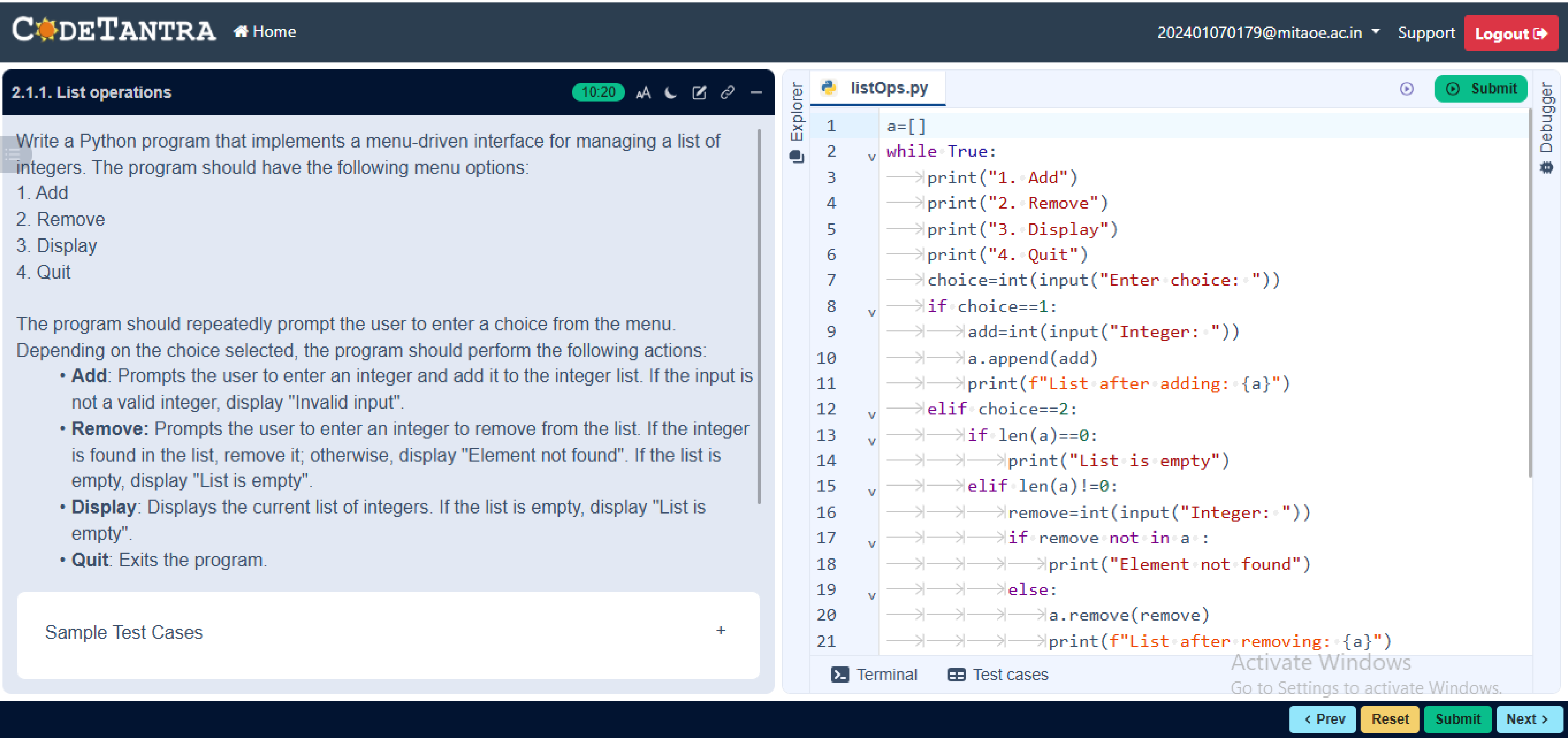


2.1.1. List operations

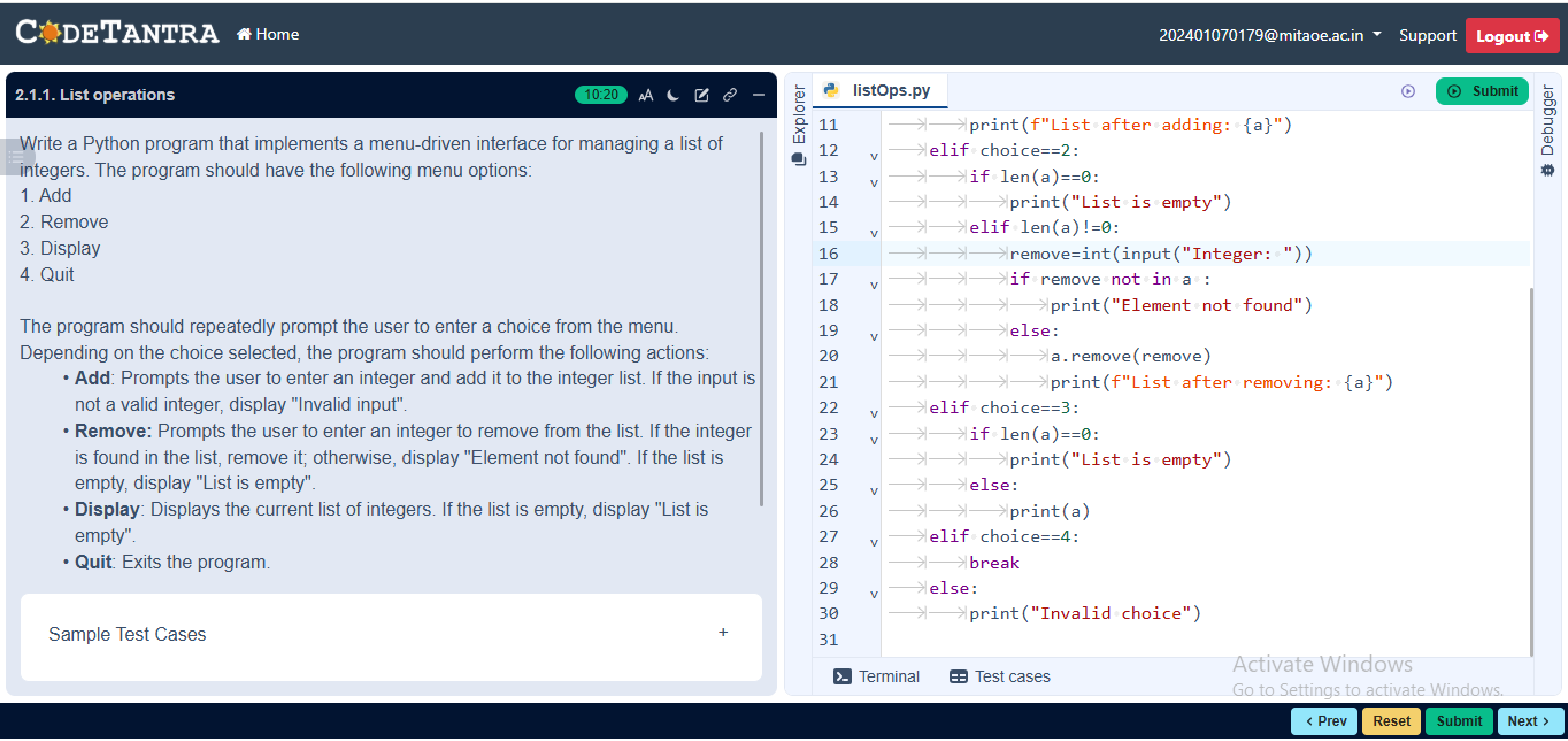
Write a Python program that implements a menu-driven interface for managing a list of integers. The program should have the following menu options:
1. Add
2. Remove
3. Display
4. Quit

The program should repeatedly prompt the user to enter a choice from the menu. Depending on the choice selected, the program should perform the following actions:

- **Add**: Prompts the user to enter an integer and add it to the integer list. If the input is not a valid integer, display "Invalid input".
- **Remove:** Prompts the user to enter an integer to remove from the list. If the integer is found in the list, remove it; otherwise, display "Element not found". If the list is empty, display "List is empty".
- **Display**: Displays the current list of integers. If the list is empty, display "List is empty".
- **Quit**: Exits the program.

Sample Test Cases +

```python
a=[]
while True:
	print("1. Add")
	print("2. Remove")
	print("3. Display")
	print("4. Quit")
	choice=int(input("Enter choice: "))
	if choice==1:
		add=int(input("Integer: "))
		a.append(add)
		print(f"List after adding: {a}")
	elif choice==2:
		if len(a)==0:
			print("List is empty")
		elif len(a)!=0:
			remove=int(input("Integer: "))
			if remove not in a :
				print("Element not found")
			else:
				a.remove(remove)
				print(f"List after removing: {a}")
```



2.1.1. List operations

Write a Python program that implements a menu-driven interface for managing a list of integers. The program should have the following menu options:

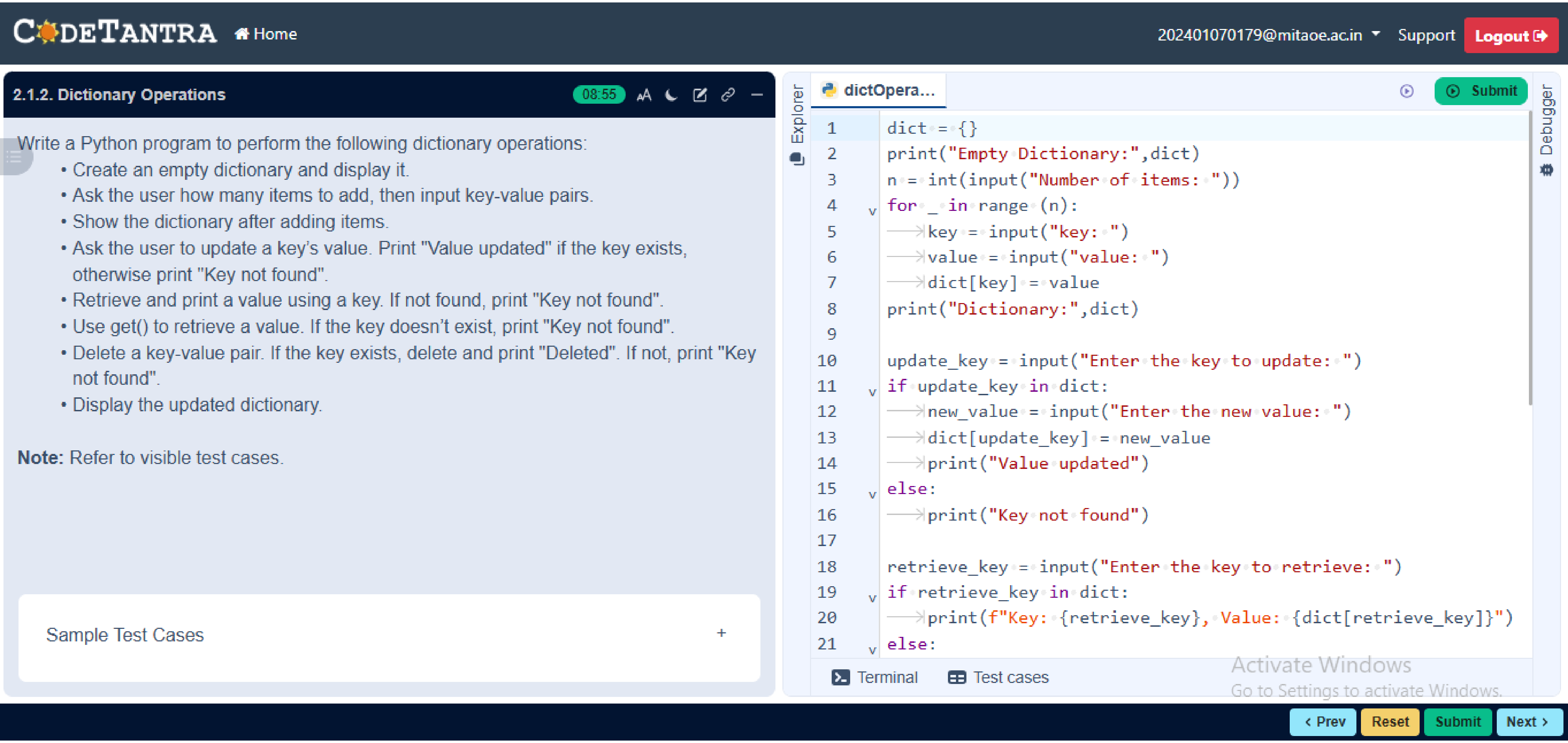
1. Add
2. Remove
3. Display
4. Quit

The program should repeatedly prompt the user to enter a choice from the menu. Depending on the choice selected, the program should perform the following actions:

- **Add**: Prompts the user to enter an integer and add it to the integer list. If the input is not a valid integer, display "Invalid input".
- **Remove:** Prompts the user to enter an integer to remove from the list. If the integer is found in the list, remove it; otherwise, display "Element not found". If the list is empty, display "List is empty".
- **Display**: Displays the current list of integers. If the list is empty, display "List is empty".
- **Quit**: Exits the program.

Sample Test Cases +

```python
		print(f"List after adding: {a}")
	elif choice==2:
		if len(a)==0:
			print("List is empty")
		elif len(a)!=0:
			remove=int(input("Integer: "))
			if remove not in a :
				print("Element not found")
			else:
				a.remove(remove)
				print(f"List after removing: {a}")
	elif choice==3:
		if len(a)==0:
			print("List is empty")
		else:
			print(a)
	elif choice==4:
		break
	else:
		print("Invalid choice")
```



2.1.2. Dictionary Operations

Write a Python program to perform the following dictionary operations:

- Create an empty dictionary and display it.
- Ask the user how many items to add, then input key-value pairs.
- Show the dictionary after adding items.
- Ask the user to update a key's value. Print "Value updated" if the key exists, otherwise print "Key not found".
- Retrieve and print a value using a key. If not found, print "Key not found".
- Use get() to retrieve a value. If the key doesn't exist, print "Key not found".
- Delete a key-value pair. If the key exists, delete and print "Deleted". If not, print "Key not found".
- Display the updated dictionary.

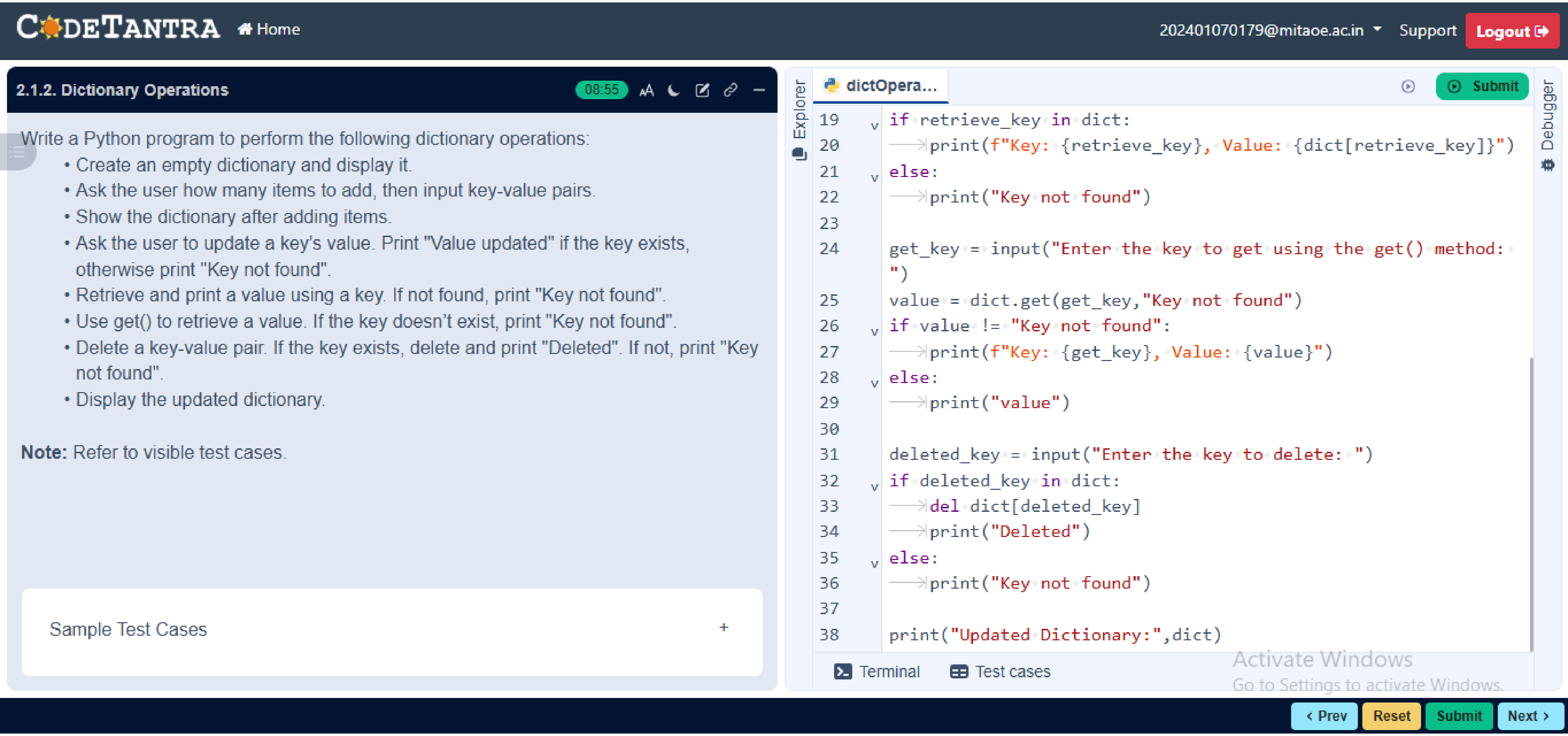
Note: Refer to visible test cases.

Sample Test Cases +

```python
dict = {}
print("Empty Dictionary:",dict)
n = int(input("Number of items: "))
for _ in range (n):
	key = input("key: ")
	value = input("value: ")
	dict[key] = value
print("Dictionary:",dict)

update_key = input("Enter the key to update: ")
if update_key in dict:
	new_value = input("Enter the new value: ")
	dict[update_key] = new_value
	print("Value updated")
else:
	print("Key not found")

retrieve_key = input("Enter the key to retrieve: ")
if retrieve_key in dict:
	print(f"Key: {retrieve_key}, Value: {dict[retrieve_key]}")
else:
```



2.1.2. Dictionary Operations

Write a Python program to perform the following dictionary operations:

- Create an empty dictionary and display it.
- Ask the user how many items to add, then input key-value pairs.
- Show the dictionary after adding items.
- Ask the user to update a key's value. Print "Value updated" if the key exists, otherwise print "Key not found".
- Retrieve and print a value using a key. If not found, print "Key not found".
- Use get() to retrieve a value. If the key doesn't exist, print "Key not found".
- Delete a key-value pair. If the key exists, delete and print "Deleted". If not, print "Key not found".
- Display the updated dictionary.

Note: Refer to visible test cases.

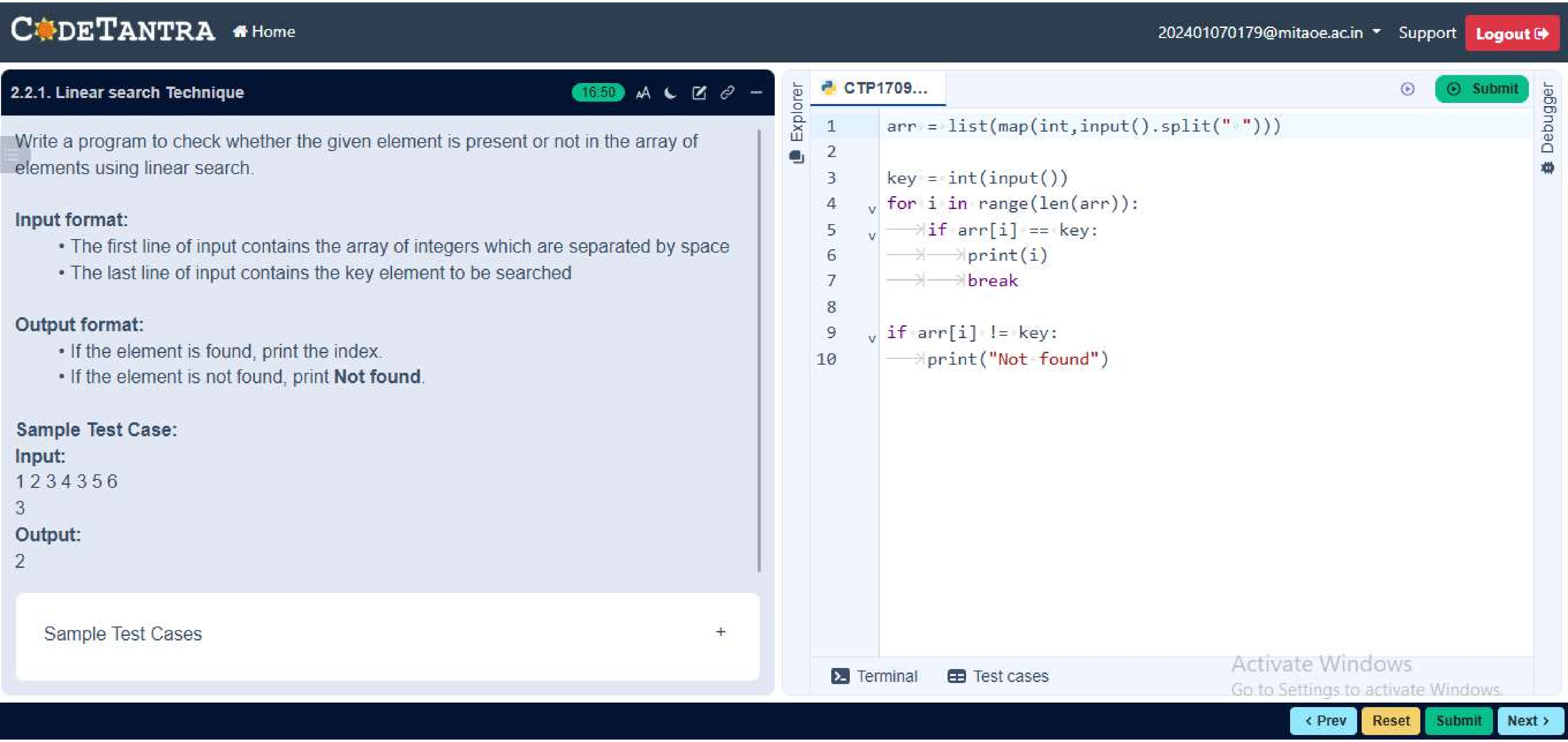
Sample Test Cases +

```python
if retrieve_key in dict:
	print(f"Key: {retrieve_key}, Value: {dict[retrieve_key]}")
else:
	print("Key not found")

get_key = input("Enter the key to get using the get() method: ")
value = dict.get(get_key,"Key not found")
if value != "Key not found":
	print(f"Key: {get_key}, Value: {value}")
else:
	print("value")

deleted_key = input("Enter the key to delete: ")
if deleted_key in dict:
	del dict[deleted_key]
	print("Deleted")
else:
	print("Key not found")

print("Updated Dictionary:",dict)
```



2.2.1. Linear search Technique

Write a program to check whether the given element is present or not in the array of elements using linear search.

Input format:

- The first line of input contains the array of integers which are separated by space
- The last line of input contains the key element to be searched

Output format:

- If the element is found, print the index.
- If the element is not found, print **Not found**.

Sample Test Case:

Input:

1 2 3 4 3 5 6

3

Output:

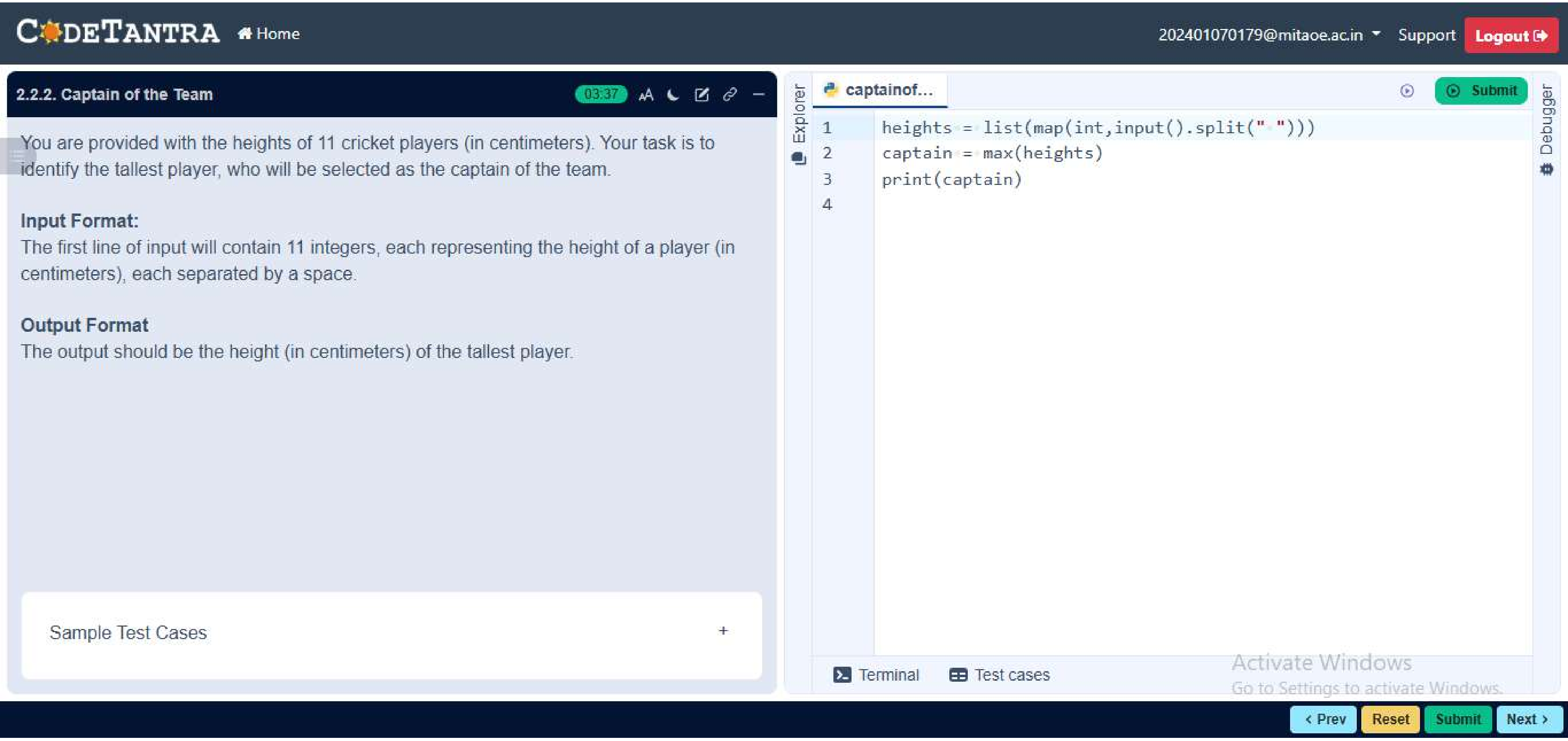
2

Sample Test Cases +

```python
arr = list(map(int,input().split(" ")))

key = int(input())
for i in range(len(arr)):
	if arr[i] == key:
		print(i)
		break

if arr[i] != key:
	print("Not found")
```



2.2.2. Captain of the Team

You are provided with the heights of 11 cricket players (in centimeters). Your task is to identify the tallest player, who will be selected as the captain of the team.

Input Format:
The first line of input will contain 11 integers, each representing the height of a player (in centimeters), each separated by a space.

Output Format
The output should be the height (in centimeters) of the tallest player.

Sample Test Cases

```python
heights = list(map(int,input().split(" ")))
captain = max(heights)
print(captain)

```