Section : 1 of 1 | Question : 5 of 8 | Marks: 5 | Jump to | General Instructions | 30% | Test time left: 02:02:08

Write the code

Your task is to :

- Define a class **shapes** that use the **init** method to initialize the **length(l), and width(w)** of the square, rectangle, and circle respectively.
- Also define three methods **area_sq(), area_tri(),** and **area_cube()**, that take the values of **l and w** from the **init** method and calculate the respective area.
- Create an **object** of **class shapes** to call the methods defined in class shapes.

Note:

- Understand the menu-driven format as displayed in the test cases
- *Area of square = l^2 , Area of cube = $6l^2$, Area of triangle = 1/2 * l * w*

Constraints:

length l, width w must be positive integers > 0.

Sample Test case:

1. square ------> Display the menu to select the shape.

2. triangle

3. cube

2 ------> Select the option

10 -----> Represent's l

5 -----> Represent's w

25.0 ------> Area of the shape selected

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:

Expected Output:

```
1.square
2.triangle
3.cube
1
4
16
```

Test Case 2:

Expected Output:

```
1.square
2.triangle
3.cube
2
10
```

Correct/complete the code. The code highlighted in is non-editable.

oops.py

Submit

```
class Shapes:

    def __init__(self):
        print('1.square')
        print('2.triangle')
        print('3.cube')
        i = int(input())
        if i == 1 or i == 3:
            l = int(input())
            if i == 1:
                print(self.area_sq(l))
            else:
                print(self.area_cube(l))
        elif i == 2:
            print(self.area_tri(int(input()), int(input())))
        else:
            print('Invalid')

    def area_sq(self, a):
        return a*a


    def area_tri(self, l, w):
        return 0.5*l*w


    def area_cube(self, l):
        return 6*l*l
Shapes()
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 136.6 ms , Max. Time: 233 ms)

Test Case - 1 (Execution Time: 233 ms) — Correct Output

Expected Output	User Output	Status
1.square	1.square	✓
2.triangle	2.triangle	✓
3.cube	3.cube	✓
1	1	
4	4	
16	16	✓

Test Case - 2 (Execution Time: 133 ms) — Correct Output

Expected Output	User Output	Status
1.square	1.square	✓
2.triangle	2.triangle	✓
3.cube	3.cube	✓
2	2	
10	10	

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 1 of 8 | Marks: 5 | Jump to | General Instructions | 100%

Write the code

Test time left: 02:27:23

Create a class **Test** with a single method **division()** that returns the division of two numbers.

Constraints:

1 <= numbers <= 100

Sample Test case:

100 -----> Enter number 1.

2 ------> Enter number 2.

50 ------> Print the result

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
class Test:
    def division(self, a, b):
        return a//b


obj= Test()
a = int(input())
b = int(input())
print(obj.division(a, b))
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 164.0 ms , Max. Time: 234 ms)

Sample Test Cases

Test Case 1:
Expected Output:
100
2
50

Test Case 2:
Expected Output:
47
3
15

Test Case - 1 (Execution Time: 213 ms)	Correct Output	
Expected Output	**User Output**	**Status**
100	100	
2	2	
50	50	✓

Test Case - 2 (Execution Time: 105 ms)	Correct Output	
Expected Output	**User Output**	**Status**
47	47	
3	3	
15	15	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 3 of 8 | Marks: 5 | Jump to | General Instructions | 100%

Write the code

Test time left: 02:20:43

Your task is to :

- Define a class **Person**, that uses **init** method to initialize two attributes: **firstname** and **lastname** of a person.
- Create an object of a class person and print only the last name of the person using the **object.**

Constraints:

1 <= length of the string <= 10

Note:

- **Refer to the displayed test cases for better understanding.**

Sample test case:

Lincoln

Issac

Issac

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:
Expected Output:
Roland
triumph
triumph

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
1  class Person:
2  #WRITE YOUR CODE HERE
3      def __init__(self, f, l):
4          self.f = f
5          self.l = l
6
7
8  firstname=input()
9  lastname=input()
10 myname=Person(firstname,lastname)
11 print(myname.l)
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 87 ms , Max. Time: 163 ms)

Test Case - 1 (Execution Time: 165 ms)	Correct Output	
Expected Output	**User Output**	**Status**
Roland	Roland	
triumph	triumph	
triumph	triumph	✓

Test Case - 2 (Execution Time: 78 ms)	Correct Output	
Expected Output	**User Output**	**Status**
Lincoln	Lincoln	
Issac	Issac	
Issac	Issac	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 4 of 8 | Marks: 5 | Jump to | General Instructions | 80% | Test time left: 02:11:18

Write the code

Write a menu-driven program to perform **swap case and reverse** operations of the string given by the user Using the **concept of classes**.

The functionalities of the program:

- **swap case:** Swaps cases, the lower case becomes the upper case and vice versa.
- **reverse:** return the reversed version of the string.
- You need to display the string after performing the selected operation.

Constraints:

The input string should only contain the alphabets.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**
- **The class name is given in the uneditable code itself.**

Sample test case:

1. swapcase -----> Display the menu.

2. reverse

Select: 1 ------> Enter the option

python -----> Input string

PYTHON -----> Print the resultant string.

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:
Expected Output:
1. swapcase 2. reverse

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py — Submit

```
class Check():
    def swapcase(self, s):
        n = ''
        for i in s:
            n += i.lower() if i.isupper() else i.upper()
        return n

    def reverse(self, s):
        return s[::-1]

obj=Check()
print(f'1. swapcase\n2. reverse')
i = input('Select: ')
if i == '1':
    s = input()
    print(obj.swapcase(s))
elif i == '2':
    s = input()
    print(obj.reverse(s))
else:
    print('Invalid')
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 1142 ms, Max. Time: 150 ms)

Test Case - 1 (Execution Time: 150 ms)		Correct Output
Expected Output	**User Output**	**Status**
1. swapcase	1. swapcase	✓
2. reverse	2. reverse	✓
Select: 1	Select: 1	✓
python	python	
PYTHON	PYTHON	✓

Test Case - 2 (Execution Time: 98 ms)		Correct Output
Expected Output	**User Output**	**Status**
1. swapcase	1. swapcase	✓
2. reverse	2. reverse	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 6 of 8 | Marks: 5 | Jump to | General Instructions | 130% | Test time left: 01:53:38

Write the code

Write a menu-driven program to **Add, Delete and Display** the elements of the list Using the **concept of classes**.

The functionalities of the program:

- The program should terminate only when the user selects **Exit**.
- **Add:** Add an element to the list and display the list.
- **Delete:** Remove an element from the list and display the list.
- While performing the deletion operation, if the element entered by the user is not in the list, it should display **"No element'.**

Constraints:

The list should only contain the integers.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**

Sample test case:

- - - - - ----> Print these hyphens just to separate each operation

0. Exit -------> Display the menu.

1. Add

2. Delete

3. Display

Select: 1 ------> Enter the option

22 -----> Enter the element

[22] -----> Display the list after performing the operation.

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
class Check():

    def __init__(self):
        self.li = []

    def add(self, a):
        self.li.append(a)
        self.dis()

    def remove(self, a):
        if a in self.li:
            self.li.remove(a)
            self.dis()
        else:
            print('No element')
    def dis(self):
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 33 ms , Max. Time: 75 ms)

Test Case - 1 (Execution Time: 33 ms)	Correct Output	
Expected Output	**User Output**	**Status**
- - - - - - -	- - - - - - -	✓
0. Exit	0. Exit	✓
1. Add	1. Add	✓
2. Delete	2. Delete	✓
3. Display	3. Display	✓
Select: 1	Select: 1	✓
22	22	
[22]	[22]	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 2 of 8 | Marks: 5 | Jump to | General Instructions | 100%

Writing a class in Python

Test time left: 02:21:52

Follow the given instructions to add the details of the 3 number of student names and their skill rating in python language by taking the inputs from the user.

- Create a class of students.
- Create an instance **student_1** of class students.
- Create another instance **student_2** of class students.
- Take the **Name and rating(self) in the python language** of the student as inputs from the user.
- Print the Name of the student who has lowest skill rating as shown in the Sample Test Case

Note:

- Refer to the displayed test cases for better understanding.

Sample Test case:

Input:

patrick

0

pranav

5

emiway

2

Output:

lowest skilled student to take care of

patrick

Instructions:

Your input and output must follow the input and output layout mentioned in the visible sample test

Correct/complete the code. The code highlighted in ■ is non-editable.

Classes1.py

Submit

```
class students:
    def __init__(self):
        self.name = input()
        self.rating = int(input())
a = students()
b = students()
c = students()

print("lowest skilled student to take care of")
z = a.name
lowest = a.rating
x = [b, c]
for i in x:
    if i.rating < lowest:
        lowest = i.rating
        z = i.name
print(z)

```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 110.5 ms, Max. Time: 160 ms)

Test Case - 1 (Execution Time: 168 ms)	Correct Output	
Expected Output	**User Output**	**Status**
john	john	
3	3	
deol	deol	
4	4	
kuldeep	kuldeep	
9	9	
lowest skilled student to take care of	lowest skilled student to take care of	✓
john	john	✓

Test Case - 2 (Execution Time: 156 ms)	Correct Output

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 8 of 8 | Marks: 5 | Jump to | General Instructions | 130% | Test time left: 01:50:51

Write the code

A bugged code is given to you having errors (**Syntax errors and/or logical errors**). Your task is to **debug the code** and execute it successfully.

Problem statement:

class Record is defined with **__init__** method to initialize **name, roll_no, and email_id** attributes. And code logic is create **record** instance of **Record** class , With the help of that instance or object read and display the record **n time's**, here **n** is an integer taken as input from the user.

Note: For more clarity go through the test case, and fine tune the syntax as well as logic of the code given in the editor.

Sample test case:

2 ----------> n i.e. Number of records (Type positive integer)

bruce lee----------> name (Type string)

1234---------->roll_no (Type positive integer)

lee@yahoo.com----------> email_id (Type string)

Record: 1 is bruce lee 1234 lee@yahoo.com ----------> Record 1 displayed

jacky chain

6789

jacky@gmail.com

Record: 2 is jacky chain 6789 jacky@gmail.com ----------> Record 2 displayed

Note: You have to rectify and execute the code available and full fill the requirements of the problem.

Instructions:

- Your input and output must follow the input and output layout mentioned in the

Correct/complete the code.

debug1.py | Submit

```
''' Debug the code.

'''
class Record:
    def __init__(self, name, roll_no, email_id):
        self.name = name
        self.roll_no = roll_no
        self.email_id = email_id
n = int(input())
for i in range(1,n+1):
    name = input()
    roll_no = int(input())
    email_id = input()
    record = Record(name, roll_no, email_id)
    print('Record:',i,'is' ,record.name, record.roll_no, record.ema
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 35.7 ms, Max. Time: 52 ms)

Test Case - 1 (Execution Time: 35 ms)	Correct Output	
Expected Output	**User Output**	**Status**
2	2	
bruce lee	bruce lee	
1234	1234	
lee@yahoo.com	lee@yahoo.com	
Record: 1 is bruce lee 1234 lee@yahoo.com	Record: 1 is bruce lee 1234 lee@yahoo.com	✓
jacky chain	jacky chain	
6789	6789	
jacky@gmail.com	jacky@gmail.com	
Record: 2 is jacky chain 6789 jacky@gmail.com	Record: 2 is jacky chain 6789 jacky@gmail.com	✓

Finish | Clear | Submit | Prev | Next

Write a menu-driven program to **Add, Delete and Display** the elements of the list Using the **concept of classes**.

The functionalities of the program:

- The program should terminate only when the user selects **Exit**.
- **Add:** Add an element to the list and display the list.
- **Delete:** Remove an element from the list and display the list.
- While performing the deletion operation, if the element entered by the user is not in the list, it should display **"No element'.**

Constraints:

The list should only contain the integers.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**

Sample test case:

- - - - - - ----> Print these hyphens just to separate each operation

0. Exit -------> Display the menu.

1. Add

2. Delete

3. Display

Select: 1 ------> Enter the option

22 -----> Enter the element

[22] -----> Display the list after performing the operation.

- - - - -

0. Exit -------> After each operation, display the menu

Jump to

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
run = True
while run:
    print('- - - - - -')
    print('0. Exit\n1. Add\n2. Delete\n3. Display')
    i = int(input('Select: '))
    if i == 0:
        print('Exit')
        run = False
    elif i == 1:
        z.add(int(input()))
    elif i == 2:
        z.remove(int(input()))
    elif i == 3:
        z.dis()
    # else:
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 61 ms , Max. Time: 76 ms)

Test Case - 1 (Execution Time: 33 ms)	Correct Output	
Expected Output	**User Output**	**Status**
- - - - - -	- - - - - -	✓
0. Exit	0. Exit	✓
1. Add	1. Add	✓
2. Delete	2. Delete	✓
3. Display	3. Display	✓
Select: 1	Select: 1	✓
22	22	
[22]	[22]	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 6 of 8 | Marks: 5 | Jump to | General Instructions | 130% | Test time left: 01:53:33

Write the code

Write a menu-driven program to **Add, Delete and Display** the elements of the list Using the **concept of classes**.

The functionalities of the program:

- The program should terminate only when the user selects **Exit**.
- **Add:** Add an element to the list and display the list.
- **Delete:** Remove an element from the list and display the list.
- While performing the deletion operation, if the element entered by the user is not in the list, it should display **"No element'.**

Constraints:

The list should only contain the integers.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**

Sample test case:

- - - - - ----> Print these hyphens just to separate each operation

0. Exit -------> Display the menu.

1. Add

2. Delete

3. Display

Select: 1 ------> Enter the option

22 -----> Enter the element

[22] -----> Display the list after performing the operation.

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
            print('No element')
    def dis(self):
        print(self.li)
z = Check()
run = True
while run:
    print('- - - - - -')
    print('0. Exit\n1. Add\n2. Delete\n3. Display')
    i = int(input('Select: '))
    if i == 0:
        print('Exit')
        run = False
    elif i == 1:
        z.add(int(input()))
    elif i == 2:
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 33 ms , Max. Time: 75 ms)

Test Case - 1 (Execution Time: 33 ms)	Correct Output	
Expected Output	**User Output**	**Status**
- - - - - -	- - - - - -	✓
0. Exit	0. Exit	✓
1. Add	1. Add	✓
2. Delete	2. Delete	✓
3. Display	3. Display	✓
Select: 1	Select: 1	✓
22	22	
[22]	[22]	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 7 of 8 | Marks: 5 | Jump to | General Instructions | 130% | Test time left: 01:52:35

Write the code

A bugged code is given to you having errors (**Syntax errors and/or logical errors**). Your task is to **debug the code** and execute it successfully.

Problem statement:

Class **chair** is defined with attribute **legs** as data member of the class. Code is updating the value in **legs** attribute i.e. 4 by multiplying the value in **legs** attribute n times, here n is an integer value taken as input from the user.

Note: You have to rectify and execute the code available and full fill the requirements of the problem.

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:
Expected Output:
50
1267650600228229401496703205376

Correct/complete the code.

debug1.py — Submit

```
1  '''
2  Debug the code.
3
4  '''
5  class Chair:
6      legs = 4
7  n = int(input())
8  c = Chair()
9  c.legs = c.legs**n
10 print(c.legs)
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

Test Case - 1 (Execution Time: 33 ms)	Correct Output	
Expected Output	**User Output**	**Status**
50	50	
1267650600228229401496703205376	1267650600228229401496703205376	✓

Test Case - 1 (Execution Time: 33 ms)	Correct Output	
Expected Output	**User Output**	**Status**
50	50	
1267650600228229401496703205376	1267650600228229401496703205376	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 5 of 8 | Marks: 5 | Jump to | General Instructions | 100% | Test time left: 02:08:49

Write the code

Your task is to :

- Define a class **shapes** that use the **init** method to initialize the **length(l), width(w),** and **radius(r)** of the square, rectangle, and circle respectively.
- Also define three methods **area_sq(), area_rec(),** and **area_cir()**, that take the values of **l, w,** and **r** from the **init** method and calculate the respective area.
- Create an **object** of **class shapes** to call the methods defined in class shapes.

Note:

- Understand the menu-driven format as displayed in the test cases

Constraints:

length l, width w, and radius r must be positive integers > 0.

The value of Pi is 3.14

Sample Test case:

1. square ------> Display the menu to select the shape.

2. rectangle

3. circle

2 -------> Select the option for the shape mentioned in the menu.

3 --------> Represent's l

5 --------> Represent's w.

Correct/complete the code. The code highlighted in ■ is non-editable.

oops.py

Submit

```
class Shapes:
    PI = 3.14

    def __init__(self):
        print('1.square\n2.rectangle\n3.circle')
        choice = input()
        if choice == '1':
            l = int(input())
            print(self.area_sq(l))
        elif choice == '2':
            l, w = int(input()), int(input())
            print(self.area_rec(l,w))
        elif choice == '3':
            r = int(input())
            print(self.area_cir(r))
        else:
            print('Invalid')

```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 43.8 ms , Max. Time: 72 ms)

Test Case - 1 (Execution Time: 33 ms)	Correct Output	
Expected Output	**User Output**	**Status**
1.square	1.square	✓
2.rectangle	2.rectangle	✓
3.circle	3.circle	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 3 of 8 | Marks: 5 | Jump to | General Instructions | 100% | Test time left: 02:23:38

Write the code

Your task is to :

- Define a class **Person**, that uses **init** method to initialize two attributes: **firstname, middlename** and **lastname** of a person.
- Create an object of a class person and print only the middle name of the person using the **object.**

Constraints:

1 <= length of the string <= 10

Note:

- **Refer to the displayed test cases for better understanding.**

Sample test case:

input

jMary

Elizabeth

Smith

Output

Elizabeth

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
1  class Person:
2  #WRITE YOUR CODE HERE
3      def __init__(self, fName, mName, lName):
4          self.firstName = fName
5          self.lastName = lName
6          self.middleName = mName
7
8
9  firstname=input()
10 middlename = input()
11 lastname=input()
12 myname=Person(firstname,middlename,lastname)
13 print(myname.middleName)
14
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 45.5 ms , Max. Time: 76 ms)

Test Case - 1 (Execution Time: 28 ms)	Correct Output	
Expected Output	**User Output**	**Status**
Mary	Mary	
Elizabeth	Elizabeth	
Smith	Smith	

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 4 of 8 | Marks: 5 | Jump to | General Instructions | 100% | Test time left: 02:16:46

Write the code

Write a menu-driven program to perform the **index and reverse** operations of the string given by the user Using the **concept of classes**.

The functionalities of the program:

- **index:** Searches the string for a specified character and returns the position of where it was found. If the user-specified character is not in the string it should return **not found**.
- **reverse:** return the reversed version of the string.
- You need to display the string after performing the selected operation.

Constraints:

The input string should only contain the alphabet.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**
- **The class name is given in the uneditable code itself.**

Sample test case:

1. count -----> Display the menu.

2. reverse

Select: 1 ------> Enter the option

python -----> Input string

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
class Check():
    def __init__(self, string="demo"):
        self.string = string

    def index(self, ch):
        if ch in self.string:
            return self.string.index(ch)
        else:
            return 'not found'

    def reverse(self):
        return self.string[::-1]

    def menu(self):
        print(f'1. index\n2. reverse')
        choice = input('Select: ')
        if choice == '1':
            self.string = input()
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 39 ms , Max. Time: 72 ms)

Test Case - 1 (Execution Time: 24 ms)	Correct Output	
Expected Output	**User Output**	**Status**
1. index	1. index	✓
2. reverse	2. reverse	✓
Select: 1	Select: 1	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 4 of 8 | Marks: 5 | Jump to | General Instructions | 100% | Test time left: 02:16:41

Write the code

Write a menu-driven program to perform the **index and reverse** operations of the string given by the user Using the **concept of classes**.

The functionalities of the program:

- **index:** Searches the string for a specified character and returns the position of where it was found. If the user-specified character is not in the string it should return **not found**.
- **reverse:** return the reversed version of the string.
- You need to display the string after performing the selected operation.

Constraints:

The input string should only contain the alphabet.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**
- **The class name is given in the uneditable code itself.**

Sample test case:

1. count -----> Display the menu.

2. reverse

Select: 1 ------> Enter the option

python -----> Input string

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
10
11      def reverse(self):
12          return self.string[::-1]
13
14      def menu(self):
15          print(f'1. index\n2. reverse')
16          choice = input('Select: ')
17          if choice == '1':
18              self.string = input()
19              ch = input()
20              print(self.index(ch))
21          elif choice == '2':
22              self.string = input()
23              print(self.reverse())
24          else:
25              print('Invalid')
26  obj=Check()
27  obj.menu()
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 39 ms , Max. Time: 72 ms)

Test Case - 1 (Execution Time: 24 ms)	Correct Output	
Expected Output	**User Output**	**Status**
1. index	1. index	✓
2. reverse	2. reverse	✓
Select: 1	Select: 1	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 2 of 8 | Marks: 5 | Jump to | General Instructions | 100%

Writing a class in Python

Test time left: 02:25:16

Follow the given instructions to add the details of the two companies by taking the inputs from the user.

- Create a class Company
- Create an instance **Com_1** of class Company.
- Create another instance **Com_2** of class Company.
- Take the **Name, No.of employees, and Revenue(in crores)** of the company as inputs from the user.
- Print the details of the Company as shown in the Sample Test Case

Note:

- Refer to the displayed test cases for better understanding.

Sample Test case:

No.of employees and Revenue must be positive integers > 0.

Company 1

CodeTantra ----> Name of the Company 1.

100 ----> No. of employees of Company 1.

2 ----> Revenue of the Company 1.

Name: CodeTantra ----> Display the details of Company 1.

Employee count: 100

Revenue: 2

Company 2

Correct/complete the code. The code highlighted in ■ is non-editable.

Classes1.py

Submit

```
class Company:
    def __init__(self, name, employees, revenue):
        self.name = name
        self.employees = employees
        self.revenue = revenue

    def info(self):
        print(f'Name: {self.name}')
        print(f'Employee count: {self.employees}')
        print(f'Revenue: {self.revenue}')


print("Company 1")
c1 = Company(input(), int(input()), int(input()))
c1.info()

print("Company 2")
c2 = Company(input(), int(input()), int(input()))
c2.info()
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 28.5 ms , Max. Time: 29 ms)

Test Case - 1 (Execution Time: 29 ms)	Correct Output	
Expected Output	**User Output**	**Status**
Company 1	Company 1	✓
CodeTantra	CodeTantra	
100	100	

Finish | Clear | Submit | Prev | Next

Write the code

A bugged code is given to you having errors (**Syntax errors and/or logical errors**). Your task is to **debug the code** and execute it successfully.

Problem statement:

Class **chair** is defined with attribute **legs** as data member of the class. Code is updating the value in **legs** attribute i.e. 4 by multiplying the value in **legs** attribute n times, here n is an integer value taken as input from the user.

Note: You have to rectify and execute the code available and full fill the requirements of the problem.

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:
Expected Output:

Correct/complete the code.

debug1.py

Submit

```
1  '''
2  Debug the code.
3
4  '''
5  class Chair:
6      legs = 4
7  n = int(input())
8  c = Chair()
9  c.legs = c.legs**n
10 print(c.legs)
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!
(Avg. Time: 40 ms , Max. Time: 72 ms)

Test Case - 1 (Execution Time: 36 ms)	Correct Output	
Expected Output	**User Output**	**Status**
50	50	
126765060022822940149670320537 6	126765060022822940149670320537 6	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 5 of 8 | Marks: 5 | Jump to | General Instructions | 100% | Test time left: 02:08:45

Write the code

Your task is to :

- Define a class **shapes** that use the **init** method to initialize the **length(l), width(w),** and **radius(r)** of the square, rectangle, and circle respectively.
- Also define three methods **area_sq(), area_rec(),** and **area_cir()**, that take the values of **l, w,** and **r** from the **init** method and calculate the respective area.
- Create an **object** of **class shapes** to call the methods defined in class shapes.

Note:

- Understand the menu-driven format as displayed in the test cases

Constraints:

length l, width w, and radius r must be positive integers > 0.

The value of Pi is 3.14

Sample Test case:

1. square ------> Display the menu to select the shape.

2. rectangle

3. circle

2 -------> Select the option for the shape mentioned in the menu.

3 --------> Represent's l

5 --------> Represent's w.

Correct/complete the code. The code highlighted in ■ is non-editable.

oops.py

Submit

```
            l, w = int(input()), int(input())
            print(self.area_rec(l,w))
        elif choice == '3':
            r = int(input())
            print(self.area_cir(r))
        else:
            print('Invalid')

    def area_sq(self, l):
        return l**2

    def area_rec(self, l, w):
        return l*w

    def area_cir(self, r):
        return self.PI*r**2

Shapes()
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 43.8 ms , Max. Time: 72 ms)

Test Case - 1 (Execution Time: 33 ms)	Correct Output	
Expected Output	**User Output**	**Status**
1.square	1.square	✓
2.rectangle	2.rectangle	✓
3.circle	3.circle	✓

Finish | Clear | Submit | Prev | Next

end of the current list.

- **Delete:** Remove an element from the list and display the list.
- While performing the deletion operation, If the element entered by the user is not in the list, it should display **"No element'.**

Constraints:

The list should only contain the integers.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**

Sample test case:

- - - - - ----> Print these hyphens just to separate each operation

0. Exit ------> Display the menu

1. Extend

2. Delete

3. Display

Select: 1 -----> Enter option

1,2,3 -----> Enter the list to add it to the existing list.

[1, 2, 3] ------> Display the list.

- - - - -

0. Exit

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
class Check():

    def __init__(self):
        self.li = []

    def extend(self, l):
        self.li.extend(l)
        self.dis()

    def remove(self, el):
        if el in self.li:
            self.li.remove(el)
            self.dis()
        else:
            print('No element')

    def dis(self):
        print(self.li)
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!
(Avg. Time: 32.7 ms , Max. Time: 38 ms)

Test Case - 1 (Execution Time: 32 ms)	Correct Output	
Expected Output	**User Output**	**Status**
---------	---------	✓
0. Exit	0. Exit	✓

Finish | Clear | Submit | Prev | Next

Write the code

Write a menu-driven program to **Extend, Delete and Display** the elements of the list Using the **concept of classes**.

The functionalities of the program:

- The program should terminate only when the user selects **Exit**.
- **Extend:** Extend method adds the specified list elements (or any iterable) to the end of the current list.
- **Delete:** Remove an element from the list and display the list.
- While performing the deletion operation, If the element entered by the user is not in the list, it should display **"No element'.**

Constraints:

The list should only contain the integers.

Note:

- **Understand the Menu-driven format which is displayed in the test cases.**

Sample test case:

- - - - - - ----> Print these hyphens just to separate each operation

0. Exit ------> Display the menu

1. Extend

2. Delete

Correct/complete the code. The code highlighted in ■ is non-editable.

classes.py

Submit

```
run = Check()
while True:
    print('- - - - - -')
    print('0. Exit\n1. Extend\n2. Delete\n3. Display')
    choice = input('Select: ')
    if choice == '0':
        print('Exit')
        break
    elif choice == '1':
        l = [int(i) for i in input().split(',')]
        run.extend(l)
    elif choice == '2':
        el = int(input())
        run.remove(el)
    elif choice == '3':
        run.dis()
    else:
        print('Invalid')
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!
(Avg. Time: 32.7 ms , Max. Time: 38 ms)

Test Case - 1 (Execution Time: 32 ms)	Correct Output	
Expected Output	**User Output**	**Status**
--------	--------	✓
0. Exit	0. Exit	✓

Finish | Clear | Submit | Prev | Next

Section : 1 of 1 | Question : 1 of 8 | Marks: 5 | Jump to | General Instructions | 100% | Test time left: 02:29:06

Write the code

Create a class **Test** with a single method **multiplication()** that returns the multiplied value of two numbers.

Constraints:

1 <= numbers <= 100

Sample Test case:

input:

2 -----> Enter number 1.

6 ------> Enter number 2.

output:

12 ------> Print the result

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when the user's input and output match the expected input and output.

Sample Test Cases

Test Case 1:
Expected Output:

Correct/complete the code. The code highlighted in is non-editable.

classes.py

Submit

```
1  class Test:
2      def multiplication(self, a, b):
3          return a*b
4
5
6
7  obj= Test()
8  a = int(input())
9  b = int(input())
10 print(obj.multiplication(a, b))
```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 42.8 ms , Max. Time: 85 ms)

Test Case - 1 (Execution Time: 32 ms)	Correct Output	
Expected Output	User Output	Status
2	2	
6	6	
12	12	

Finish | Clear | Submit | Prev | Next

Problem statement:

class Record is defined with **__init__** method to initialize **name, roll_no, and email_id** attributes. And code logic is create **record** instance of **Record** class , With the help of that instance or object read and display the record **n time's**, here **n** is an integer taken as input from the user.

Note: For more clarity go through the test case, and fine tune the syntax as well as logic of the code given in the editor.

Sample test case:

2 ----------> n i.e. Number of records (Type positive integer)

bruce lee----------> name (Type string)

1234---------->roll_no (Type positive integer)

lee@yahoo.com----------> email_id (Type string)

Record: 1 is bruce lee 1234 lee@yahoo.com ----------> Record 1 displayed

jacky chain

6789

jacky@gmail.com

Record: 2 is jacky chain 6789 jacky@gmail.com ----------> Record 2 displayed

Note: You have to rectify and execute the code available and full fill the requirements of the problem.

Instructions:

Correct/complete the code.

debug1.py

Submit

```
'''
Debug the code.

'''
class Record:
    def __init__(self, name, roll_no, email_id):
        self.name = name
        self.roll_no = roll_no
        self.email_id = email_id

n = int(input())
for i in range(1, n + 1):
    name = input()
    roll_no = int(input())
    email_id = input()
    record = Record(name, roll_no, email_id)
    print('Record:',i ,'is' ,record.name, record.roll_no, recor

```

Terminal | Execution Results

Execution Results - All test cases have succeeded!
(Avg. Time: 34.7 ms , Max. Time: 47 ms)

Test Case - 1 (Execution Time: 31 ms)	Correct Output
Expected Output	**User Output**
2	2
bruce lee	bruce lee
1234	1234

Section : 1 of 1 | Question : 2 of 8 | Marks: 5 | Jump to | General Instructions | 100%

Writing a class in Python

Test time left: 02:21:52

Follow the given instructions to add the details of the 3 number of student names and their skill rating in python language by taking the inputs from the user.

- Create a class of students.
- Create an instance **student_1** of class students.
- Create another instance **student_2** of class students.
- Take the **Name and rating(self) in the python language** of the student as inputs from the user.
- Print the Name of the student who has lowest skill rating as shown in the Sample Test Case

Note:

- Refer to the displayed test cases for better understanding.

Sample Test case:

Input:

patrick

0

pranav

5

emiway

2

Output:

lowest skilled student to take care of

patrick

Instructions:

Your input and output must follow the input and output layout mentioned in the visible sample test

Correct/complete the code. The code highlighted in ■ is non-editable.

Classes1.py

Submit

```
class students:
    def __init__(self):
        self.name = input()
        self.rating = int(input())
a = students()
b = students()
c = students()

print("lowest skilled student to take care of")
z = a.name
lowest = a.rating
x = [b, c]
for i in x:
    if i.rating < lowest:
        lowest = i.rating
        z = i.name
print(z)

```

Terminal | Execution Results

Execution Results - All test cases have succeeded!

(Avg. Time: 110.5 ms, Max. Time: 160 ms)

Test Case - 1 (Execution Time: 168 ms)	Correct Output	
Expected Output	**User Output**	**Status**
john	john	
3	3	
deol	deol	
4	4	
kuldeep	kuldeep	
9	9	
lowest skilled student to take care of	lowest skilled student to take care of	✓
john	john	✓

Test Case - 2 (Execution Time: 156 ms)	Correct Output

Finish | Clear | Submit | Prev | Next