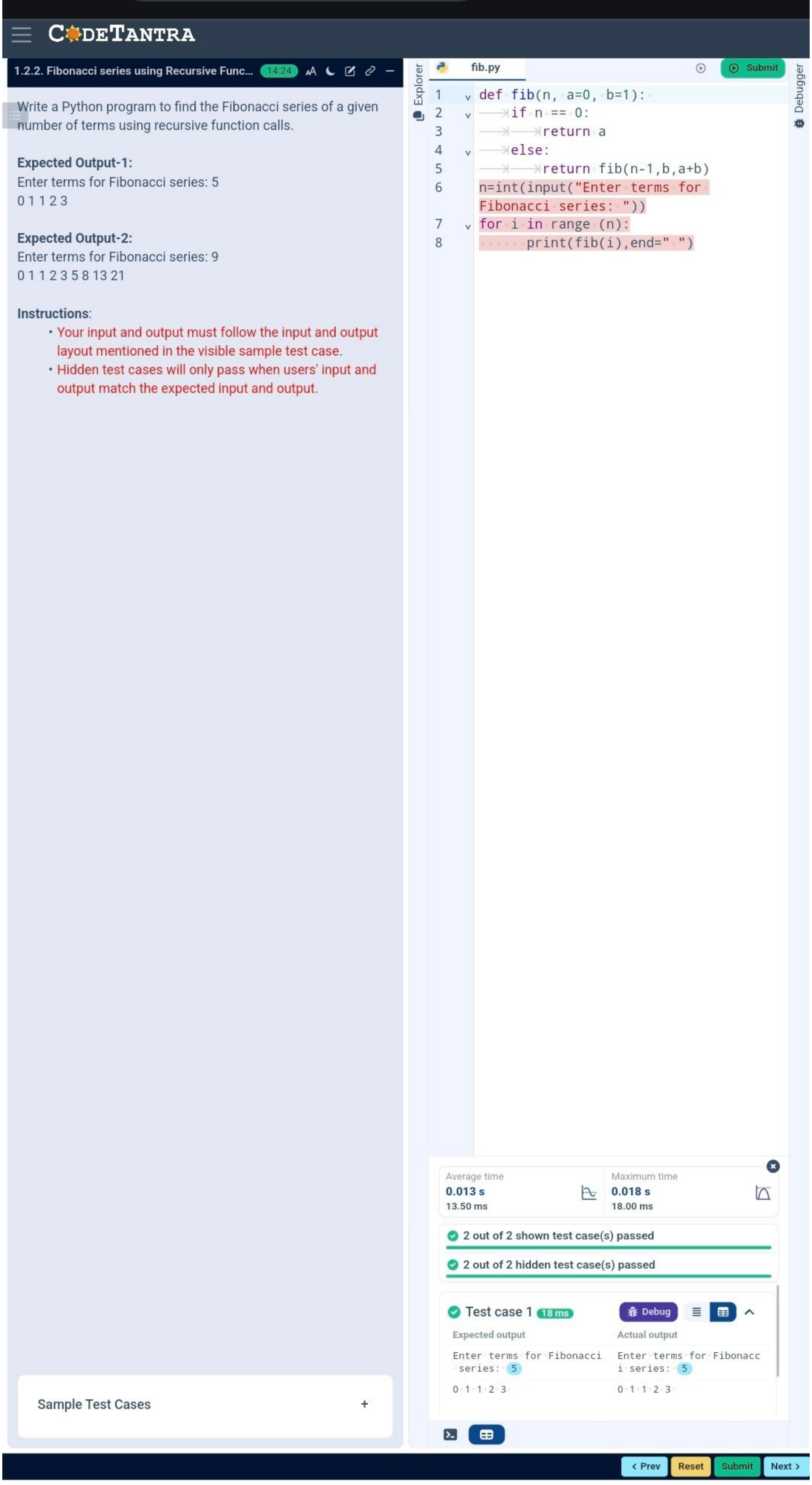
2 out of 2 hidden test case(s) passed

Test case 1 41 ms

Debug

Expected output	Actual output
5	5
56 78 97 86 93	56 78 97 86 93
Aggregate Percentage: 82.00	Aggregate Percentage: 82.00

< Prev Reset Submit Next >



Write a Python program to find the Fibonacci series of a given number of terms using recursive function calls.

Expected Output-1:
Enter terms for Fibonacci series: 5
0 1 1 2 3

Expected Output-2:
Enter terms for Fibonacci series: 9
0 1 1 2 3 5 8 13 21

Instructions:
- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when users' input and output match the expected input and output.

```python
def fib(n, a=0, b=1):
	if n == 0:
		return a
	else:
		return fib(n-1,b,a+b)
n=int(input("Enter terms for Fibonacci series: "))
for i in range (n):
      print(fib(i),end=" ")
```

Average time
0.013 s
13.50 ms

Maximum time
0.018 s
18.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

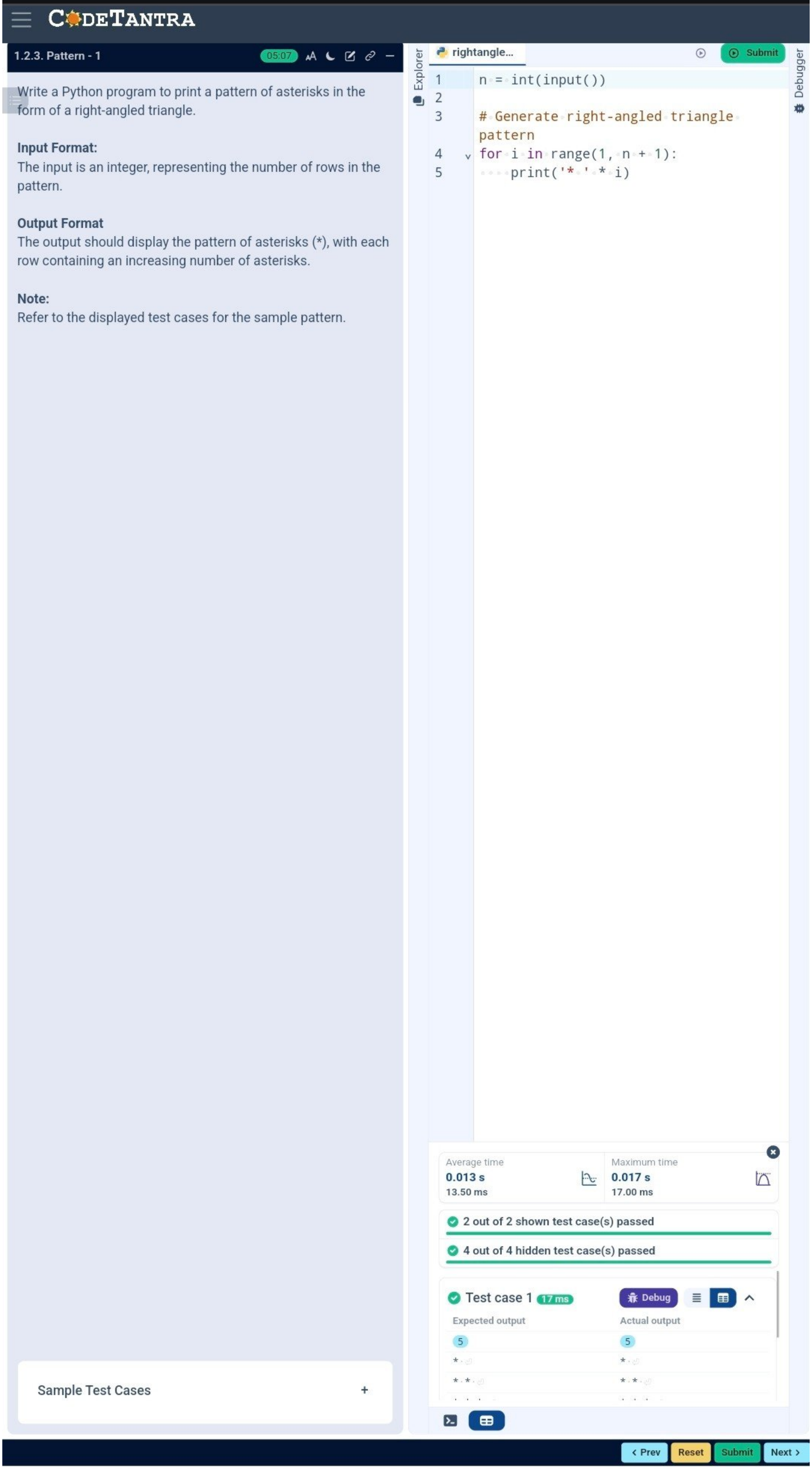
Test case 1 18 ms

Expected output

```
Enter terms for Fibonacci series: 5
0 1 1 2 3
```

Actual output

```
Enter terms for Fibonacci series: 5
0 1 1 2 3
```



Write a Python program to print a pattern of asterisks in the form of a right-angled triangle.

Input Format:
The input is an integer, representing the number of rows in the pattern.

Output Format
The output should display the pattern of asterisks (*), with each row containing an increasing number of asterisks.

Note:
Refer to the displayed test cases for the sample pattern.

```python
n = int(input())

# Generate right-angled triangle pattern
for i in range(1, n + 1):
    print('* ' * i)
```

Average time
0.013 s
13.50 ms

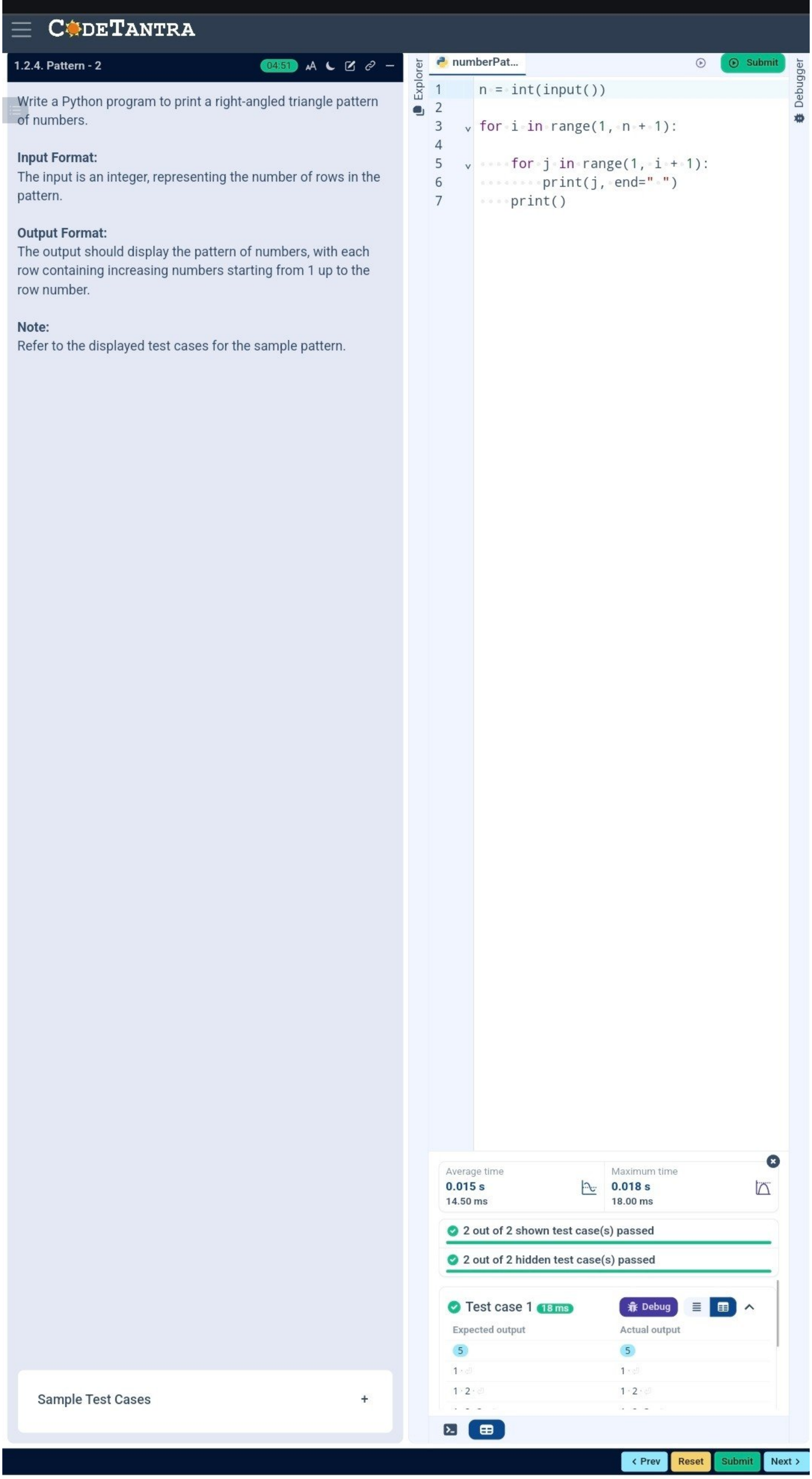
Maximum time
0.017 s
17.00 ms

2 out of 2 shown test case(s) passed

4 out of 4 hidden test case(s) passed

Test case 1 17 ms

Expected output	Actual output
5	5
*	*
* *	* *



1.2.4. Pattern - 2

Write a Python program to print a right-angled triangle pattern of numbers.

Input Format:
The input is an integer, representing the number of rows in the pattern.

Output Format:
The output should display the pattern of numbers, with each row containing increasing numbers starting from 1 up to the row number.

Note:
Refer to the displayed test cases for the sample pattern.

Sample Test Cases +

numberPat...

```
n = int(input())

for i in range(1, n + 1):

    for j in range(1, i + 1):
        print(j, end=" ")
    print()
```

Average time
0.015 s
14.50 ms

Maximum time
0.018 s
18.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 18 ms

Expected output	Actual output
5	5
1	1
1 2	1 2