You can view this report online at : https://www.hackerrank.com/x/tests/1731394/candidates/57322921/report

Full Name:	Instructor
Email:	aisha.batool@sse.habib.edu.pk
Test Name:	**CS101 - PW10 - Fall23**
Taken On:	22 Oct 2023 19:40:21 PKT
Time Taken:	20 min 42 sec/ 10080 min
Work Experience:	> 5 years
Invited by:	Aisha
Skills Score:	
Tags Score:	CS101 20/20 Python 3 10/10 Recursion 10/10

100%

50/50

scored in **CS101 - PW10 - Fall23** in 20 min 42 sec on 22 Oct 2023 19:40:21 PKT

Recruiter/Team Comments:

No Comments.

Question Description	Time Taken	Score	Status
Q1 **Number Sequence** > **Coding**	54 sec	10/ 10	✓
Q2 **Ranged countdown - Recursively** > **Coding**	41 sec	10/ 10	✓
Q3 **Recursively slow conceal** > **Coding**	1 min 6 sec	10/ 10	✓
Q4 **Guess that number** > **Coding**	14 min 52 sec	10/ 10	✓
Q5 **Cup** > **Coding**	1 min 10 sec	10/ 10	✓
Q6 **Difficulty Meter** > **Multiple Choice**	23 sec	0/ 0	✓

QUESTION 1

Correct Answer

Score 10

Number Sequence > Coding

QUESTION DESCRIPTION

Problem

Write a *recursive* function named `sequence` that takes integer parameters named `x` and `y` and prints all the characters between `x` and `y` inclusive.

Sample

```
>>> sequence(1,11)
1
```

```
2
3
4
5
6
7
8
9
10
11
>>> sequence(5,5)
5
```

Constraints

- `isinstance(x, int)` is `True`
- `isinstance(y, int)` is `True`
- `x <= y` is `True`

Note

The function *must* be recursive. Non-recursve functions will receive a score of 0.

INTERVIEWER GUIDELINES

Solution

```
def sequence(a, b):
    print(a)
    if a < b:
        sequence(a+1, b)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def sequence(a, b):
2     print(a)
3     if a < b:
4         sequence(a+1, b)
5
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	2	0.075 sec	11.7 KB
Testcase 1	Easy	Sample case	Success	2	0.1138 sec	11.7 KB
Testcase 2	Easy	Hidden case	Success	2	0.0832 sec	11.3 KB
Testcase 3	Easy	Hidden case	Success	2	0.0649 sec	11.7 KB
Testcase 4	Easy	Hidden case	Success	2	0.0774 sec	11.7 KB

No Comments

QUESTION 2

Correct Answer

Ranged countdown - Recursively › Coding Recursion CS101

QUESTION DESCRIPTION

Score 10

Problem

Write a *recursive* function named `countdown2` that takes two parameters `a` and `b` and prints all numbers from `b` down to `a` inclusive.

Sample

```
>>> countdown2(-3, 5)
5, 4, 3, 2, 1, 0, -1, -2, -3
>>> countdown2(0, 0)
0
```

Input

Input `a` and `b` from the console without any prompt.

Constraints

- `isinstance(a, int)` is `True`
- `isinstance(b, int)` is `True`
- `b >= a` is `True`

INTERVIEWER GUIDELINES

Solution

```
a = int(input())
b = int(input())
def countdown2(a, b):
    if b == a:
        print(b)
    else:
        print(b, end=', ')
        countdown2(a, b-1)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  # Enter your code here.
2  a = int(input())
3  b = int(input())
4  def countdown2(a, b):
5      if b == a:
6          print(b)
7      else:
8          print(b, end=', ')
9          countdown2(a, b-1)
10
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	2	0.0904 sec	11.7 KB
Testcase 1	Easy	Sample case	Success	2	0.091 sec	11.5 KB
Testcase 2	Easy	Hidden case	Success	3	0.1347 sec	11.7 KB
Testcase 3	Easy	Hidden case	Success	3	0.107 sec	11.5 KB

No Comments

QUESTION 3

Correct Answer

Score 10

Recursively slow conceal › Coding

QUESTION DESCRIPTION

Challenge

Write a **recursive** function `slow_conceal(s)` that prints the lines in reverse order, i.e., the entire string `s` is printed first, then the string `s` with the last letter missing, then the last two letters missing, and so on, until only the first letter in `s` is printed.

Note: you may not use while or for loops.

Sample

```
>>> slow_conceal('Pizza!')
Pizza!
Pizza
Pizz
Piz
Pi
P
```

INTERVIEWER GUIDELINES

Solution

```
def slow_conceal(s):
    if s == "":
        return ""
    else:
        print(s)
        slow_conceal(s[:-1])
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def slow_conceal(s):
2     if s == "":
3         return ""
4     else:
5         print(s)
6         slow_conceal(s[:-1])
7
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	2.5	0.0544 sec	9.54 KB
Testcase 1	Easy	Sample case	Success	2.5	0.0719 sec	9.35 KB
Testcase 2	Easy	Sample case	Success	2.5	0.0674 sec	9.12 KB
Testcase 3	Easy	Sample case	Success	2.5	0.0386 sec	9.13 KB

No Comments

QUESTION 4

Correct Answer

Score 10

Guess that number › Coding CS101 Python 3

QUESTION DESCRIPTION

Write a program that simulates a guessing game between two players. See sample interaction below. All input values are provided by HackerRank. Input values in red are assumed to be provided by Poser, the player who knows the secret number. Input values in blue are assumed to be provided by Guess, the player who has to guess the secret number.

Sample Interaction

```
>>> guess(37, 5)
Attempts remaining: 5
50
You guessed: 50
Your guess is too high. Try again!
Attempts remaining: 4
25
You guessed: 25
Your guess is too low. Try again!
Attempts remaining: 3
40
You guessed: 40
Your guess is too high. Try again!
Attempts remaining: 2
30
You guessed: 30
Your guess is too low. Try again!
Attempts remaining: 1
35
You guessed: 35
Sorry, you lose! The secret number is 37 and you have no more attempts
remaining.

>>> guess(80, 10)
Attempts remaining: 10
50
You guessed: 50
Your guess is too low. Try again!
Attempts remaining: 9
75
You guessed: 75
Your guess is too low. Try again!
Attempts remaining: 8
80
You guessed: 80
Congratulations, you won! You guessed the secret number 80 with 7
attempt(s) remaining.
```

Function Description

To implement the given task, write the following function and supporting code:

1. ***guess*** that takes two arguments: ***secret*** and ***attempts***. The function should simulate the interaction as shown above by reading guesses as input, until either the secret is correctly guessed, or there are no more attempts remaining.
2. take two inputs and store their values as integers in the variables ***s*** and ***a***.

Constraints

- You must use recursion to **print** the interaction--you may not use for or while loops, or any other form of iteration.
- You may not reference global variables.

- Input and output should be handled by you--HackerRank will call the function *guess* with correct arguments.
- *secret* is a positive integer below 1000, and *attempt* is a positive integer below 20.

▼ Input Format For Custom Testing

The first line contains *secret*, the number that the poser knows, and the guesser has to discern from the clues.

The second line contains *attempts*, the initial number of attempts available to the guesser.

▼ Sample Case 0

Sample Input For Custom Testing

```
37
5
50
25
40
30
35
```

Sample Output

```
Attempts remaining: 5
You guessed: 50
Your guess is too high. Try again!
Attempts remaining: 4
You guessed: 25
Your guess is too low. Try again!
Attempts remaining: 3
You guessed: 40
Your guess is too high. Try again!
Attempts remaining: 2
You guessed: 30
Your guess is too low. Try again!
Attempts remaining: 1
You guessed: 35
Sorry, you lose! The secret number is 37 and you have no more attempts
remaining.
```

Explanation

The secret number is 37, and the guesser has 5 attempts to guess the number. Guesser begins with the number 50, which is more than 37. In the next attempt, guesser tries 25, which is less than 37. In the next attempt, guesser tries 40, which is more than 37. In the next attempt, guesser tries 30, which is less than 37. In the next attempt, guesser tries 35, which is less than 37. Since the guesser has exhausted all five attempts, a message with regrets is displayed.

▼ Sample Case 1

Sample Input For Custom Testing

```
80
10
50
75
80
```

Sample Output

```
Attempts remaining: 10
You guessed: 50
Your guess is too low. Try again!
Attempts remaining: 9
You guessed: 75
Your guess is too low. Try again!
Attempts remaining: 8
```

```
You guessed: 80
Congratulations, you won! You guessed the secret number 80 with 7
attempt(s) remaining.
```

Explanation

The secret number is 80, and the guesser has 10 attempts to guess the number. Guesser begins with the number 50, which is less than 80. In the next attempt, guesser tries 75, which is less than 80. In the next attempt, guesser correctly chooses 80. A message with congratulations is displayed. Since he entered three numbers in all, he has 10 - 3 = 7 attempts remaining.

INTERVIEWER GUIDELINES

```
def guess(secret, attempts):
    print('Attempts remaining:', attempts)
    g = int(input())
    print('You guessed:', g)
    if g == secret:
        print('Congratulations, you won! You guessed the secret number',
secret, 'with', attempts - 1, 'attempt(s) remaining.')
    elif attempts == 1:
        print('Sorry, you lose! The secret number is', secret, 'and you
have no more attempts remaining.')
    else:
        if g < secret:
            print('Your guess is too low. Try again!')
        else:
            print('Your guess is too high. Try again!')
        guess(secret, attempts - 1)

s = int(input())
a = int(input())
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  def guess(secret, attempts):
2      print('Attempts remaining:', attempts)
3      g = int(input())
4      print('You guessed:', g)
5      if g == secret:
6          print('Congratulations, you won! You guessed the secret number',
7  secret, 'with', attempts - 1, 'attempt(s) remaining.')
8      elif attempts == 1:
9          print('Sorry, you lose! The secret number is', secret, 'and you have
10 no more attempts remaining.')
11     else:
12         if g < secret:
13             print('Your guess is too low. Try again!')
14         else:
15             print('Your guess is too high. Try again!')
16         guess(secret, attempts - 1)
17
18 s = int(input())
   a = int(input())
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	1	0.0964 sec	11.7 KB

TestCase 1	Easy	Sample case	Success	1	0.0885 sec	11.6 KB
TestCase 2	Easy	Sample case	Success	1	0.0621 sec	11.8 KB
TestCase 3	Easy	Sample case	Success	1	0.0752 sec	11.6 KB
TestCase 4	Easy	Sample case	Success	1	0.0826 sec	11.6 KB
TestCase 5	Easy	Hidden case	Success	1	0.1033 sec	11.7 KB
TestCase 6	Easy	Hidden case	Success	1	0.1107 sec	11.7 KB
TestCase 7	Easy	Hidden case	Success	1	0.0924 sec	11.7 KB
TestCase 8	Easy	Hidden case	Success	1	0.1241 sec	11.7 KB
TestCase 9	Easy	Hidden case	Success	1	0.0946 sec	11.5 KB

No Comments

QUESTION 5

Correct Answer

Score 10

Cup › Coding

QUESTION DESCRIPTION

Problem

Write a **recursive** function named `cup` to generate the following pattern for a parameter, `n`.

Sample

```
>>> cup(4)
* * * *
 * * *
  * *
   *
>>> cup(2)
* *
 *
```

Hint

Define a *recursive* helper function to call from `cup`.

Input

You should input `n` from the console with any prompt.

Constraints

- `isinstance(n, int)` is `True`

INTERVIEWER GUIDELINES

Solution

```
n=int(input())
def cup(n):
    cup1(n,n)

def cup1(i,n):
    if i > 0:
        spaces = ' ' * (n-i)
        stars = '* ' * (i-1) + '*'
        print(spaces + stars)
        cup1(i-1, n)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
# Enter your code here.
n=int(input())
def cup(n):
    cup1(n,n)

def cup1(i,n):
    if i > 0:
        spaces = ' ' * (n-i)
        stars = '* ' * (i-1) + '*'
        print(spaces + stars)
        cup1(i-1, n)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	2	0.0626 sec	9.46 KB
TestCase 1	Easy	Hidden case	Success	2	0.051 sec	9.34 KB
TestCase 2	Easy	Hidden case	Success	2	0.0465 sec	9.55 KB
TestCase 3	Easy	Sample case	Success	2	0.0367 sec	9.43 KB
Testcase 4	Easy	Sample case	Success	2	0.0392 sec	9.41 KB

No Comments

QUESTION 6

Correct Answer

Score 0

Difficulty Meter › Multiple Choice

QUESTION DESCRIPTION

On a scale of 1 to 5, with 1 being very easy and 5 being extremely challenging, how would you rate this worksheet?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- (●) 1
- ( ) 2
- ( ) 3
- ( ) 4
- ( ) 5

No Comments