

PRACTICAL 2

ALL PRACTICALS:

The image displays two screenshots of the CodeTANTRA online coding platform, showing two different programming problems and their solutions.

Top Screenshot: 2.2.2. Captain of the Team

Problem Statement: 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:

Code:

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

Test Results:

- Average time: 0.005 s, Maximum time: 0.006 s
- 1 out of 1 shown test case(s) passed
- 2 out of 2 hidden test case(s) passed

Test Case 1:

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

Bottom Screenshot: 2.2.1. Linear search Technique

Problem Statement: 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
```

Code:

```
1 # Function to perform linear search
2 def linear_search(arr, key):
3     # Iterate through the array
4     for i in range(len(arr)):
5         if arr[i] == key:
6             return i # Return the index if the element is found
7     return -1 # Return -1 if the element is not found
8
9 # Input: array of integers and key to search
10 arr = list(map(int, input().split())) # Convert the input string to a list of integers
```

Test Results:

- Average time: 0.011 s, Maximum time: 0.012 s
- 2 out of 2 shown test case(s) passed
- 2 out of 2 hidden test case(s) passed

Test Case 1:

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

Test Case 2:

Expected output	Actual output
3	3
2	2

Upload files · Harshada02 · create · Course

mitaoe.codetantra.com/secure/course.jsp?eucdd=6773e3f2f1f9c5320ca6bce85#/contents/6773e44bf1f9c5320ca6bce8/6773e49ef1f9c5320ca6bd7d/677bb1ca63fd602f55ad634f

CODETANTRA Home202401120055@mitaoe.ac.inSupportLogout

2.1.2. Dictionary Operations17/50

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

dictOpera...

```
1 d={}
2 print("Empty Dictionary: {}")
3 n=int(input("Number of items: "))
4 for i in range(n):
5     key=input("key: ")
6     value=input("value: ")
7     d[key]=value
8     print("Dictionary:",d)
9
10 update_key=input("Enter the key to update: ")
11 if update_key in d:
```

Average time0.050 s49.75 msMaximum time0.081 s81.00 ms

2 out of 2 shown test case(s) passed2 out of 2 hidden test case(s) passed

Test case 161 ms

Expected output	Actual output
Empty Dictionary: {}	Empty Dictionary: {}
Number of items: 1	Number of items: 1
key: Name	key: Name
value: Alice	value: Alice
Dictionary: {'Name': 'Alice'}	Dictionary: {'Name': 'Alice'}
Enter the key to update: Name	Enter the key to update: Name

TerminalTest cases

Upload files · Harshada02 · create · Course

mitaoe.codetantra.com/secure/course.jsp?eucdd=6773e3f2f1f9c5320ca6bce85#/contents/6773e44bf1f9c5320ca6bce8/6773e49ef1f9c5320ca6bd7d/667bbd7f01060f3d690f3f83

CODETANTRA Home202401120055@mitaoe.ac.inSupportLogout

2.1.1. List operations62/33

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

- Add
- Remove
- Display
- 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

listOps.py

```
1 # write the code..
2 ll=[]
3 while True:
4     print("1. Add")
5     print("2. Remove")
6     print("3. Display")
7     print("4. Quit")
8     n = int(input("Enter choice: "))
9     if n == 1:
10         add = int(input("Integer: "))
11         ll.append(add)
```

Average time0.110 s109.67 msMaximum time0.180 s180.00 ms

2 out of 2 shown test case(s) passed1 out of 1 hidden test case(s) passed

Test case 164 ms

Expected output	Actual output
1. Add	1. Add
2. Remove	2. Remove
3. Display	3. Display
4. Quit	4. Quit
Enter choice: 1	Enter choice: 1
Integer: 11	Integer: 11

TerminalTest cases