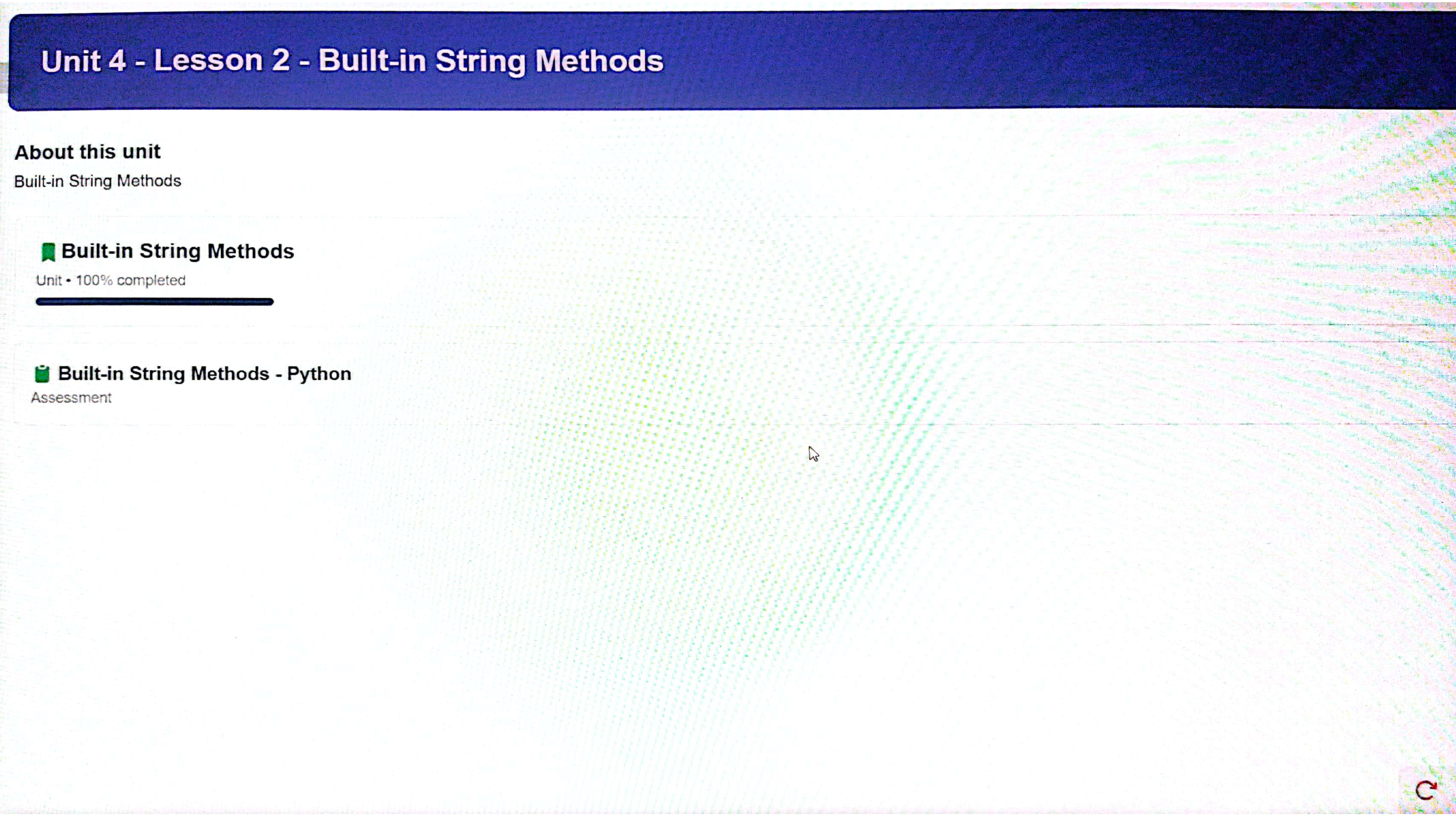


Unit 4 - Lesson 2 - Built-in String Methods

About this unit

Built-in String Methods

Built-in String Methods

Unit • 100% completed

Built-in String Methods - Python

Assessment

Built-in String Methods

About this unit

Built-in String Methods

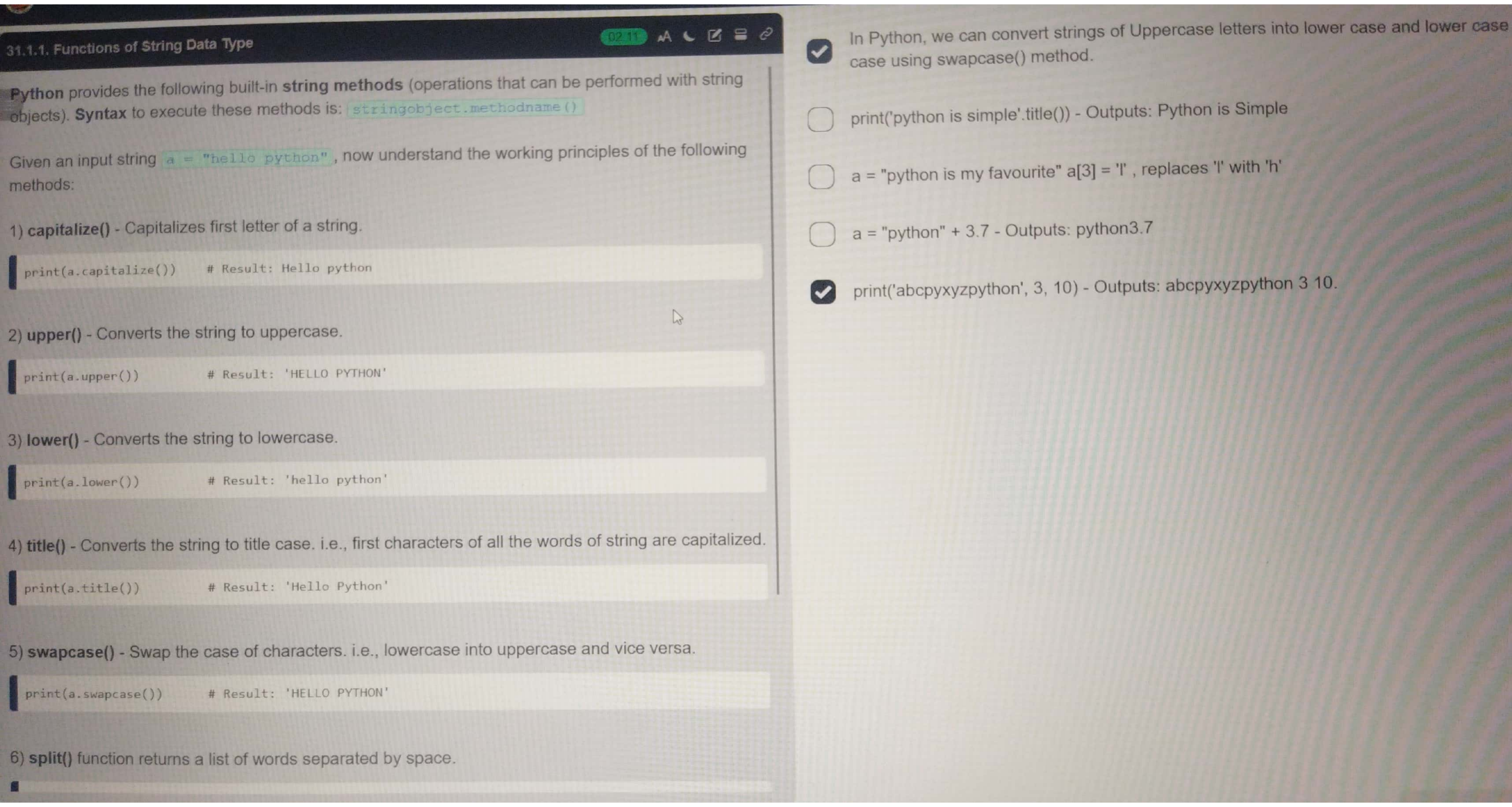
Functions of String Data Type
Question

Methods Of Strings - count(), replace(), join()
Question

Methods Of Strings - isupper(), islower(), isalpha(), isalnum()
Question

Methods of Strings - startswith(), endswith(), find(), len(), min(), max()
Question

Write a program to remove all digit in given string.
Question



31.1.1. Functions of String Data Type

Python provides the following built-in **string methods** (operations that can be performed with string objects). **Syntax** to execute these methods is: `stringobject.methodname()`

Given an input string `a = "hello python"`, now understand the working principles of the following methods:

1) **capitalize()** - Capitalizes first letter of a string.

```
print(a.capitalize())     # Result: Hello python
```

2) **upper()** - Converts the string to uppercase.

```
print(a.upper())          # Result: 'HELLO PYTHON'
```

3) **lower()** - Converts the string to lowercase.

```
print(a.lower())          # Result: 'hello python'
```

4) **title()** - Converts the string to title case. i.e., first characters of all the words of string are capitalized.

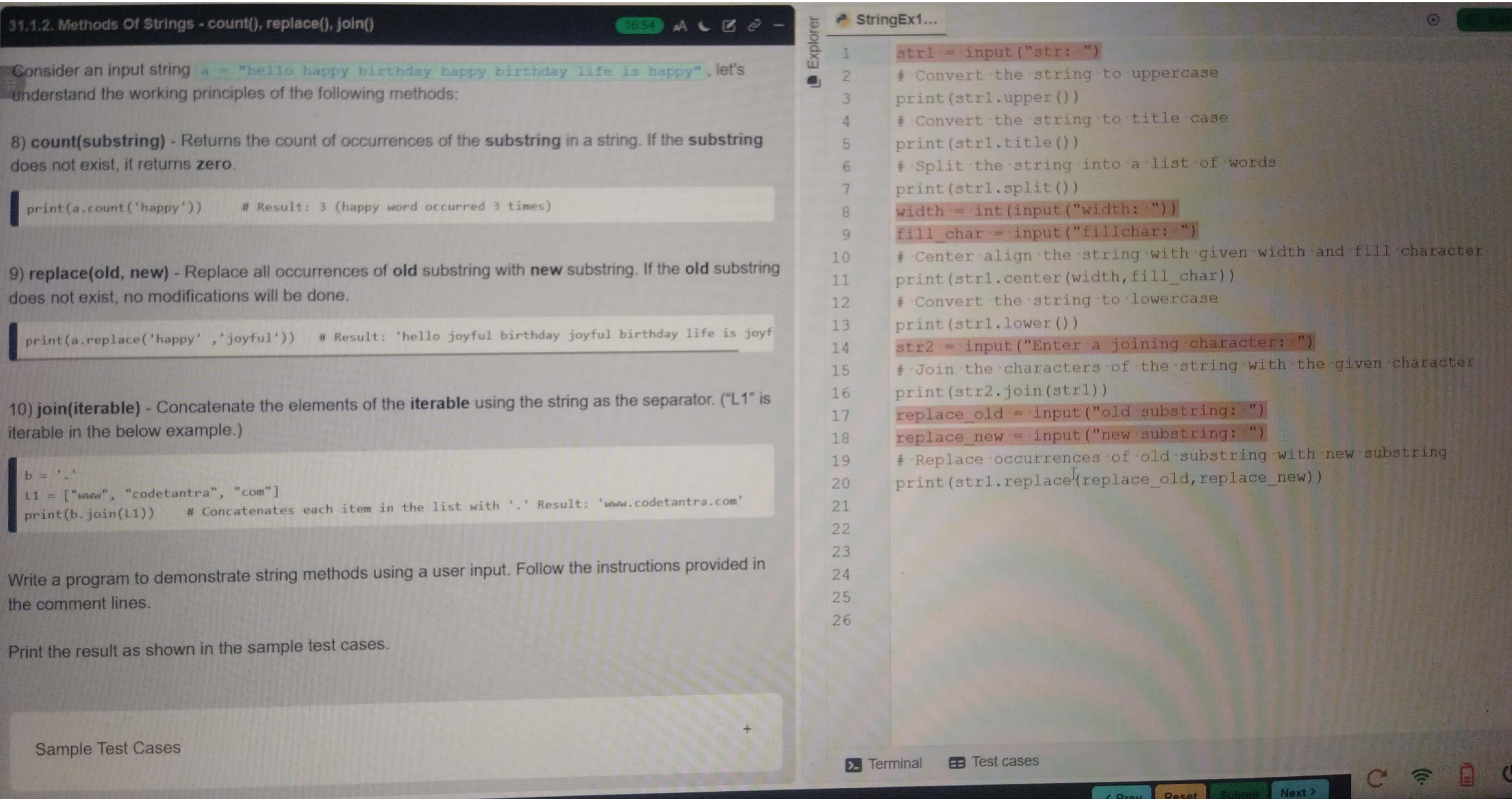
```
print(a.title())          # Result: 'Hello Python'
```

5) **swapcase()** - Swap the case of characters. i.e., lowercase into uppercase and vice versa.

```
print(a.swapcase())       # Result: 'HELLO PYTHON'
```

6) **split()** function returns a list of words separated by space.

- [✓] In Python, we can convert strings of Uppercase letters into lower case and lower case case using swapcase() method.
- [ ] print('python is simple'.title()) - Outputs: Python is Simple
- [ ] a = "python is my favourite" a[3] = 'l' , replaces 'l' with 'h'
- [ ] a = "python" + 3.7 - Outputs: python3.7
- [✓] print('abcpyxyzpython', 3, 10) - Outputs: abcpyxyzpython 3 10.



Consider an input string `a = "hello happy birthday happy birthday life is happy"`, let's understand the working principles of the following methods:

8) **count(substring)** - Returns the count of occurrences of the **substring** in a string. If the **substring** does not exist, it returns **zero**.

```
print(a.count('happy'))      # Result: 3 (happy word occurred 3 times)
```

9) **replace(old, new)** - Replace all occurrences of **old** substring with **new** substring. If the **old** substring does not exist, no modifications will be done.

```
print(a.replace('happy' ,'joyful'))   # Result: 'hello joyful birthday joyful birthday life is joyf
```

10) **join(iterable)** - Concatenate the elements of the **iterable** using the string as the separator. ("L1" is iterable in the below example.)

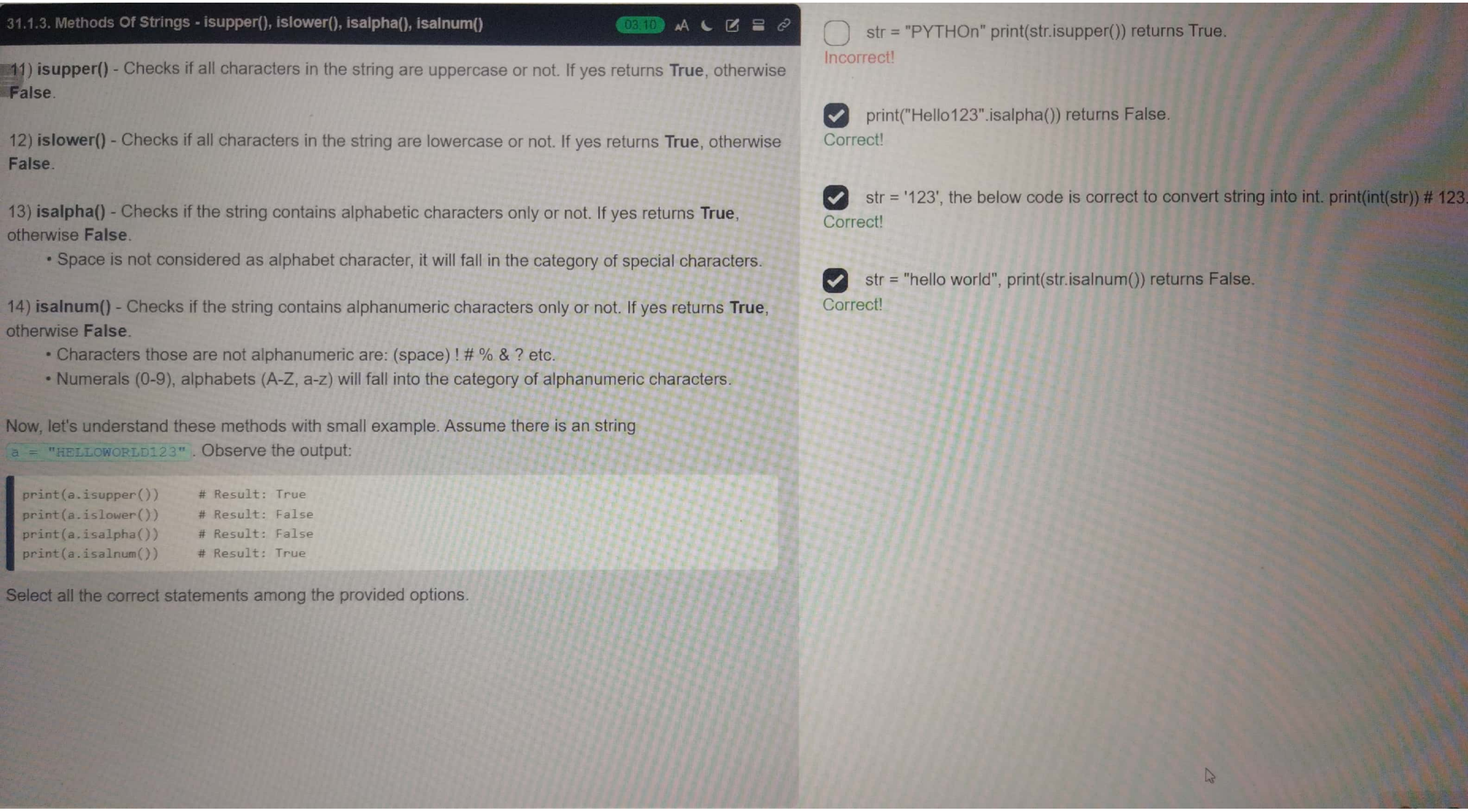
```
b = '.'
L1 = ["www", "codetantra", "com"]
print(b.join(L1))    # Concatenates each item in the list with '.' Result: 'www.codetantra.com'
```

Write a program to demonstrate string methods using a user input. Follow the instructions provided in the comment lines.

Print the result as shown in the sample test cases.

Sample Test Cases

```
str1 = input("str: ")
# Convert the string to uppercase
print(str1.upper())
# Convert the string to title case
print(str1.title())
# Split the string into a list of words
print(str1.split())
width = int(input("width: "))
fill_char = input("fillchar: ")
# Center align the string with given width and fill character
print(str1.center(width,fill_char))
# Convert the string to lowercase
print(str1.lower())
str2 = input("Enter a joining character: ")
# Join the characters of the string with the given character
print(str2.join(str1))
replace_old = input("old substring: ")
replace_new = input("new substring: ")
# Replace occurrences of old substring with new substring
print(str1.replace(replace_old,replace_new))
```



31.1.3. Methods Of Strings - isupper(), islower(), isalpha(), isalnum()

11) **isupper()** - Checks if all characters in the string are uppercase or not. If yes returns **True**, otherwise **False**.

12) **islower()** - Checks if all characters in the string are lowercase or not. If yes returns **True**, otherwise **False**.

13) **isalpha()** - Checks if the string contains alphabetic characters only or not. If yes returns **True**, otherwise **False**.

- Space is not considered as alphabet character, it will fall in the category of special characters.

14) **isalnum()** - Checks if the string contains alphanumeric characters only or not. If yes returns **True**, otherwise **False**.

- Characters those are not alphanumeric are: (space) ! # % & ? etc.
- Numerals (0-9), alphabets (A-Z, a-z) will fall into the category of alphanumeric characters.

Now, let's understand these methods with small example. Assume there is an string `a = "HELLOWORLD123"`. Observe the output:

```
print(a.isupper())     # Result: True
print(a.islower())     # Result: False
print(a.isalpha())     # Result: False
print(a.isalnum())     # Result: True
```

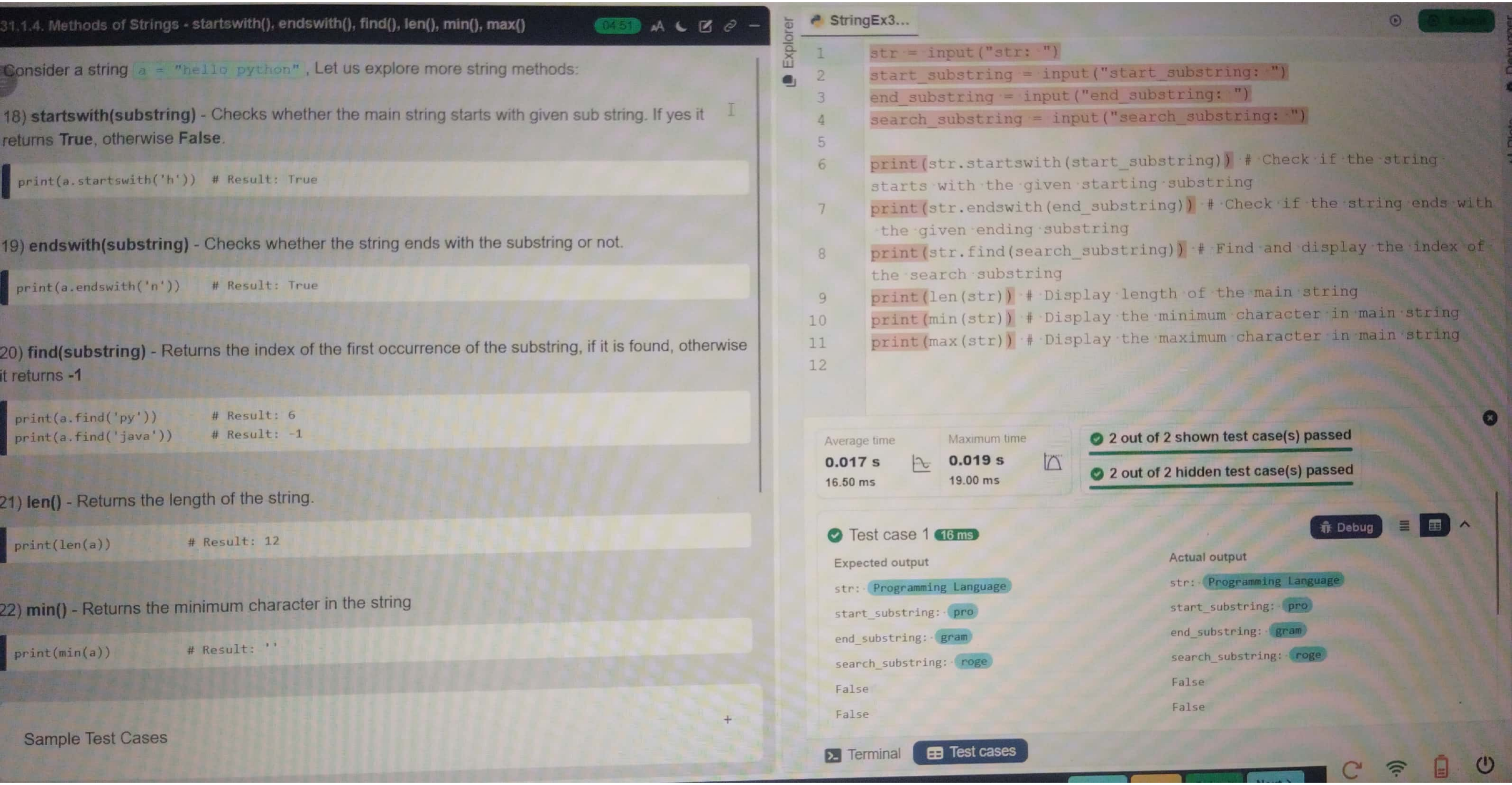
Select all the correct statements among the provided options.

str = "PYTHOn" print(str.isupper()) returns True.
Incorrect!

[✓ print("Hello123".isalpha()) returns False.]
Correct!

[✓ str = '123', the below code is correct to convert string into int. print(int(str)) # 123.]
Correct!

[✓ str = "hello world", print(str.isalnum()) returns False.]
Correct!


31.1.4. Methods of Strings - startswith(), endswith(), find(), len(), min(), max()

Consider a string `a = "hello python"` , Let us explore more string methods:

18) **startswith(substring)** - Checks whether the main string starts with given sub string. If yes it returns **True**, otherwise **False**.

```
print(a.startswith('h'))  # Result: True
```

19) **endswith(substring)** - Checks whether the string ends with the substring or not.

```
print(a.endswith('n'))    # Result: True
```

20) **find(substring)** - Returns the index of the first occurrence of the substring, if it is found, otherwise it returns **-1**

```
print(a.find('py'))       # Result: 6
print(a.find('java'))     # Result: -1
```

21) **len()** - Returns the length of the string.

```
print(len(a))           # Result: 12
```

22) **min()** - Returns the minimum character in the string

```
print(min(a))           # Result: ''
```

Sample Test Cases

StringEx3...

```
str = input("str: ")
start_substring = input("start_substring: ")
end_substring = input("end_substring: ")
search_substring = input("search_substring: ")

print(str.startswith(start_substring)) # Check if the string starts with the given starting substring
print(str.endswith(end_substring)) # Check if the string ends with the given ending substring
print(str.find(search_substring)) # Find and display the index of the search substring
print(len(str)) # Display length of the main string
print(min(str)) # Display the minimum character in main string
print(max(str)) # Display the maximum character in main string
```

Average time 0.017 s 16.50 ms — Maximum time 0.019 s 19.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

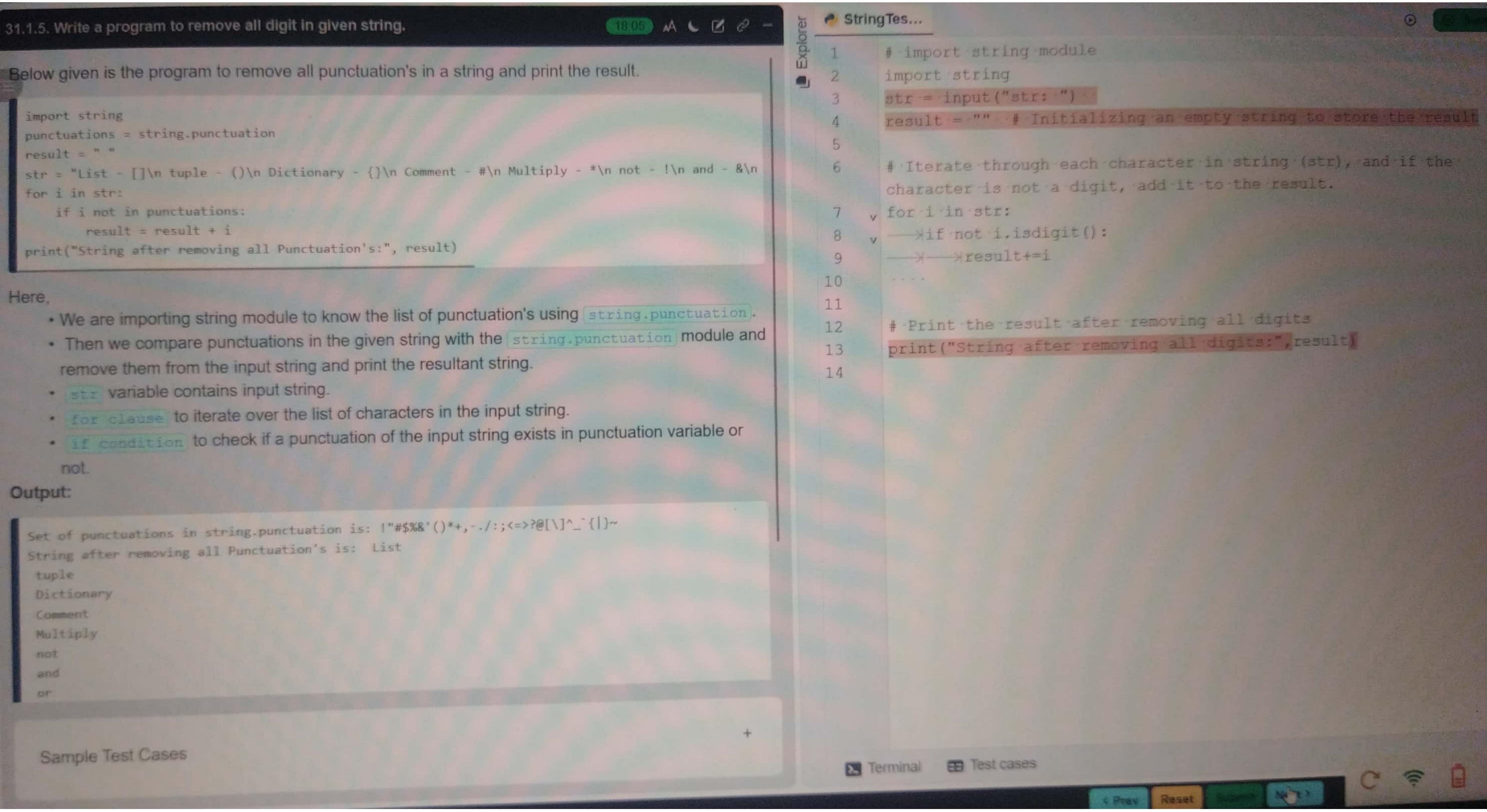
Test case 1 16 ms

Expected output

```
str: Programming Language
start_substring: pro
end_substring: gram
search_substring: roge
False
False
```

Actual output

```
str: Programming Language
start_substring: pro
end_substring: gram
search_substring: roge
False
False
```


31.1.5. Write a program to remove all digit in given string.

Below given is the program to remove all punctuation's in a string and print the result.

```
import string
punctuations = string.punctuation
result = " "
str = "List - []\n tuple - ()\n Dictionary - {}\n Comment - #\n Multiply - *\n not - !\n and - &\n
for i in str:
    if i not in punctuations:
        result = result + i
print("String after removing all Punctuation's:", result)
```

Here,

- We are importing string module to know the list of punctuation's using `string.punctuation`.
- Then we compare punctuations in the given string with the `string.punctuation` module and remove them from the input string and print the resultant string.
- `str` variable contains input string.
- `for clause` to iterate over the list of characters in the input string.
- `if condition` to check if a punctuation of the input string exists in punctuation variable or not.

Output:

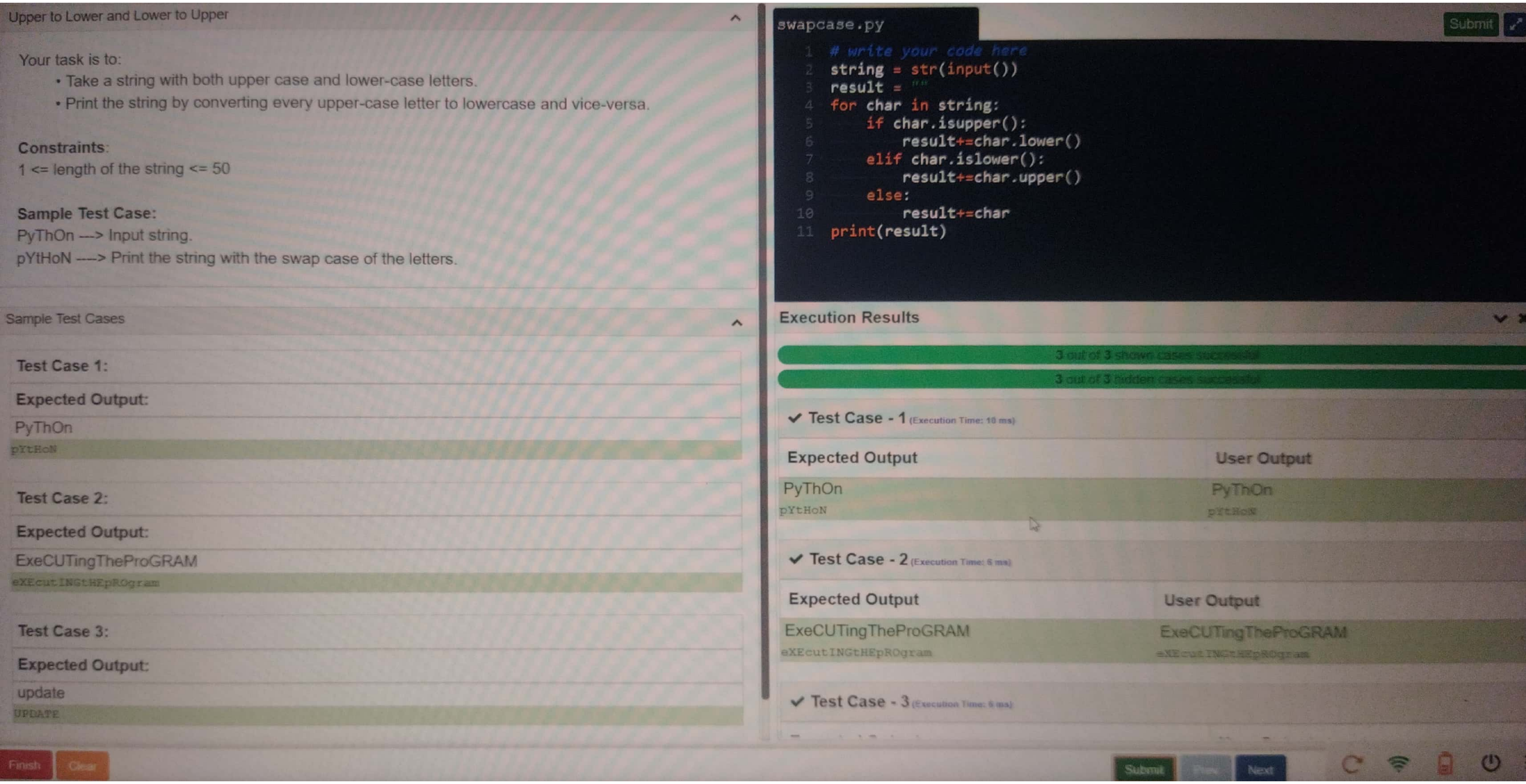
```
Set of punctuations in string.punctuation is: !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
String after removing all Punctuation's is:  List
 tuple
 Dictionary
 Comment
 Multiply
 not
 and
 or
```

Sample Test Cases

```
# import string module
import string
str = input("str: ")
result = ""  # Initializing an empty string to store the result

# Iterate through each character in string (str), and if the character is not a digit, add it to the result.
for i in str:
	if not i.isdigit():
		result+=i


# Print the result after removing all digits
print("String after removing all digits:",result)
```



Upper to Lower and Lower to Upper

Your task is to:

- Take a string with both upper case and lower-case letters.
- Print the string by converting every upper-case letter to lowercase and vice-versa.

Constraints:
1 <= length of the string <= 50

Sample Test Case:
PyThOn ---> Input string.
pYtHoN ---> Print the string with the swap case of the letters.

Sample Test Cases

Test Case 1:

Expected Output:

```
PyThOn
pYtHoN
```

Test Case 2:

Expected Output:

```
ExeCUTingTheProGRAM
eXEcutINGtHEpROgram
```

Test Case 3:

Expected Output:

```
update
UPDATE
```

swapcase.py

```python
# write your code here
string = str(input())
result = ""
for char in string:
    if char.isupper():
        result+=char.lower()
    elif char.islower():
        result+=char.upper()
    else:
        result+=char
print(result)
```

Execution Results

3 out of 3 shown cases successful

3 out of 3 hidden cases successful

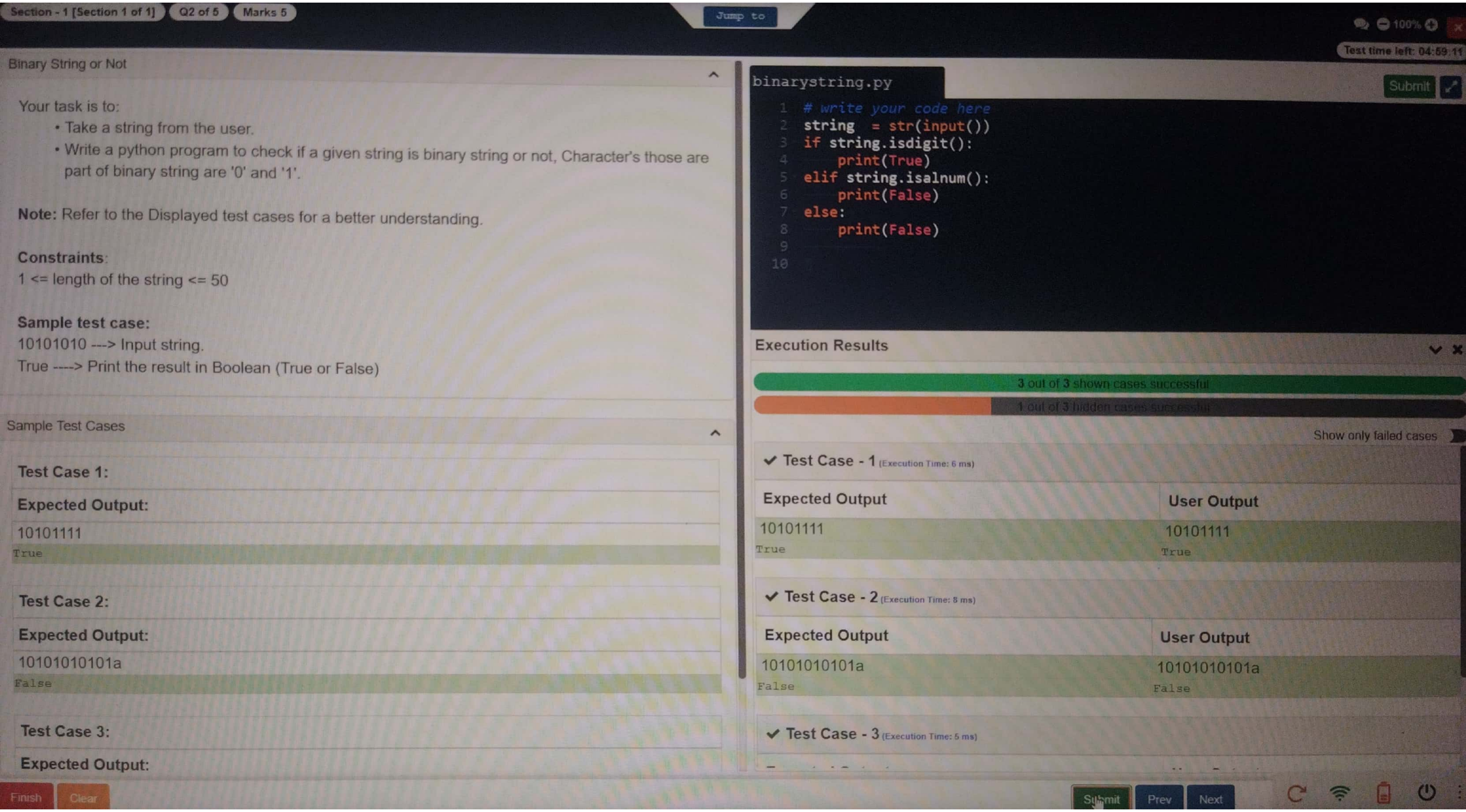
✓ Test Case - 1 (Execution Time: 10 ms)

Expected Output	User Output
PyThOn pYtHoN	PyThOn pYtHoN

✓ Test Case - 2 (Execution Time: 6 ms)

Expected Output	User Output
ExeCUTingTheProGRAM eXEcutINGtHEpROgram	ExeCUTingTheProGRAM eXEcutINGtHEpROgram

✓ Test Case - 3 (Execution Time: 6 ms)



Binary String or Not

Your task is to:

- Take a string from the user.
- Write a python program to check if a given string is binary string or not, Character's those are part of binary string are '0' and '1'.

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

Constraints:
1 <= length of the string <= 50

Sample test case:
10101010 ---> Input string.
True ----> Print the result in Boolean (True or False)

Sample Test Cases

Test Case 1:

Expected Output:

```
10101111
True
```

Test Case 2:

Expected Output:

```
10101010101a
False
```

Test Case 3:

Expected Output:

binarystring.py

```python
# write your code here
string  = str(input())
if string.isdigit():
    print(True)
elif string.isalnum():
    print(False)
else:
    print(False)
```

Execution Results

3 out of 3 shown cases successful

1 out of 3 hidden cases successful

Show only failed cases

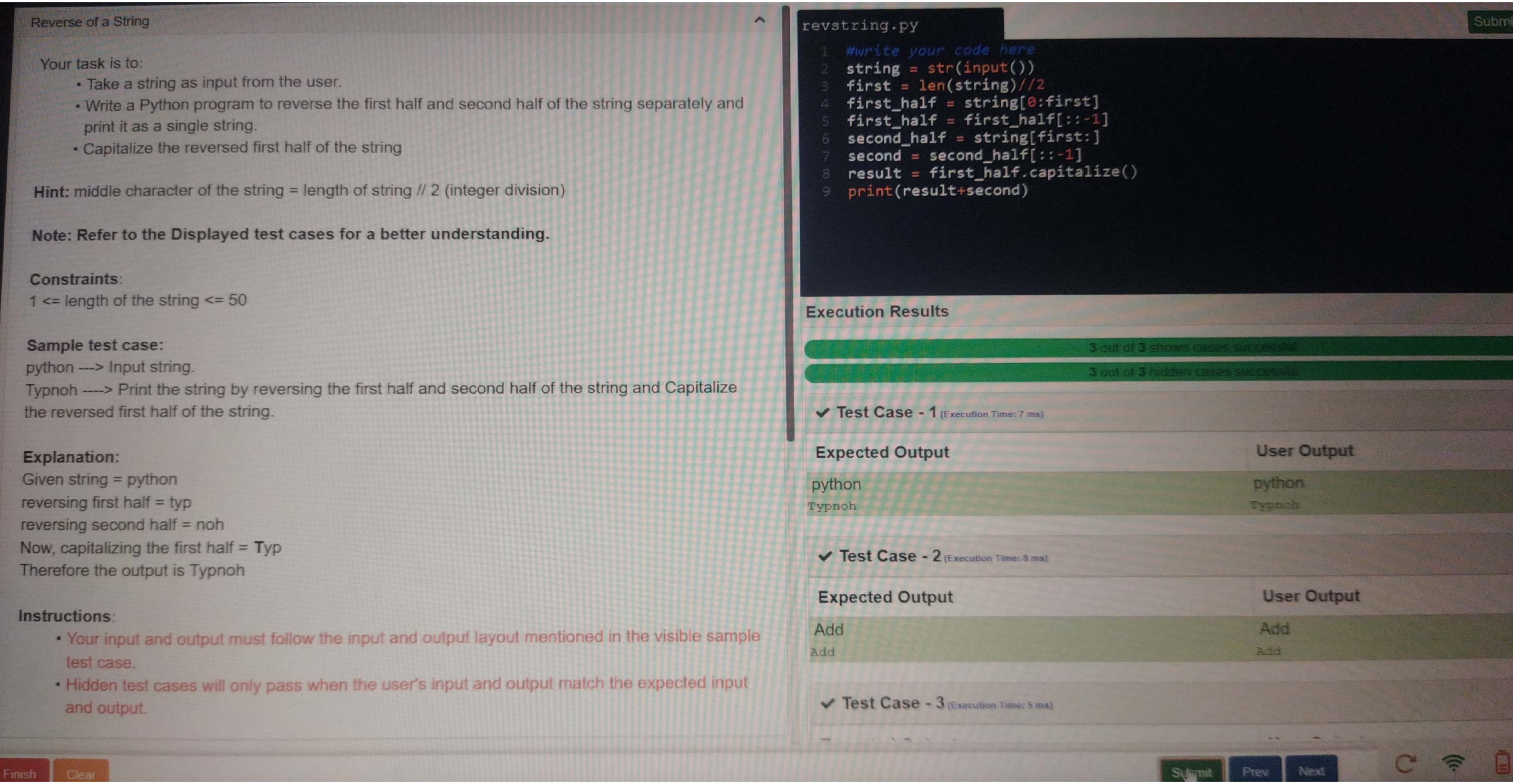
✓ Test Case - 1 (Execution Time: 6 ms)

Expected Output	User Output
10101111 True	10101111 True

✓ Test Case - 2 (Execution Time: 8 ms)

Expected Output	User Output
10101010101a False	10101010101a False

✓ Test Case - 3 (Execution Time: 5 ms)



Your task is to:

- Take a string as input from the user.
- Write a Python program to reverse the first half and second half of the string separately and print it as a single string.
- Capitalize the reversed first half of the string

Hint: middle character of the string = length of string // 2 (integer division)

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

Constraints:
1 <= length of the string <= 50

Sample test case:
python ---> Input string.
Typnoh ----> Print the string by reversing the first half and second half of the string and Capitalize the reversed first half of the string.

Explanation:
Given string = python
reversing first half = typ
reversing second half = noh
Now, capitalizing the first half = **T**yp
Therefore the output is Typnoh

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.

revstring.py

```
#write your code here
string = str(input())
first = len(string)//2
first_half = string[0:first]
first_half = first_half[::-1]
second_half = string[first:]
second = second_half[::-1]
result = first_half.capitalize()
print(result+second)
```

Execution Results

3 out of 3 shown cases successful
3 out of 3 hidden cases successful

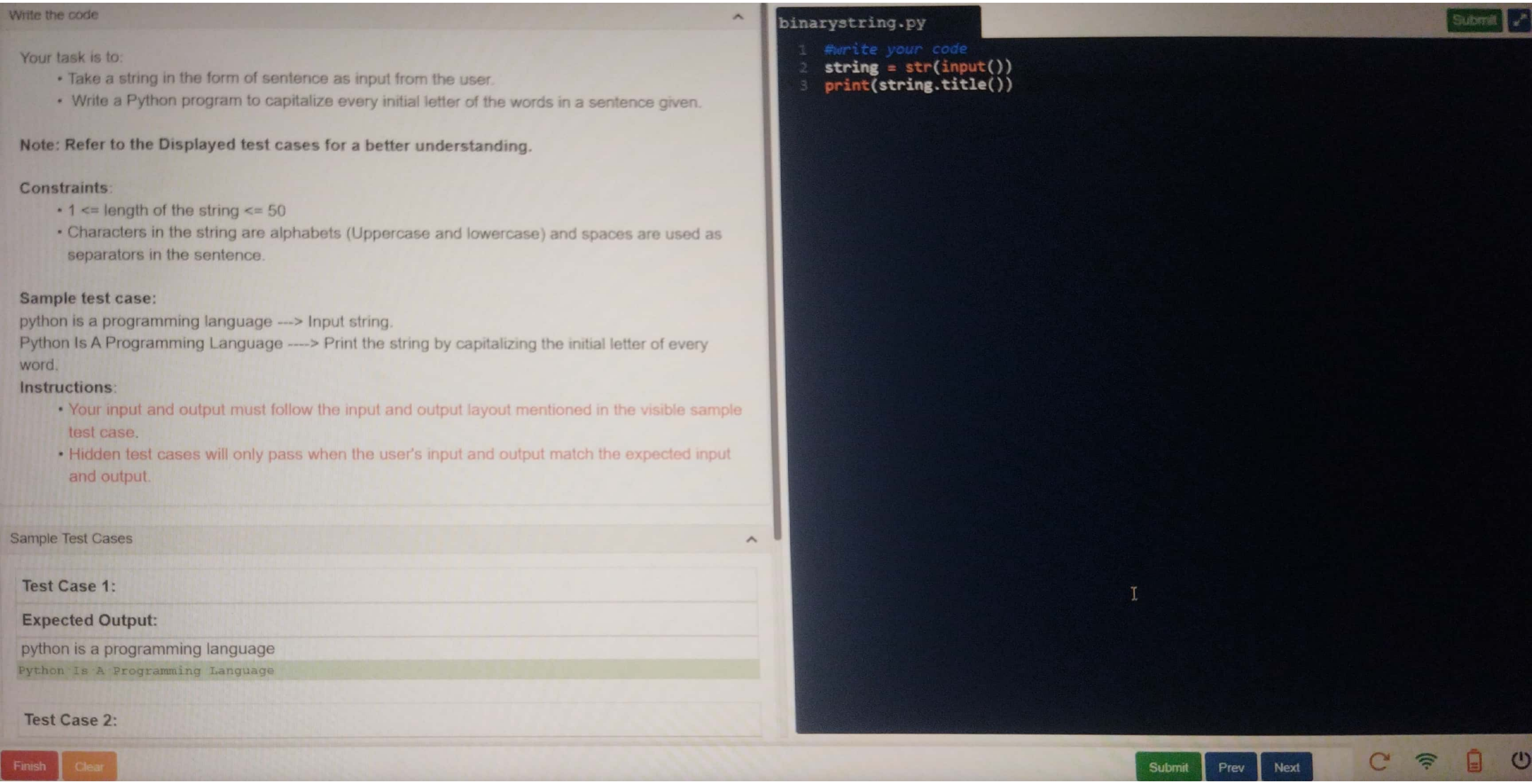
✓ Test Case - 1 (Execution Time: 7 ms)

Expected Output	User Output
python Typnoh	python Typnoh

✓ Test Case - 2 (Execution Time: 8 ms)

Expected Output	User Output
Add Add	Add Add

✓ Test Case - 3 (Execution Time: 5 ms)



Write the code

Your task is to:

- Take a string in the form of sentence as input from the user.
- Write a Python program to capitalize every initial letter of the words in a sentence given.

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

Constraints:

- 1 <= length of the string <= 50
- Characters in the string are alphabets (Uppercase and lowercase) and spaces are used as separators in the sentence.

Sample test case:

python is a programming language ---> Input string.

Python Is A Programming Language ----> Print the string by capitalizing the initial letter of every word.

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:

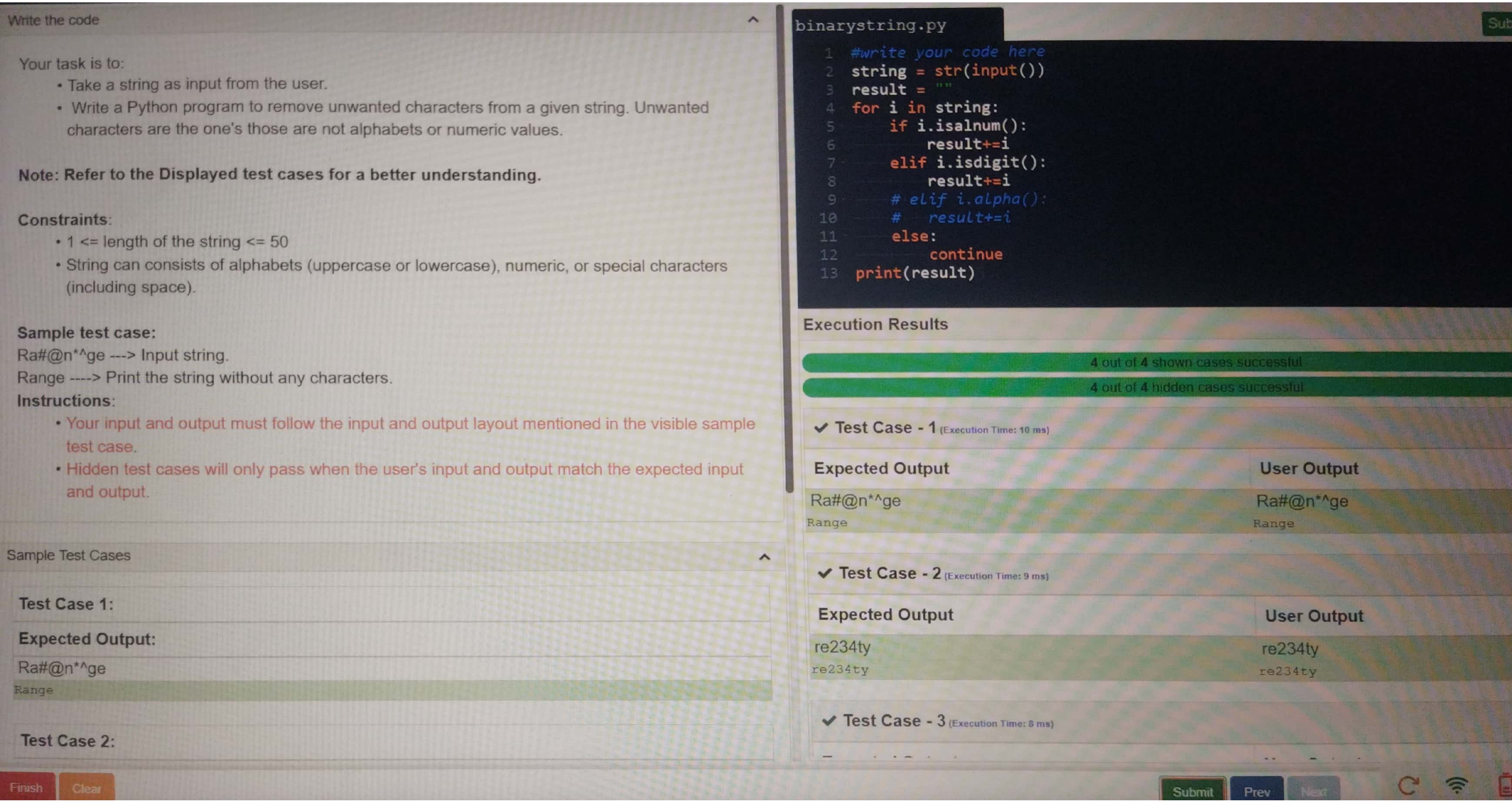
python is a programming language

```
Python Is A Programming Language
```

Test Case 2:

binarystring.py

```python
#write your code
string = str(input())
print(string.title())
```



Write the code

Your task is to:
- Take a string as input from the user.
- Write a Python program to remove unwanted characters from a given string. Unwanted characters are the one's those are not alphabets or numeric values.

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

Constraints:
- 1 <= length of the string <= 50
- String can consists of alphabets (uppercase or lowercase), numeric, or special characters (including space).

Sample test case:
Ra#@n*^ge ---> Input string.
Range ----> Print the string without any characters.
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:

```
Ra#@n*^ge
Range
```

Test Case 2:

binarystring.py

```python
#write your code here
string = str(input())
result = ""
for i in string:
    if i.isalnum():
        result+=i
    elif i.isdigit():
        result+=i
    # elif i.alpha():
    #    result+=i
    else:
        continue
print(result)
```

Execution Results

4 out of 4 shown cases successful

4 out of 4 hidden cases successful

✓ Test Case - 1 (Execution Time: 10 ms)

Expected Output	User Output
Ra#@n*^ge Range	Ra#@n*^ge Range

✓ Test Case - 2 (Execution Time: 9 ms)

Expected Output	User Output
re234ty re234ty	re234ty re234ty

✓ Test Case - 3 (Execution Time: 8 ms)