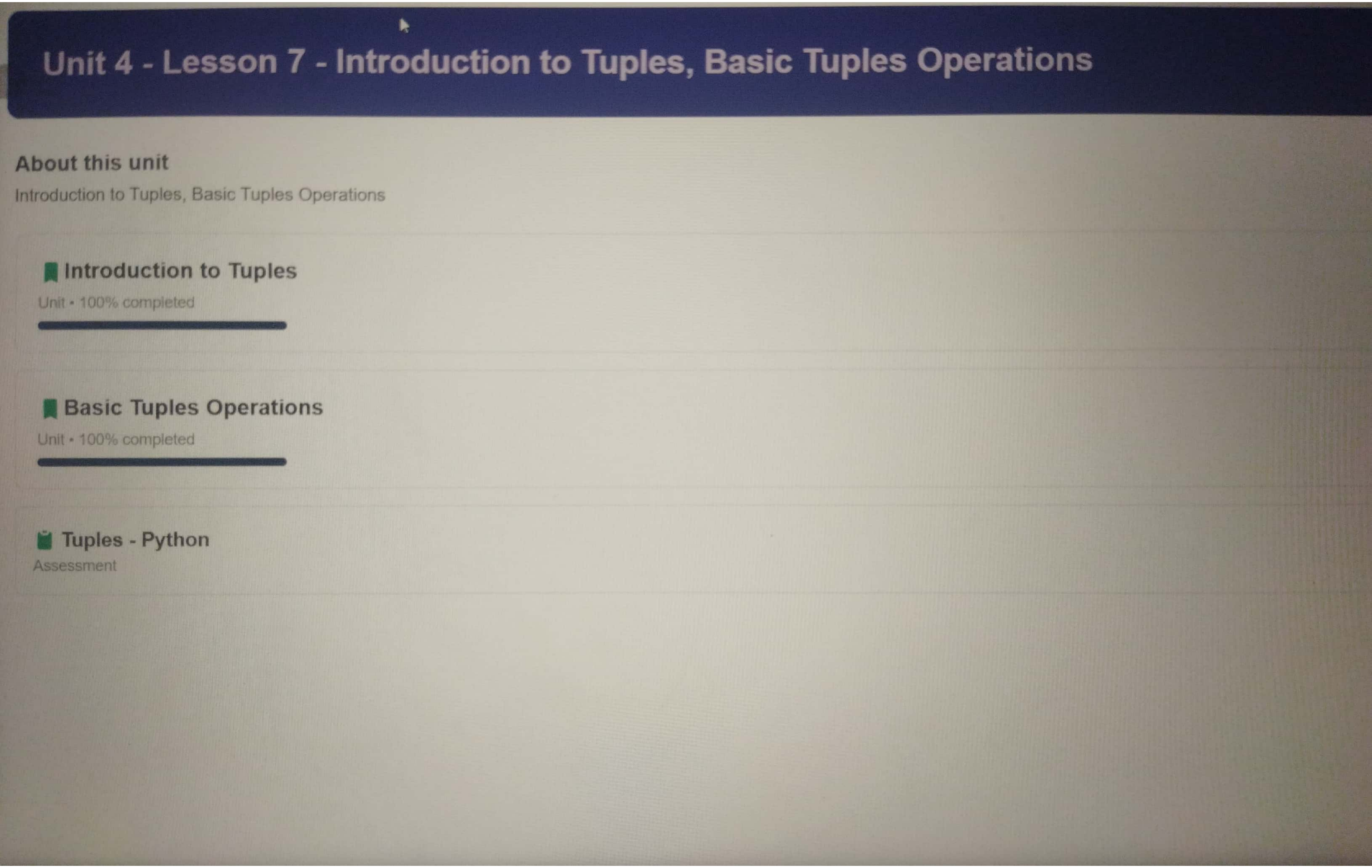


Unit 4 - Lesson 7 - Introduction to Tuples, Basic Tuples Operations

About this unit

Introduction to Tuples, Basic Tuples Operations

Introduction to Tuples

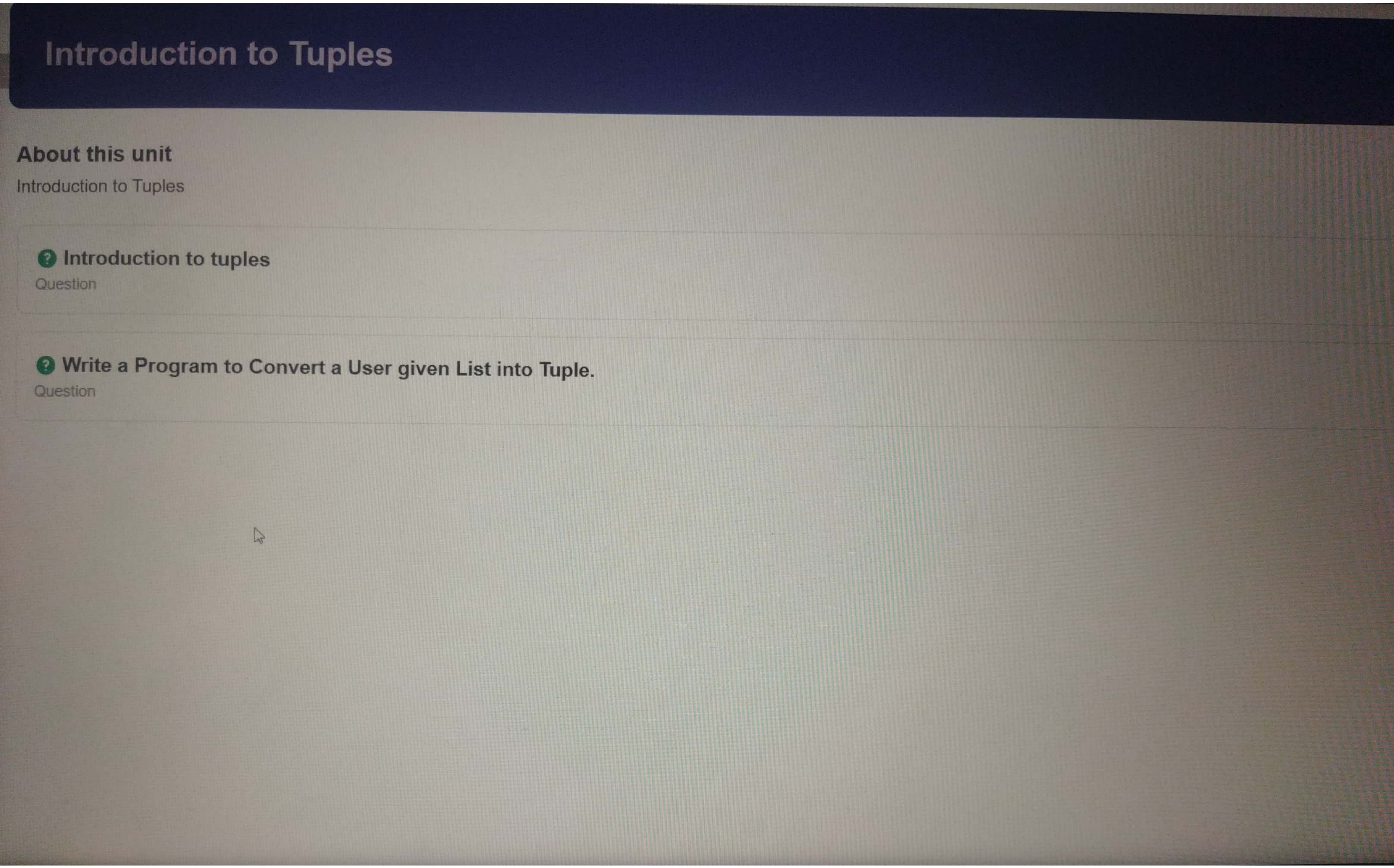
Unit • 100% completed

Basic Tuples Operations

Unit • 100% completed

Tuples - Python

Assessment



Introduction to Tuples

About this unit

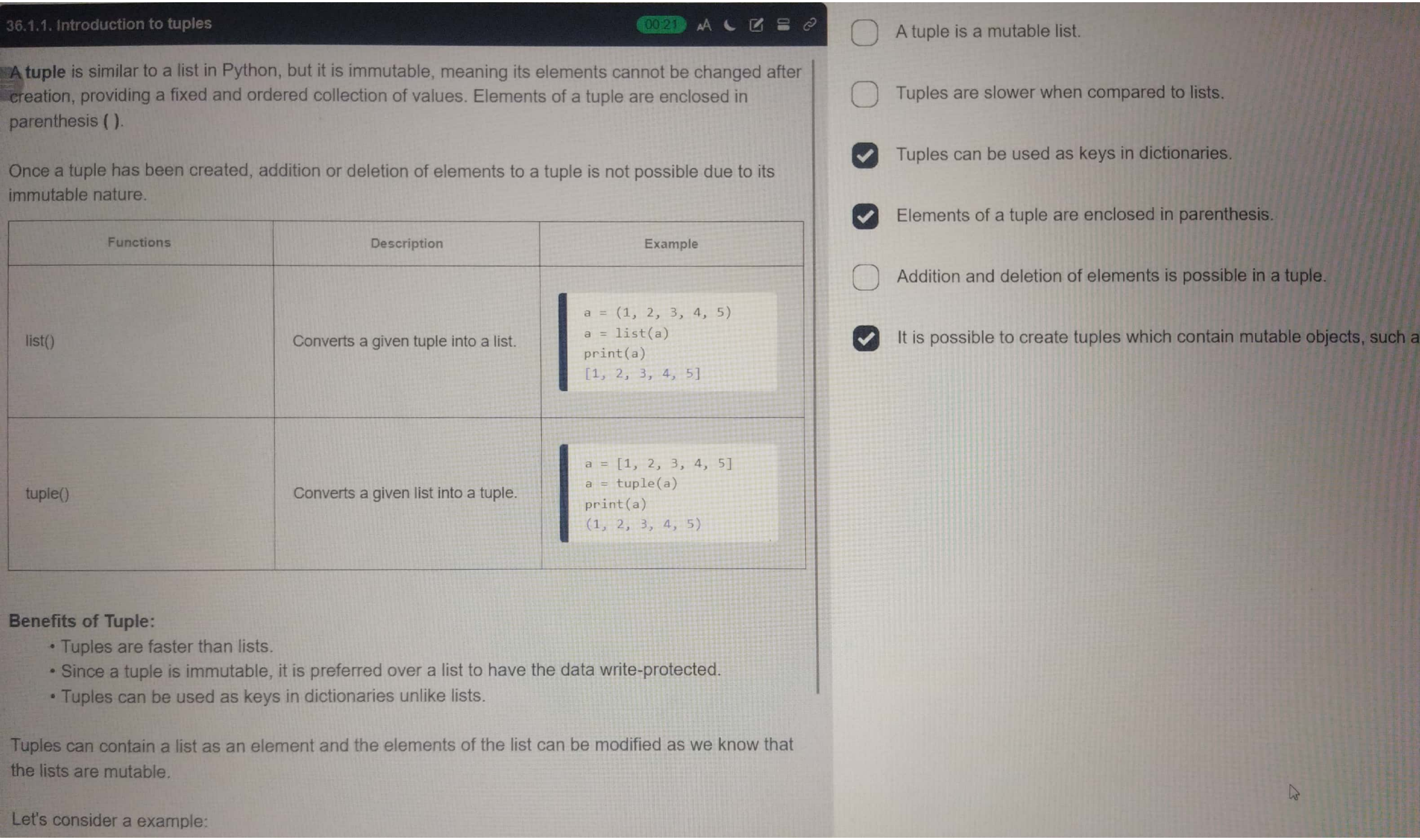
Introduction to Tuples

Introduction to tuples

Question

Write a Program to Convert a User given List into Tuple.

Question



36.1.1. Introduction to tuples

00:21

A **tuple** is similar to a list in Python, but it is immutable, meaning its elements cannot be changed after creation, providing a fixed and ordered collection of values. Elements of a tuple are enclosed in parenthesis **( )**.

Once a tuple has been created, addition or deletion of elements to a tuple is not possible due to its immutable nature.

Functions	Description	Example

list()	Converts a given tuple into a list.	`a = (1, 2, 3, 4, 5)`
		`a = list(a)`
		`print(a)`
		`[1, 2, 3, 4, 5]`

tuple()	Converts a given list into a tuple.	`a = [1, 2, 3, 4, 5]`
		`a = tuple(a)`
		`print(a)`
		`(1, 2, 3, 4, 5)`

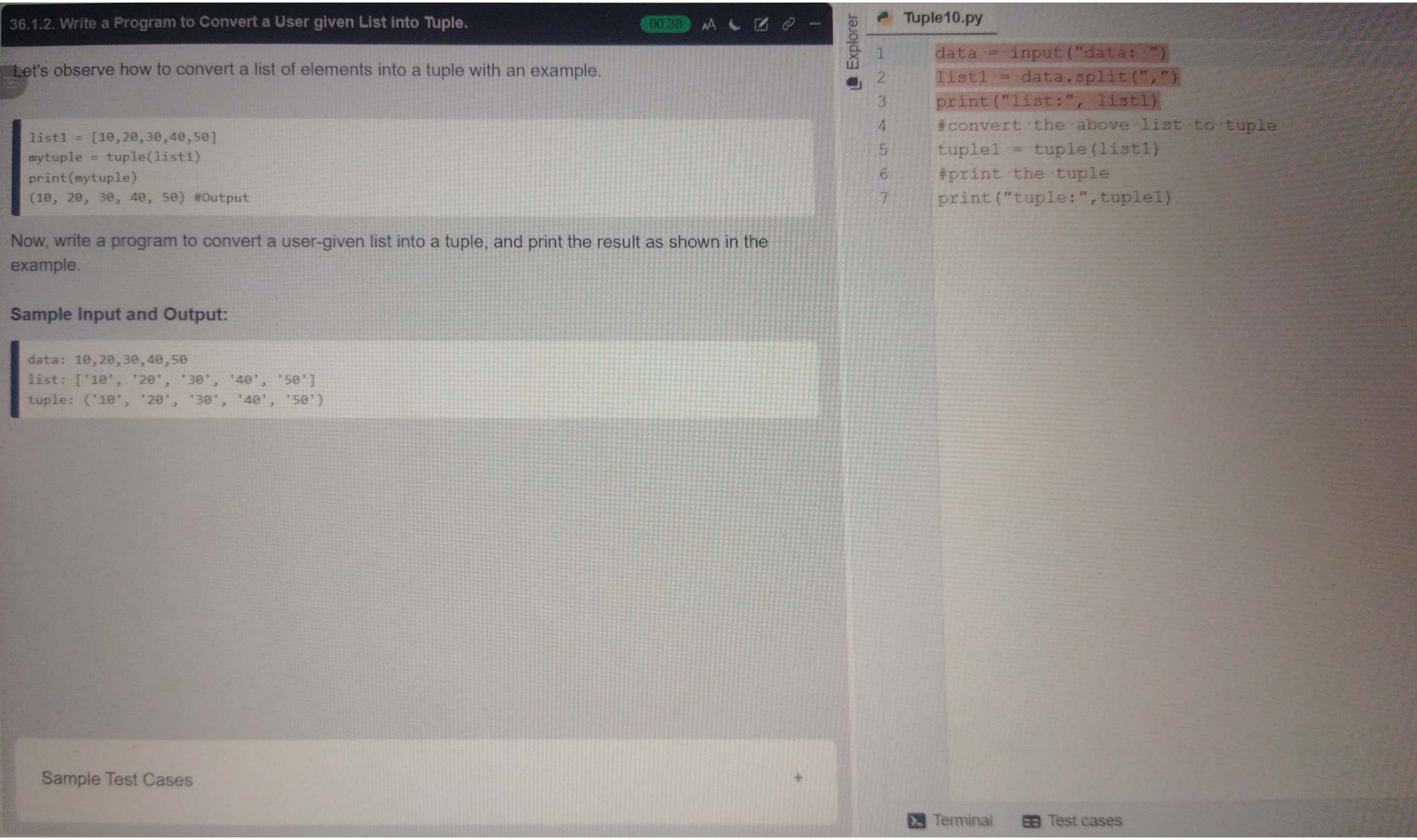
Benefits of Tuple:

- Tuples are faster than lists.
- Since a tuple is immutable, it is preferred over a list to have the data write-protected.
- Tuples can be used as keys in dictionaries unlike lists.

Tuples can contain a list as an element and the elements of the list can be modified as we know that the lists are mutable.

Let's consider a example:

- A tuple is a mutable list.
- Tuples are slower when compared to lists.
- [✓ Tuples can be used as keys in dictionaries.]
- [✓ Elements of a tuple are enclosed in parenthesis.]
- Addition and deletion of elements is possible in a tuple.
- [✓ It is possible to create tuples which contain mutable objects, such a]


36.1.2. Write a Program to Convert a User given List into Tuple.

Let's observe how to convert a list of elements into a tuple with an example.

```
list1 = [10,20,30,40,50]
mytuple = tuple(list1)
print(mytuple)
(10, 20, 30, 40, 50) #Output
```

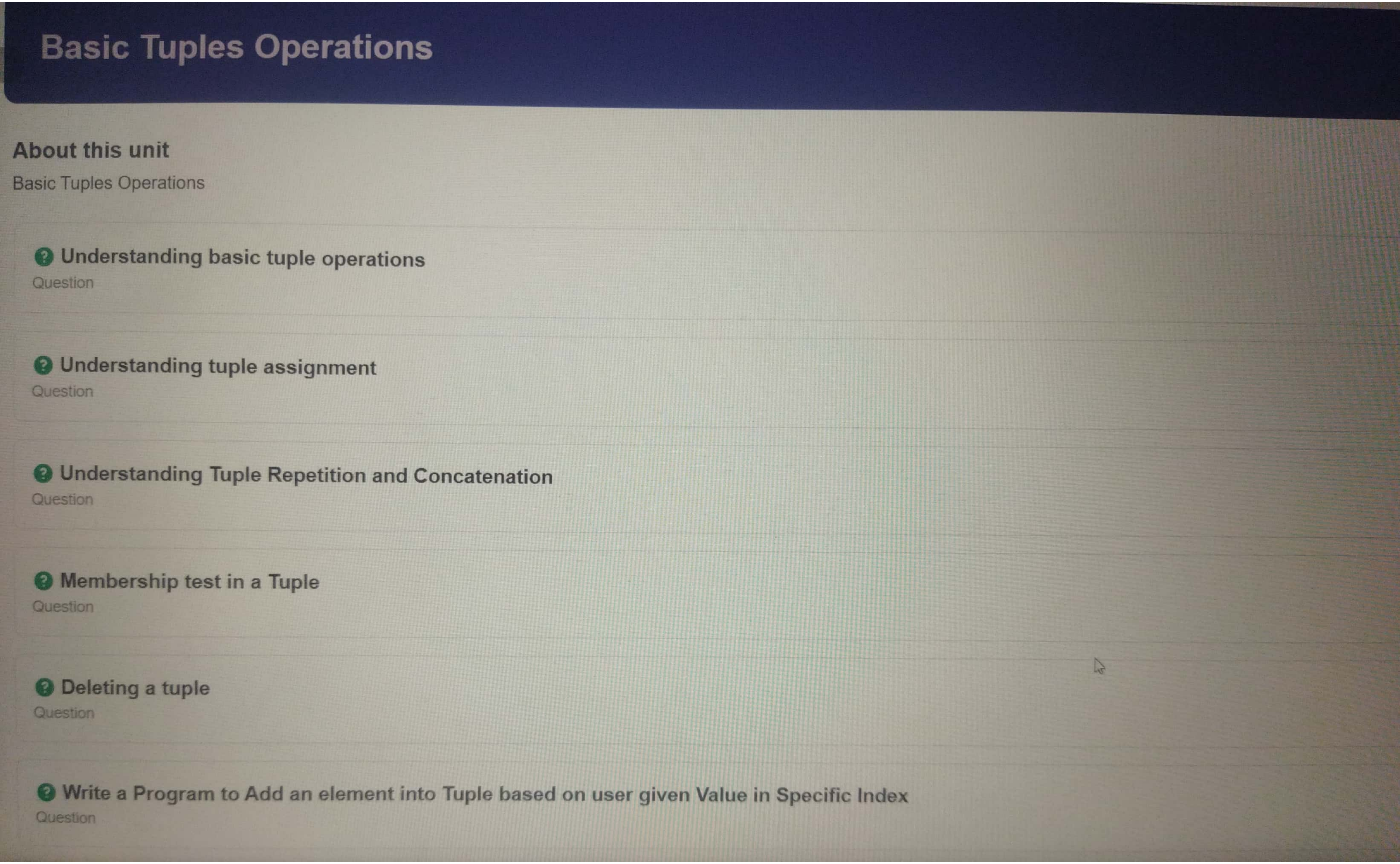
Now, write a program to convert a user-given list into a tuple, and print the result as shown in the example.

Sample Input and Output:

```
data: 10,20,30,40,50
list: ['10', '20', '30', '40', '50']
tuple: ('10', '20', '30', '40', '50')
```

Tuple10.py

```
data = input("data: ")
list1 = data.split(",")
print("list:", list1)
#convert the above list to tuple
tuple1 = tuple(list1)
#print the tuple
print("tuple:",tuple1)
```



Basic Tuples Operations

About this unit

Basic Tuples Operations

Understanding basic tuple operations
Question

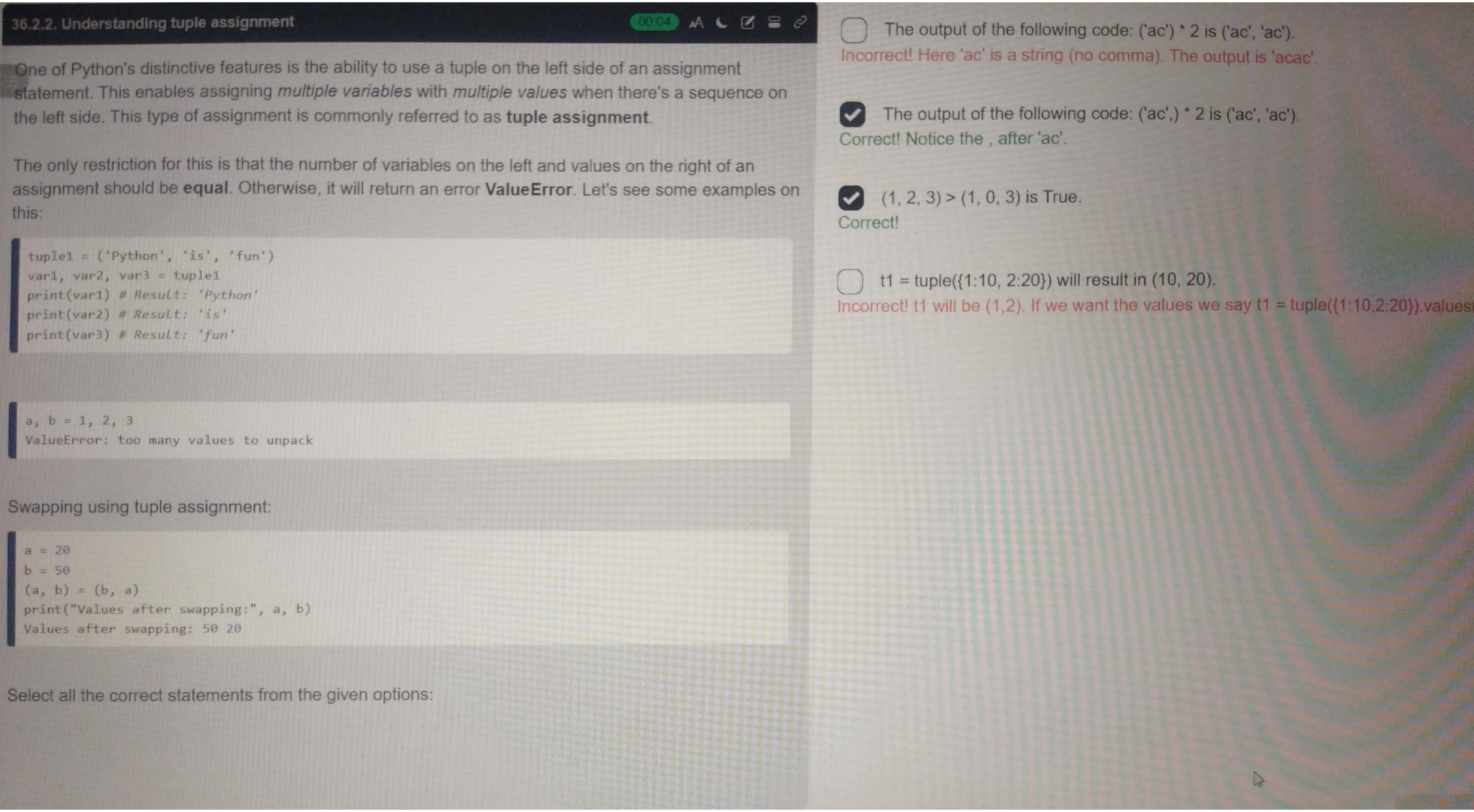
Understanding tuple assignment
Question

Understanding Tuple Repetition and Concatenation
Question

Membership test in a Tuple
Question

Deleting a tuple
Question

Write a Program to Add an element into Tuple based on user given Value in Specific Index
Question



36.2.2. Understanding tuple assignment

One of Python's distinctive features is the ability to use a tuple on the left side of an assignment statement. This enables assigning *multiple variables* with *multiple values* when there's a sequence on the left side. This type of assignment is commonly referred to as **tuple assignment**.

The only restriction for this is that the number of variables on the left and values on the right of an assignment should be **equal**. Otherwise, it will return an error **ValueError**. Let's see some examples on this:

```
tuple1 = ('Python', 'is', 'fun')
var1, var2, var3 = tuple1
print(var1) # Result: 'Python'
print(var2) # Result: 'is'
print(var3) # Result: 'fun'
```

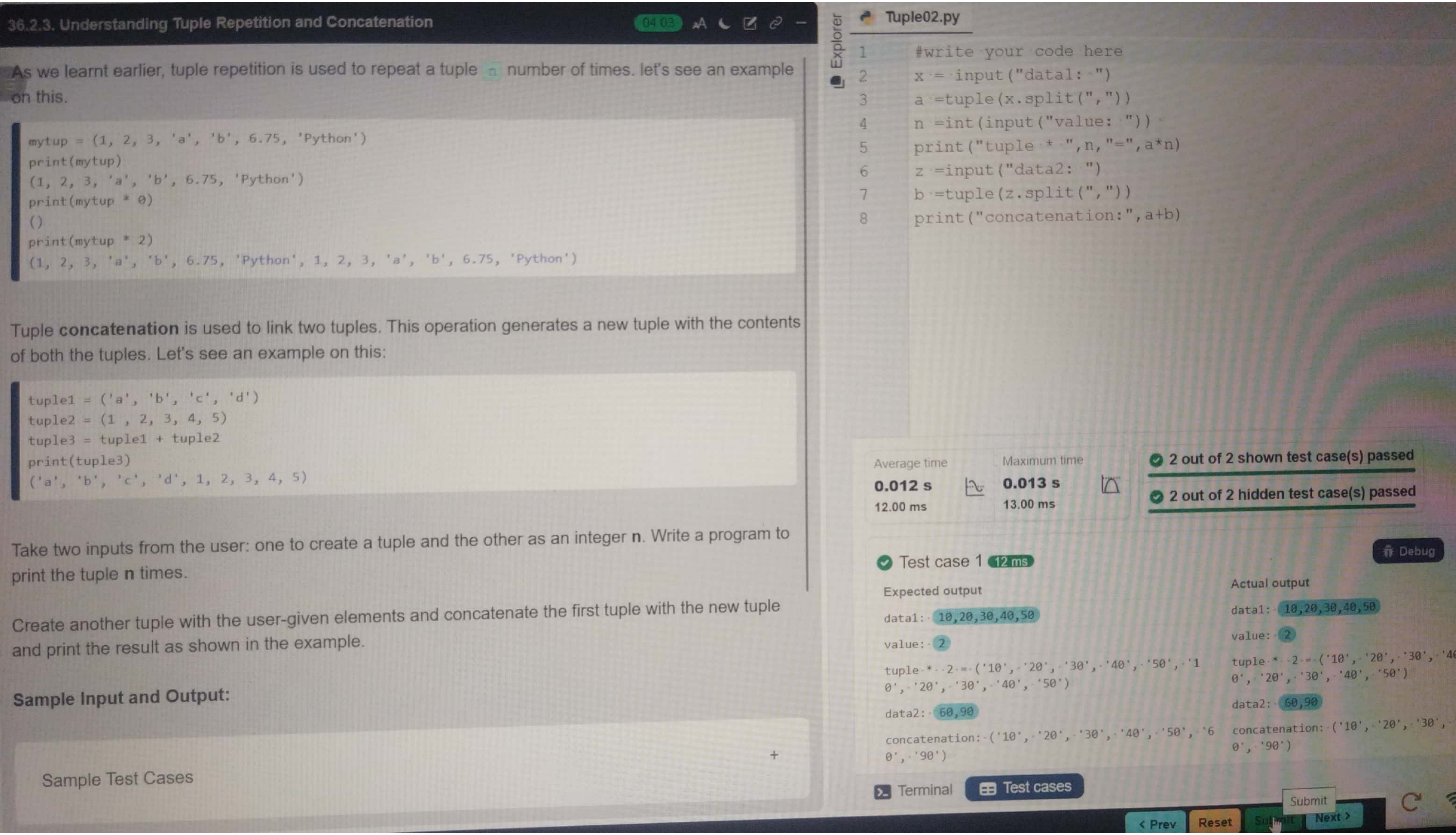
```
a, b = 1, 2, 3
ValueError: too many values to unpack
```

Swapping using tuple assignment:

```
a = 20
b = 50
(a, b) = (b, a)
print("Values after swapping:", a, b)
Values after swapping: 50 20
```

Select all the correct statements from the given options:

- [ ] The output of the following code: ('ac') * 2 is ('ac', 'ac').
  Incorrect! Here 'ac' is a string (no comma). The output is 'acac'.
- [✓] The output of the following code: ('ac',) * 2 is ('ac', 'ac').
  Correct! Notice the , after 'ac'.
- [✓] (1, 2, 3) > (1, 0, 3) is True.
  Correct!
- [ ] t1 = tuple({1:10, 2:20}) will result in (10, 20).
  Incorrect! t1 will be (1,2). If we want the values we say t1 = tuple({1:10,2:20}).values



36.2.3. Understanding Tuple Repetition and Concatenation

As we learnt earlier, tuple repetition is used to repeat a tuple `n` number of times. let's see an example on this.

```
mytup = (1, 2, 3, 'a', 'b', 6.75, 'Python')
print(mytup)
(1, 2, 3, 'a', 'b', 6.75, 'Python')
print(mytup * 0)
()
print(mytup * 2)
(1, 2, 3, 'a', 'b', 6.75, 'Python', 1, 2, 3, 'a', 'b', 6.75, 'Python')
```

Tuple **concatenation** is used to link two tuples. This operation generates a new tuple with the contents of both the tuples. Let's see an example on this:

```
tuple1 = ('a', 'b', 'c', 'd')
tuple2 = (1 , 2, 3, 4, 5)
tuple3 = tuple1 + tuple2
print(tuple3)
('a', 'b', 'c', 'd', 1, 2, 3, 4, 5)
```

Take two inputs from the user: one to create a tuple and the other as an integer **n**. Write a program to print the tuple **n** times.

Create another tuple with the user-given elements and concatenate the first tuple with the new tuple and print the result as shown in the example.

Sample Input and Output:

Sample Test Cases

Tuple02.py

```
#write your code here
x = input("data1: ")
a =tuple(x.split(","))
n =int(input("value: "))
print("tuple * ",n,"=",a*n)
z =input("data2: ")
b =tuple(z.split(","))
print("concatenation:",a+b)
```

Average time 0.012 s 12.00 ms

Maximum time 0.013 s 13.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

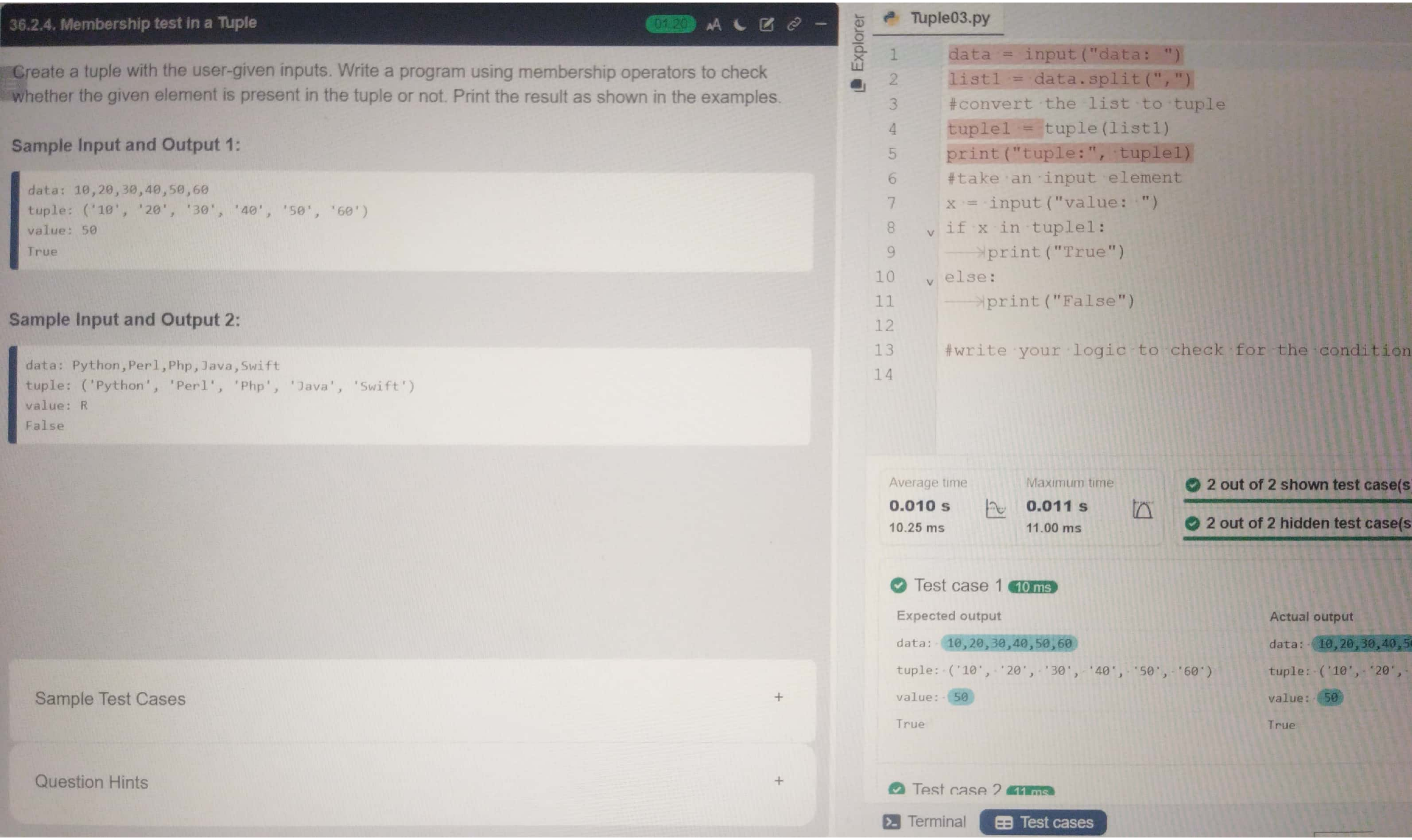
Test case 1 12 ms

Expected output

```
data1: 10,20,30,40,50
value: 2
tuple *  2 = ('10', '20', '30', '40', '50', '1
0', '20', '30', '40', '50')
data2: 60,90
concatenation: ('10', '20', '30', '40', '50', '6
0', '90')
```

Actual output

```
data1: 10,20,30,40,50
value: 2
tuple *  2 = ('10', '20', '30', '4
0', '20', '30', '40', '50')
data2: 60,90
concatenation: ('10', '20', '30',
0', '90')
```



36.2.4. Membership test in a Tuple

Create a tuple with the user-given inputs. Write a program using membership operators to check whether the given element is present in the tuple or not. Print the result as shown in the examples.

Sample Input and Output 1:

```
data: 10,20,30,40,50,60
tuple: ('10', '20', '30', '40', '50', '60')
value: 50
True
```

Sample Input and Output 2:

```
data: Python,Perl,Php,Java,Swift
tuple: ('Python', 'Perl', 'Php', 'Java', 'Swift')
value: R
False
```

Sample Test Cases +

Question Hints +

Tuple03.py

```python
data = input("data: ")
list1 = data.split(",")
#convert the list to tuple
tuple1 = tuple(list1)
print("tuple:", tuple1)
#take an input element
x = input("value: ")
if x in tuple1:
	print("True")
else:
	print("False")

#write your logic to check for the condition
```

Average time
0.010 s
10.25 ms

Maximum time
0.011 s
11.00 ms

✓ 2 out of 2 shown test case(s

✓ 2 out of 2 hidden test case(s

✓ Test case 1 10 ms

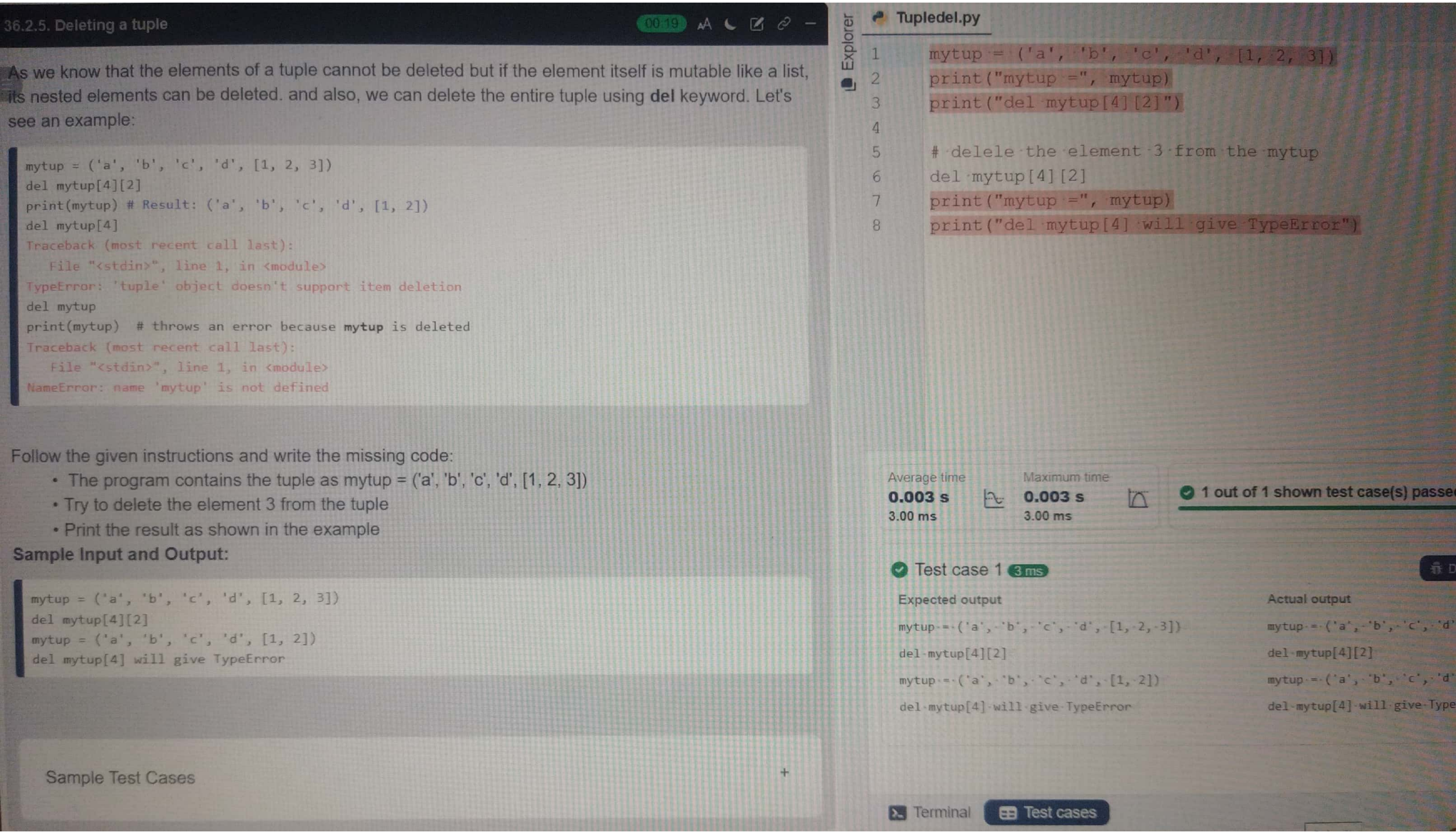
Expected output

```
data: 10,20,30,40,50,60
tuple: ('10', '20', '30', '40', '50', '60')
value: 50
True
```

Actual output

```
data: 10,20,30,40,5
tuple: ('10', '20', 
value: 50
True
```

✓ Test case 2 11 ms



36.2.5. Deleting a tuple

As we know that the elements of a tuple cannot be deleted but if the element itself is mutable like a list, its nested elements can be deleted. and also, we can delete the entire tuple using **del** keyword. Let's see an example:

```
mytup = ('a', 'b', 'c', 'd', [1, 2, 3])
del mytup[4][2]
print(mytup) # Result: ('a', 'b', 'c', 'd', [1, 2])
del mytup[4]
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
del mytup
print(mytup)  # throws an error because mytup is deleted
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
NameError: name 'mytup' is not defined
```

Follow the given instructions and write the missing code:

- The program contains the tuple as mytup = ('a', 'b', 'c', 'd', [1, 2, 3])
- Try to delete the element 3 from the tuple
- Print the result as shown in the example

Sample Input and Output:

```
mytup = ('a', 'b', 'c', 'd', [1, 2, 3])
del mytup[4][2]
mytup = ('a', 'b', 'c', 'd', [1, 2])
del mytup[4] will give TypeError
```

Sample Test Cases

Tupledel.py

```python
mytup = ('a', 'b', 'c', 'd', [1, 2, 3])
print("mytup =", mytup)
print("del mytup[4][2]")

# delele the element 3 from the mytup
del mytup[4][2]
print("mytup =", mytup)
print("del mytup[4] will give TypeError")
```

Average time 0.003 s 3.00 ms | Maximum time 0.003 s 3.00 ms

1 out of 1 shown test case(s) passe

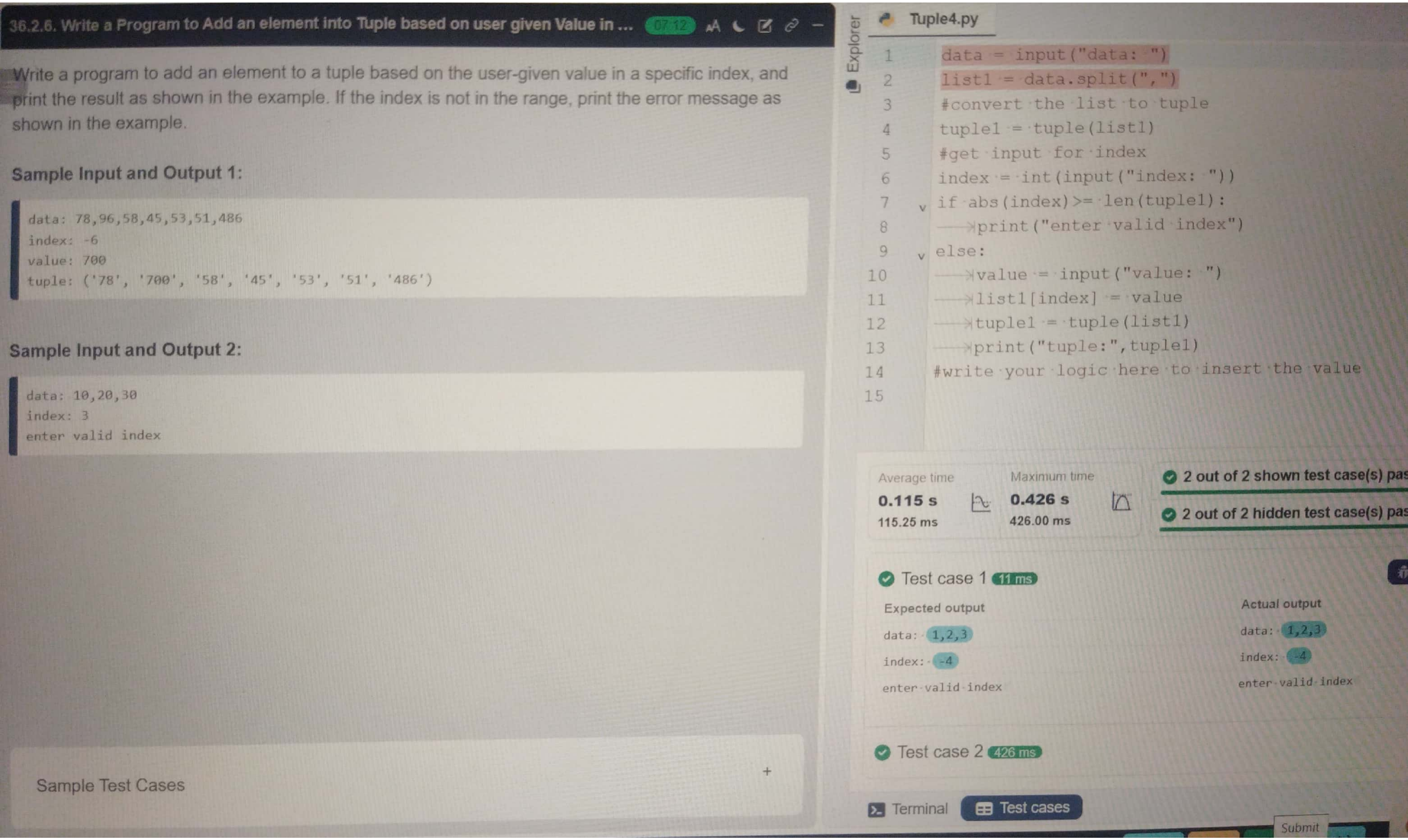
Test case 1 3 ms

Expected output

```
mytup = ('a', 'b', 'c', 'd', [1, 2, 3])
del mytup[4][2]
mytup = ('a', 'b', 'c', 'd', [1, 2])
del mytup[4] will give TypeError
```

Actual output

```
mytup = ('a', 'b', 'c', 'd'
del mytup[4][2]
mytup = ('a', 'b', 'c', 'd'
del mytup[4] will give Type
```



36.2.6. Write a Program to Add an element into Tuple based on user given Value in ... 07:12

Write a program to add an element to a tuple based on the user-given value in a specific index, and print the result as shown in the example. If the index is not in the range, print the error message as shown in the example.

Sample Input and Output 1:

```
data: 78,96,58,45,53,51,486
index: -6
value: 700
tuple: ('78', '700', '58', '45', '53', '51', '486')
```

Sample Input and Output 2:

```
data: 10,20,30
index: 3
enter valid index
```

Sample Test Cases +

Explorer

Tuple4.py

```python
data = input("data: ")
list1 = data.split(",")
#convert the list to tuple
tuple1 = tuple(list1)
#get input for index
index = int(input("index: "))
if abs(index)>= len(tuple1):
	print("enter valid index")
else:
	value = input("value: ")
	list1[index] = value
	tuple1 = tuple(list1)
	print("tuple:",tuple1)
#write your logic here to insert the value
```

Average time 0.115 s 115.25 ms

Maximum time 0.426 s 426.00 ms

2 out of 2 shown test case(s) pas

2 out of 2 hidden test case(s) pas

Test case 1 11 ms

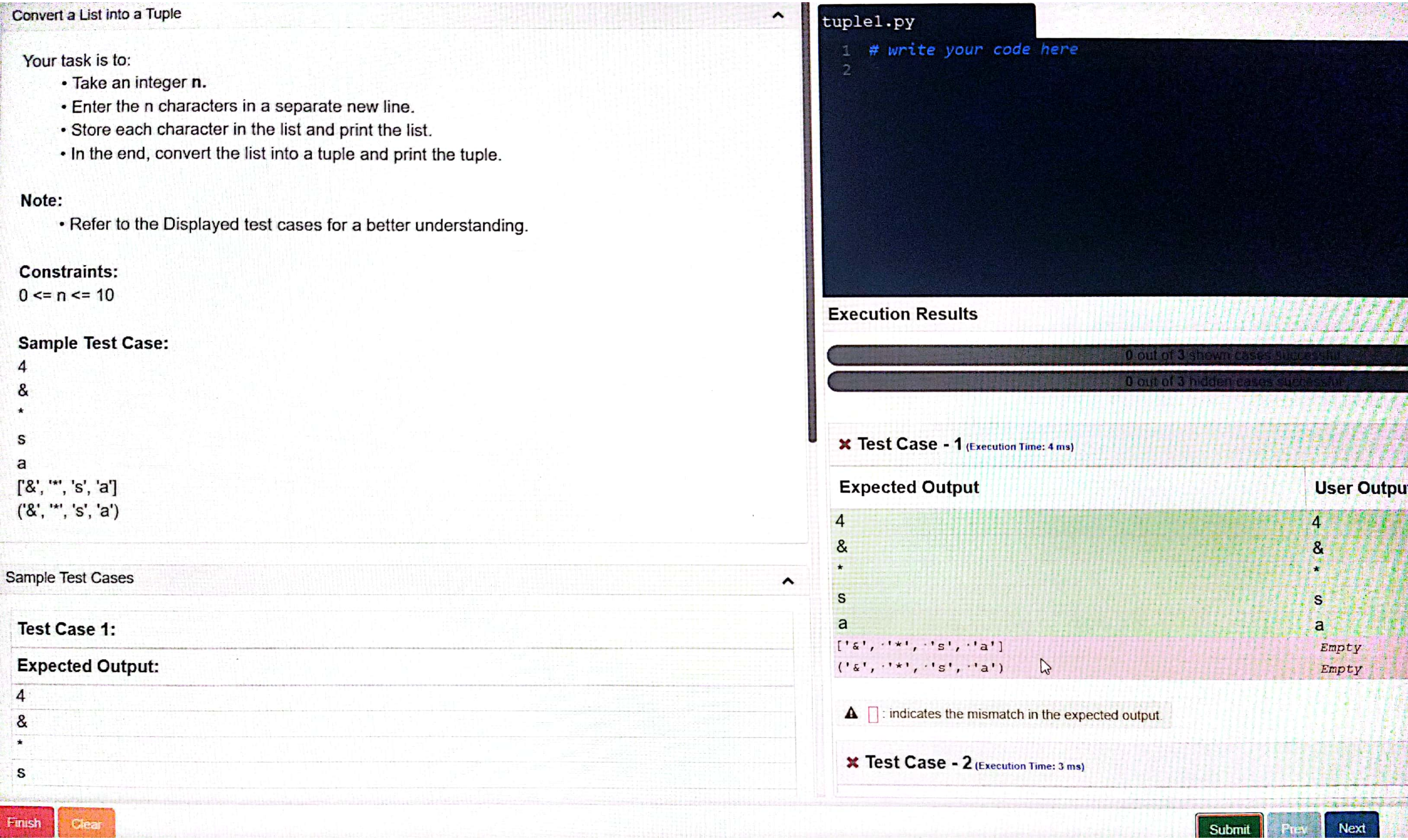
Expected output

```
data: 1,2,3
index: -4
enter valid index
```

Actual output

```
data: 1,2,3
index: -4
enter valid index
```

Test case 2 426 ms



Convert a List into a Tuple

Your task is to:

- Take an integer **n.**
- Enter the n characters in a separate new line.
- Store each character in the list and print the list.
- In the end, convert the list into a tuple and print the tuple.

Note:

- Refer to the Displayed test cases for a better understanding.

Constraints:
0 <= n <= 10

Sample Test Case:

```
4
&
*
s
a
['&', '*', 's', 'a']
('&', '*', 's', 'a')
```

Sample Test Cases

Test Case 1:

Expected Output:

```
4
&
*
s
```

tuple1.py

```python
# write your code here

```

Execution Results

0 out of 3 shown cases successful

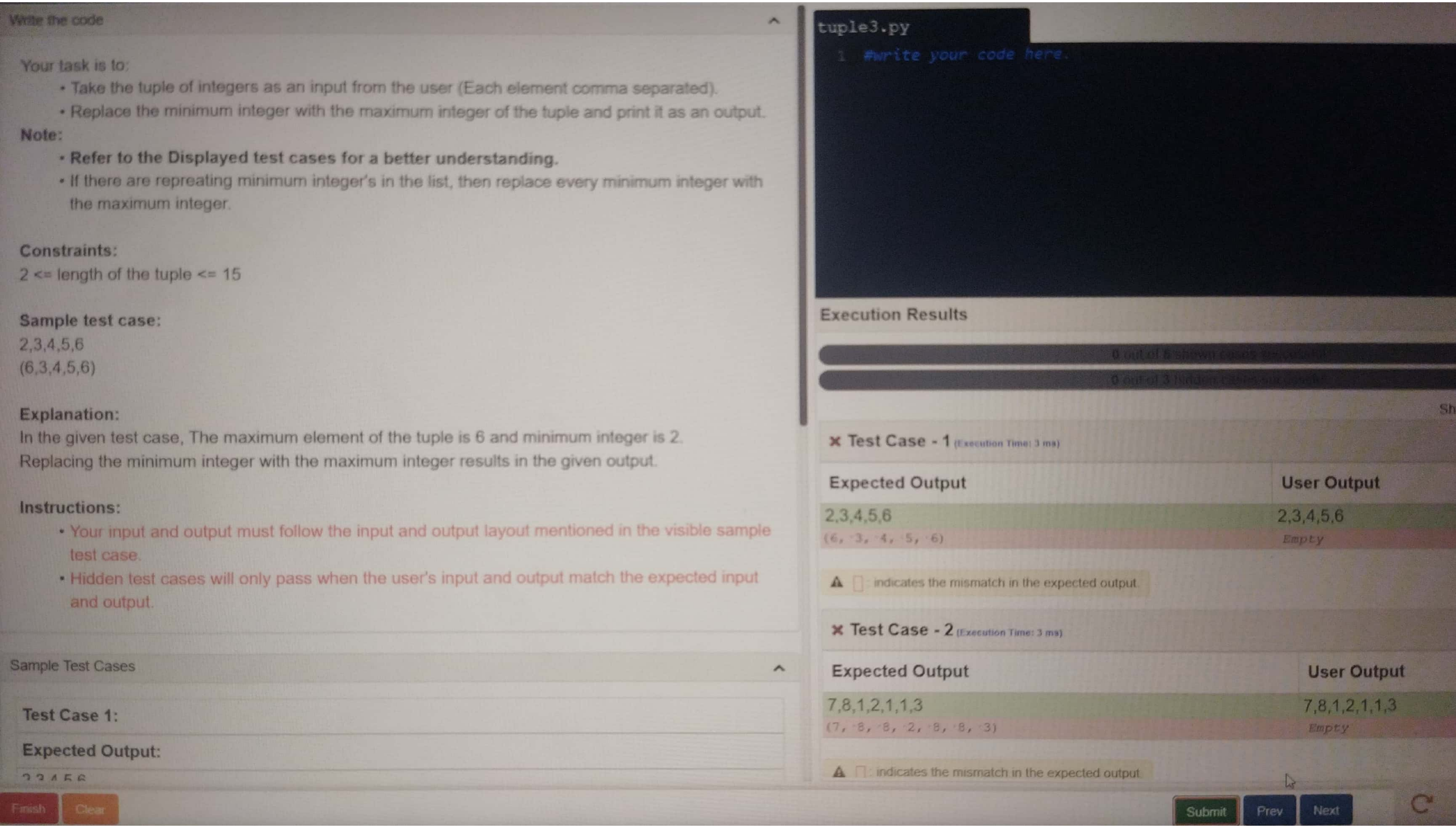
0 out of 3 hidden cases successful

Test Case - 1 (Execution Time: 4 ms)

Expected Output	User Output
4	4
&	&
*	*
s	s
a	a
['&', '*', 's', 'a']	*Empty*
('&', '*', 's', 'a')	*Empty*

⚠ ▯ : indicates the mismatch in the expected output

Test Case - 2 (Execution Time: 3 ms)



Write the code

Your task is to:

- Take the tuple of integers as an input from the user (Each element comma separated).
- Replace the minimum integer with the maximum integer of the tuple and print it as an output.

Note:

- **Refer to the Displayed test cases for a better understanding.**
- If there are repreating minimum integer's in the list, then replace every minimum integer with the maximum integer.

Constraints:

2 <= length of the tuple <= 15

Sample test case:

```
2,3,4,5,6
(6,3,4,5,6)
```

Explanation:

In the given test case, The maximum element of the tuple is 6 and minimum integer is 2.
Replacing the minimum integer with the maximum integer results in the given output.

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 the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:

Expected Output:

tuple3.py

```
#write your code here.
```

Execution Results

0 out of 6 shown cases successful

0 out of 3 hidden cases successful

✗ Test Case - 1 (Execution Time: 3 ms)

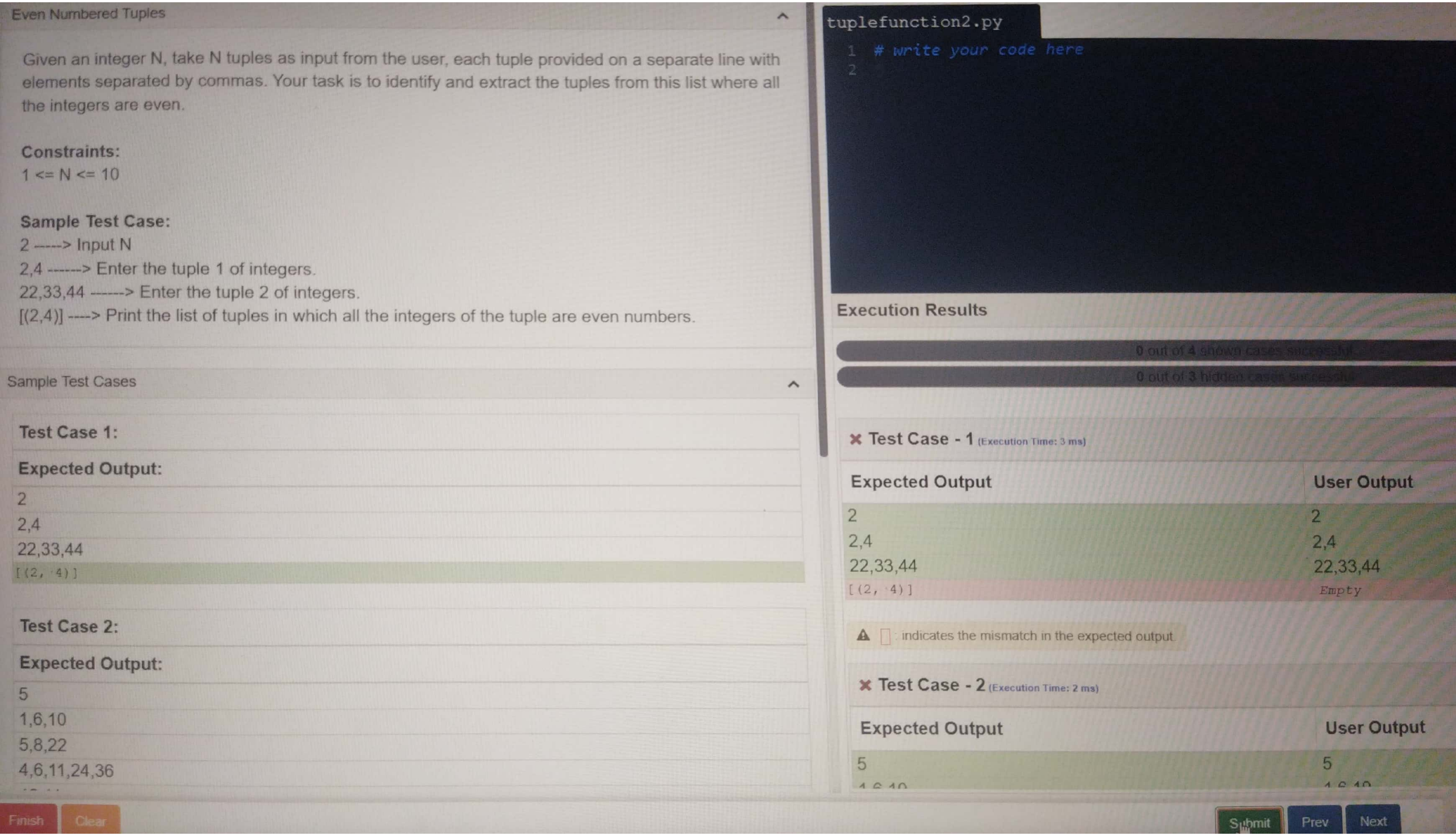
Expected Output	User Output
2,3,4,5,6	2,3,4,5,6
(6, 3, 4, 5, 6)	Empty

⚠ ▯ indicates the mismatch in the expected output.

✗ Test Case - 2 (Execution Time: 3 ms)

Expected Output	User Output
7,8,1,2,1,1,3	7,8,1,2,1,1,3
(7, 8, 8, 2, 8, 8, 3)	Empty

⚠ ▯ indicates the mismatch in the expected output.



Even Numbered Tuples

Given an integer N, take N tuples as input from the user, each tuple provided on a separate line with elements separated by commas. Your task is to identify and extract the tuples from this list where all the integers are even.

Constraints:
1 <= N <= 10

Sample Test Case:
2 -----> Input N
2,4 ------> Enter the tuple 1 of integers.
22,33,44 ------> Enter the tuple 2 of integers.
[(2,4)] ----> Print the list of tuples in which all the integers of the tuple are even numbers.

Sample Test Cases

Test Case 1:

Expected Output:

```
2
2,4
22,33,44
[(2, 4)]
```

Test Case 2:

Expected Output:

```
5
1,6,10
5,8,22
4,6,11,24,36
```

tuplefunction2.py

```python
# write your code here

```

Execution Results

0 out of 4 shown cases successful
0 out of 3 hidden cases successful

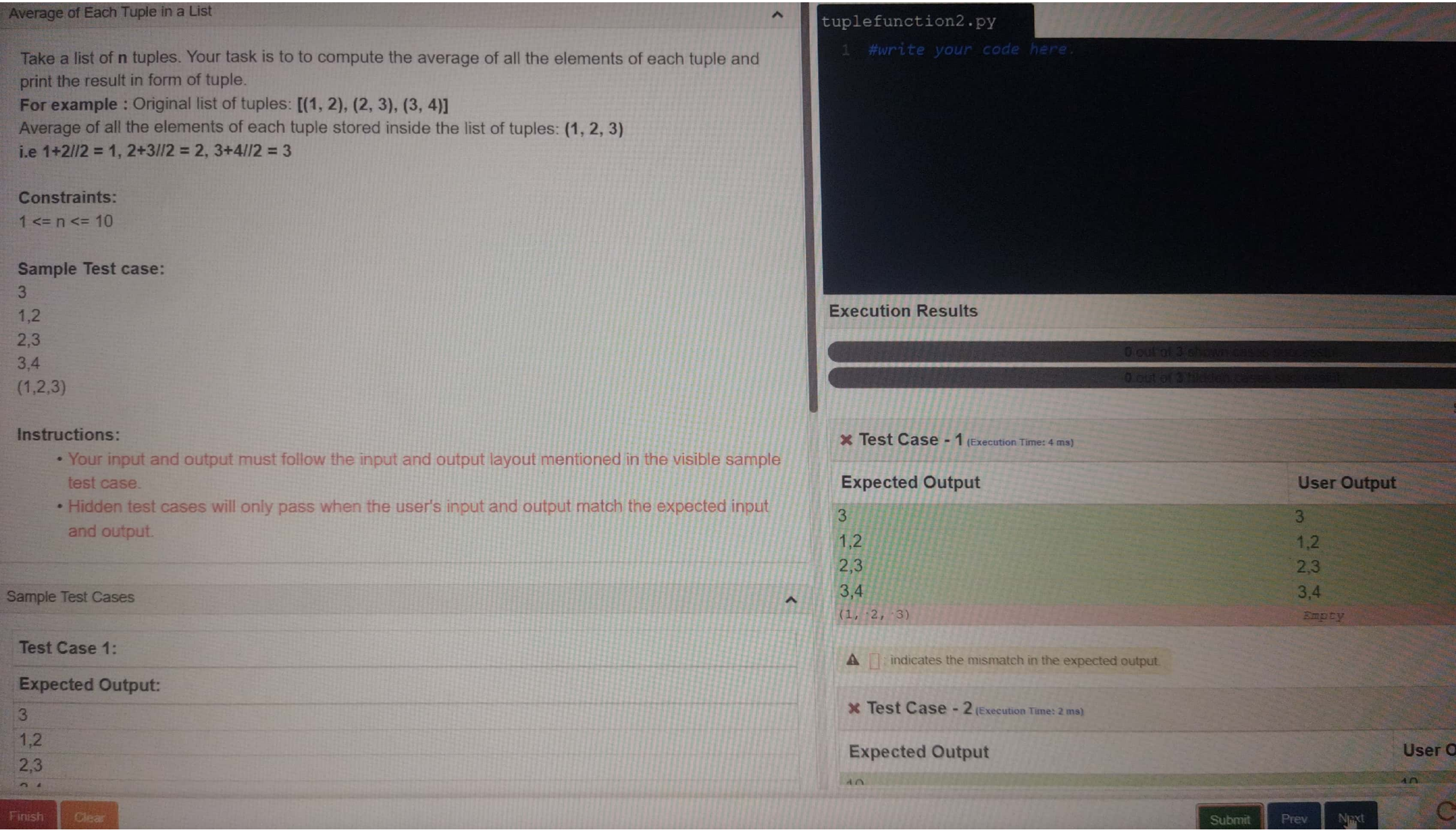
✖ Test Case - 1 (Execution Time: 3 ms)

Expected Output	User Output
2	2
2,4	2,4
22,33,44	22,33,44
[(2, 4)]	Empty

⚠ 🟥 : indicates the mismatch in the expected output.

✖ Test Case - 2 (Execution Time: 2 ms)

Expected Output	User Output
5	5



Average of Each Tuple in a List

Take a list of **n** tuples. Your task is to to compute the average of all the elements of each tuple and print the result in form of tuple.

For example : Original list of tuples: **[(1, 2), (2, 3), (3, 4)]**

Average of all the elements of each tuple stored inside the list of tuples: **(1, 2, 3)**

i.e 1+2//2 = 1, 2+3//2 = 2, 3+4//2 = 3

Constraints:

1 <= n <= 10

Sample Test case:

```
3
1,2
2,3
3,4
(1,2,3)
```

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 the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:

Expected Output:

```
3
1,2
2,3
```

tuplefunction2.py

```python
#write your code here.
```

Execution Results

0 out of 3 shown cases successful

0 out of 3 hidden cases successful

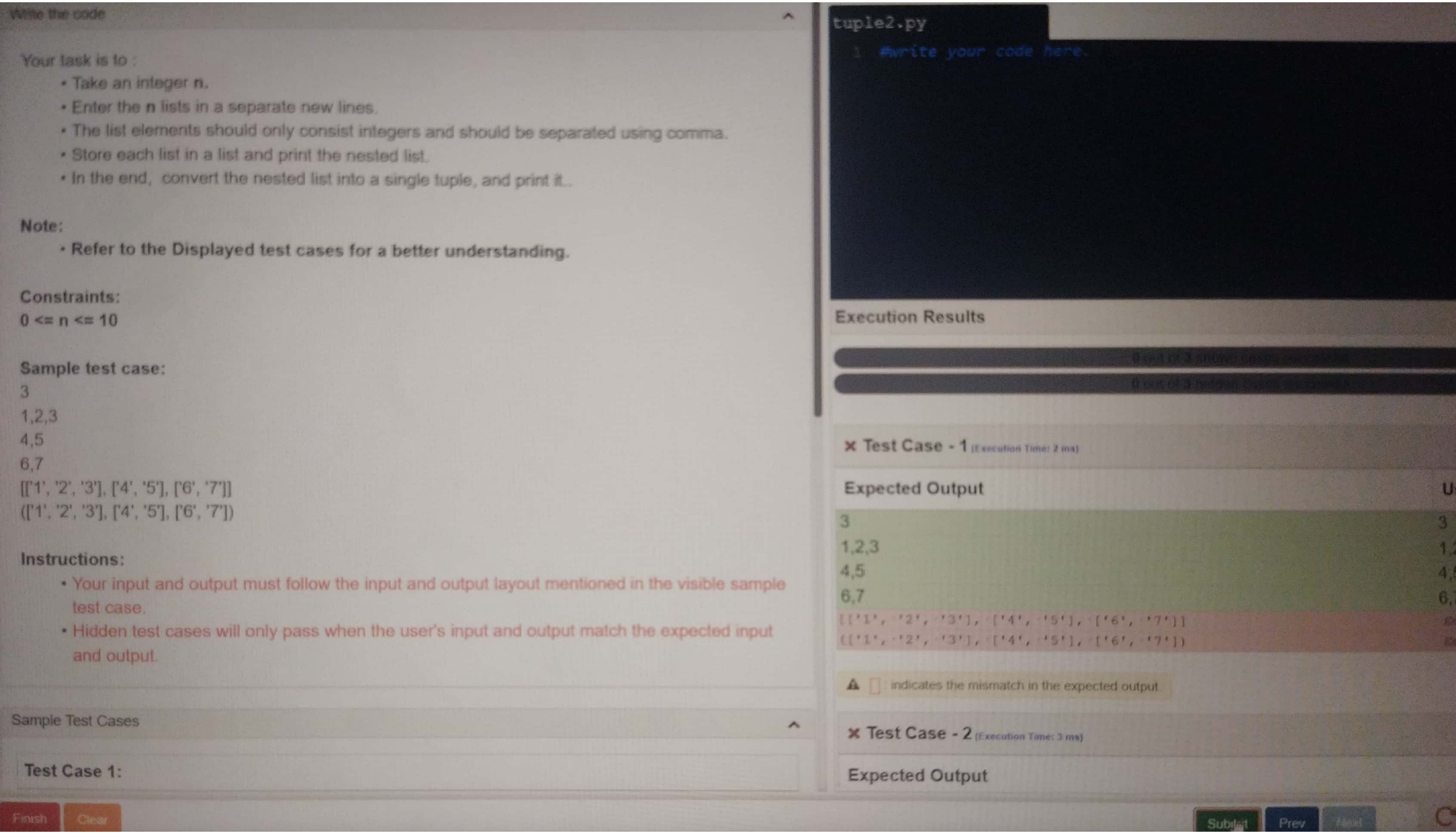
Test Case - 1 (Execution Time: 4 ms)

Expected Output	User Output
3	3
1,2	1,2
2,3	2,3
3,4	3,4
(1, 2, 3)	Empty

indicates the mismatch in the expected output.

Test Case - 2 (Execution Time: 2 ms)

Expected Output	User O


Write the code

Your task is to :
- Take an integer **n**.
- Enter the **n** lists in a separate new lines.
- The list elements should only consist integers and should be separated using comma.
- Store each list in a list and print the nested list.
- In the end, convert the nested list into a single tuple, and print it..

Note:
- **Refer to the Displayed test cases for a better understanding.**

Constraints:
0 <= n <= 10

Sample test case:
```
3
1,2,3
4,5
6,7
[['1', '2', '3'], ['4', '5'], ['6', '7']]
(['1', '2', '3'], ['4', '5'], ['6', '7'])
```

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 the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:

tuple2.py
```python
#write your code here.
```

Execution Results

0 out of 3 shown test cases passed
0 out of 3 hidden test cases passed

✗ Test Case - 1 (Execution Time: 2 ms)

Expected Output
```
3
1,2,3
4,5
6,7
[['1', '2', '3'], ['4', '5'], ['6', '7']]
(['1', '2', '3'], ['4', '5'], ['6', '7'])
```

U
```
3
1,
4,
6,
```

⚠ ▯ indicates the mismatch in the expected output

✗ Test Case - 2 (Execution Time: 3 ms)

Expected Output