Sample Test Cases

# 2.1.1. List operations ALBR-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.
- . The program should handle invalid menu choices by displaying "Invalid choice". Ensure that the program continues to prompt the user until they choose to guit (option 4).

Explorer a=[ ] while True: print("1. Add") print("2. Remove") print("3. Display") print("4. Quit") n=int(input("Enter choice: ")) Mif n==1: Hadd=int(input("Integer: ")) 10 a.append(add) 11 print(f"List after adding: {a}") 12 elif n==2: if len(a)==0: 14 >> > print("List is empty") elif len(a)!=0: remove=int(input("Integer: ")) if remove not in a : 17 print("Element not found") 19 - Helse: a.remove(remove) 20 print(f"List after removing: {a}") elif n==3: >>> >> >print("List is empty") else:

listOps.py

+

Logout (\*)

### 2.1.1. List operations



> Terminal

Test cases

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

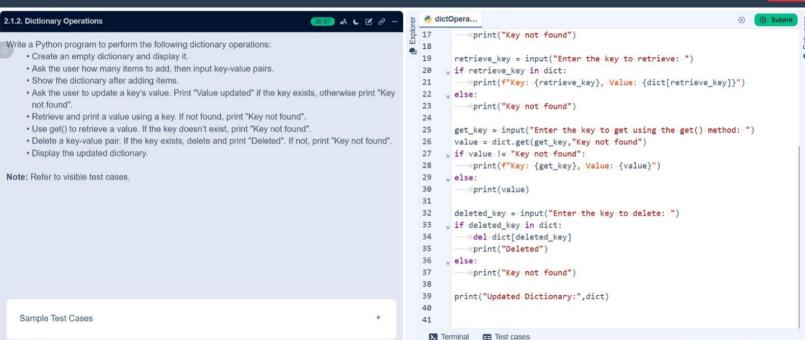
- 1. Add
- 2 Remove
- 3. Display 4. Quit

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.
- . The program should handle invalid menu choices by displaying "Invalid choice". Ensure that the program continues to prompt the user until they choose to guit (option 4).

listOps.py if n==1: add=int(input("Integer: ")) a.append(add) 10 11 print(f"List after adding: {a}") elif n==2: 13 if len(a)==0: 14 print("List is empty") elif len(a)!=0: remove=int(input("Integer: ")) > > if remove not in a : >> >> > print("Element not found") 19 -> else: 20 a.remove(remove) 21 > > > > > print(f"List after removing: {a}") elif n==3: if len(a)==0: print("List is empty") 24 25 else: 26 print(a) 27 elif n==4: 28 break else: 29 30 print("Invalid choice") 31 32

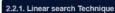
Sample Test Cases



< Prev Reset Submit

Next >

### dictOpera... 2.1.2. Dictionary Operations 3000 A L B 8 -D 17 print("Key not found") Write a Python program to perform the following dictionary operations: 18 · Create an empty dictionary and display it. retrieve key = input("Enter the key to retrieve: ") 19 · Ask the user how many items to add, then input key-value pairs. , if retrieve key in dict: 20 . Show the dictionary after adding items. 21 print(f"Key: {retrieve key}, Value: {dict[retrieve key]}") Ask the user to update a key's value. Print "Value updated" if the key exists, otherwise print "Key 22 v else: not found" 23 print("Key not found") · Retrieve and print a value using a key. If not found, print "Key not found". 24 . Use get() to retrieve a value. If the key doesn't exist, print "Key not found". 25 get\_key = input("Enter the key to get using the get() method: ") Delete a key-value pair. If the key exists, delete and print "Deleted". If not, print "Key not found". 26 value = dict.get(get\_key, "Key not found") . Display the updated dictionary. v if value != "Key not found": 28 print(f"Key: {get key}, Value: {value}") Note: Refer to visible test cases 29 , else: 30 print(value) 31 32 deleted\_key = input("Enter the key to delete: ") if deleted key in dict: 33 34 del dict[deleted kev] 35 print("Deleted") 36 , else: 37 print("Key not found") 38 39 print("Updated Dictionary:", dict) 40 Sample Test Cases 41 > Terminal ⊞ Test cases





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

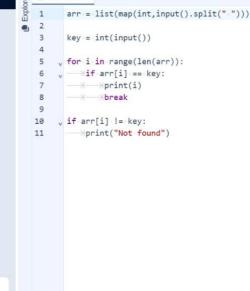
### **Output format:**

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

## Sample Test Case:

- Input:
- 1234356
- Output:

Sample Test Cases



> Terminal

@ CTP1709...

print(i) break

Logout @