

# Rajalakshmi Engineering College

Name: Shremathi K  
Email: 240701504@rajalakshmi.edu.in  
Roll no:  
Phone: 8870649491  
Branch: REC  
Department: I CSE FE  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## NeoColab\_REC\_CS23221\_Python Programming

### REC\_Python\_Week 3\_COD

Attempt : 1  
Total Mark : 50  
Marks Obtained : 50

### Section 1 : Coding

#### 1. Problem Statement

Ram is working on a program to manipulate strings. He wants to create a program that takes two strings as input, reverses the second string, and then concatenates it with the first string.

Ram needs your help to design a program.

#### ***Input Format***

The input consists of two strings in separate lines.

#### ***Output Format***

The output displays a single line containing the concatenated string of the first string and the reversed second string.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: hello  
word

Output: hellodrow

**Answer**

```
# You are using Python
str1 = input()
str2 = input()
reversed_str2 = str2[::-1]
result = str1 + reversed_str2
print(result)
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

You have a string containing a phone number in the format "(XXX) XXX-XXXX". You need to extract the area code from the phone number and create a new string that contains only the area code.

Write a Python program for the same.

Note

(XXX) - Area code

XXX-XXXX - Phone number

**Input Format**

The input consists of a string, representing the phone number in the format "(XXX) XXX-XXXX".

**Output Format**

The output displays "Area code: " followed by a string representing the area code for the given phone number.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: (123) 456-7890

Output: Area code: 123

### **Answer**

```
# You are using Python
phone_number = input()
area_code = phone_number[1:4]
print("Area code: " + area_code)
```

**Status :** Correct

**Marks :** 10/10

## **3. Problem Statement**

Given a list of positive and negative numbers, arrange them such that all negative integers appear before all the positive integers in the array. The order of appearance should be maintained.

Example

Input:

[12, 11, -13, -5, 6, -7, 5, -3, -6]

Output:

List = [-13, -5, -7, -3, -6, 12, 11, 6, 5]

Explanation:

The output is the arranged list where all the negative integers appear before the positive integers while maintaining the original order of appearance.

**Input Format**

The input consists of a single line containing a list of integers enclosed in square brackets separated by commas.

### ***Output Format***

The output displays "List = " followed by an arranged list of integers as required, separated by commas and enclosed in square brackets.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: [12, 11, -13, -5, 6, -7, 5, -3, -6]

Output: List = [-13, -5, -7, -3, -6, 12, 11, 6, 5]

### ***Answer***

```
# You are using Python
input_list = eval(input())
negatives = []
positives = []

for num in input_list:
    if num < 0:
        negatives.append(num)
    else:
        positives.append(num)

result = negatives + positives
print("List = ", result)
```

**Status :** Correct

**Marks :** 10/10

## **4. Problem Statement**

Alex is working on a Python program to manage a list of elements. He needs to append multiple elements to the list and then remove an element from the list at a specified index.

Your task is to create a program that helps Alex manage the list. The

program should allow Alex to input a list of elements, append them to the existing list, and then remove an element at a specified index.

### ***Input Format***

The first line contains an integer  $n$ , representing the number of elements to be appended to the list.

The next  $n$  lines contain integers, representing the elements to be appended to the list.

The third line of input consists of an integer  $M$ , representing the index of the element to be popped from the list.

### ***Output Format***

The first line of output displays the original list.

The second line of output displays the list after popping the element of the index  $M$ .

The third line of output displays the popped element.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 5

64

98

-1

5

26

3

Output: List after appending elements: [64, 98, -1, 5, 26]

List after popping last element: [64, 98, -1, 26]

Popped element: 5

### ***Answer***

# You are using Python

```
n = int(input())
```

```
lst = []

for _ in range(n):
    element = int(input())
    lst.append(element)

M = int(input())

print("List after appending elements:", lst)
popped_element = lst.pop(M)
print("List after popping last element:", lst)
print("Popped element:", popped_element)
```

**Status :** Correct

**Marks : 10/10**

## 5. Problem Statement

Dhruv wants to write a program to slice a given string based on user-defined start and end positions.

The program should check whether the provided positions are valid and then return the sliced portion of the string if the positions are within the string's length.

### ***Input Format***

The first line consists of the input string as a string.

The second line consists of the start position (0-based index) as an integer.

The third line consists of the end position (0-based index) as an integer.

### ***Output Format***

The output displays the following format:

If the start and end positions are valid, print the sliced string.

If the start and end positions are invalid, print "Invalid start and end positions".

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: pythonprogramming

0

5

Output: python

**Answer**

```
# You are using Python
```

```
input_string = input()
```

```
start = int(input())
```

```
end = int(input())
```

```
if 0 <= start <= end < len(input_string):
```

```
    print(input_string[start:end + 1])
```

```
else:
```

```
    print("Invalid start and end positions")
```

**Status : Correct**

**Marks : 10/10**