2.1.1. List operations

43:10

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 quit (option 4).

Sample Test Cases +

Explorer

listOps.py

Submit

Debugger

```python
# write the code..
l1=[]
while True:
    print("1. Add")
    print("2. Remove")
    print("3. Display")
    print("4. Quit")
    n = int(input("Enter choice: "))
    if n == 1:
        add = int(input("Integer: "))
        l1.append(add)
        print(f"List after adding: {l1}")
    elif n == 2:
        if len(l1) == 0:
            print("List is empty")
        elif len(l1)!=0:
            remove=int(input("Integer: "))
            if remove not in l1:
                print("Element not found")
            else:
                l1.remove(remove)
                print(f"List after removing: {l1}")
    elif n == 3:
        if len(l1) == 0:
            print("List is empty")
        else:
            print(l1)
    elif n == 4:
        break
    else:
        print("Invalid choice")

```

Average time
0.206 s
205.67 ms

Maximum time
0.274 s
274.00 ms

2 out of 2 shown test case(s) passed

1 out of 1 hidden test case(s) passed

Test case 1 214 ms Debug

Expected output	Actual output
1. Add	1. Add
2. Remove	2. Remove
3. Display	3. Display

‹ Prev Reset Submit Next ›

2.1.2. Dictionary Operations

07:40

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 +

Explorer

dictOperati...

Submit

Debugger

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

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

retrieve_key=input("Enter the key to retrieve: ")
if retrieve_key in d:
    print(f"Key: {retrieve_key}, Value: {d[retrieve_key]}")
else:
    print("Key not found")

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

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

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

Average time
0.088 s
88.00 ms

Maximum time
0.100 s
100.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 75 ms Debug

Expected output	Actual output
Empty Dictionary: {}	Empty Dictionary: {}
Number of items: 1	Number of items: 1
key: Name	key: Name

2.2.1. Linear search Technique

01:18

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 +

Explorer

CTP17092...

Submit

Debugger

```python
# Function to perform linear search
def linear_search(arr, key):
    # Iterate through the array
    for i in range(len(arr)):
        if arr[i] == key:
            return i  # Return the index if the element is found
    return -1  # Return -1 if the element is not found

# Input: array of integers and key to search
arr = list(map(int, input().split()))  # Convert the input string to a list of integers
key = int(input())  # The key element to be searched

# Perform linear search
result = linear_search(arr, key)

# Output the result
if result != -1:
    print(result)  # If found, print the index
else:
    print("Not found")  # If not found, print "Not found"
```

Average time
0.021 s
21.00 ms

Maximum time
0.025 s
25.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 22 ms Debug

Expected output	Actual output
1 2 3 4 3 5 6	1 2 3 4 3 5 6
3	3
2	2

CODETANTRA

2.2.2. Captain of the Team

00:14

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 +

Explorer

captainofT...

Submit

Debugger

```
1  heights = list(map(int, input().split()))
2  tallest_player = max(heights)
3  print(tallest_player)
4
```

Average time
0.011 s
11.33 ms

Maximum time
0.017 s
17.00 ms

1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 17 ms Debug

Expected output	Actual output
171 169 185 156 174 191 186 190 187 172 160	171 169 185 156 174 191 186 190 187 172 160
191	191

‹ Prev Reset Submit Next ›