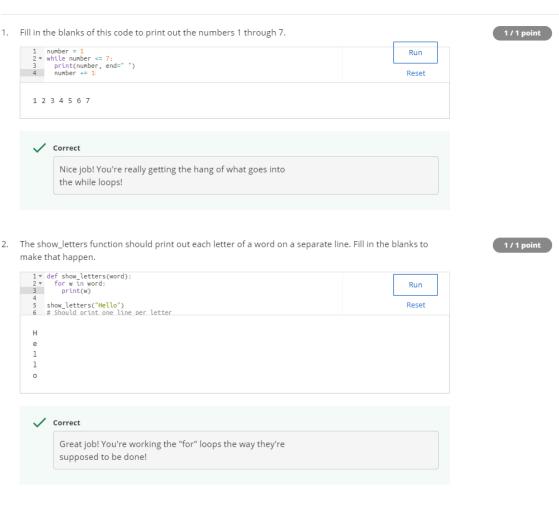


## **Module 3 Graded Assessment**

LATEST SUBMISSION GRADE

100%



Complete the function digits(n) that returns how many digits the number has. For example: 25 has 2 digits
and 144 has 3 digits. Tip: you can figure out the digits of a number by dividing it by 10 once per digit until
there are no digits left.

1/1 point

✓ Correct

Woohoo! You've cracked the code of writing code!

4. This function prints out a multiplication table (where each number is the result of multiplying the first number of its row by the number at the top of its column). Fill in the blanks so that calling multiplication\_table(1, 3) will print out:

1 / 1 point

123

246

369





Awesome! You've stepped up to the challenge of one of the more complex coding practices, nested loops!

The counter function counts down from start to stop when start is bigger than stop, and counts up from start to stop otherwise. Fill in the blanks to make this work correctly. 1 / 1 point

```
1 * def counter(start, stop):
    x = start
3 * if x > stop:
    return_string = "Counting down: "
    while x >= stop:
        return_string += str(x)
    if x |= stop:
        return_string = "Counting up: "
        while x <= stop:
        return_string = "Counting up: "
    while x <= stop:
        return_string += str(x)
    if x |= stop:
        return_string += str(x)
    if x |= stop:
        return_string += ","
        x += 1
17     return_string += ","
    x += 1
17     return_extring += ","
    x += 1
18
19     print(counter(1, 10)) # Should be "Counting up: 1,2,3,4,5,6,7,8,9,10"
    print(counter(2, 1)) # Should be "Counting up: 2,1"
        Print(counter(3, 5)) # Should be "Counting up: 5"

Counting up: 1,2,3,4,5,6,7,8,9,10

Counting up: 5
```



You nailed it! You've figured out all of the situations that need to be considered!

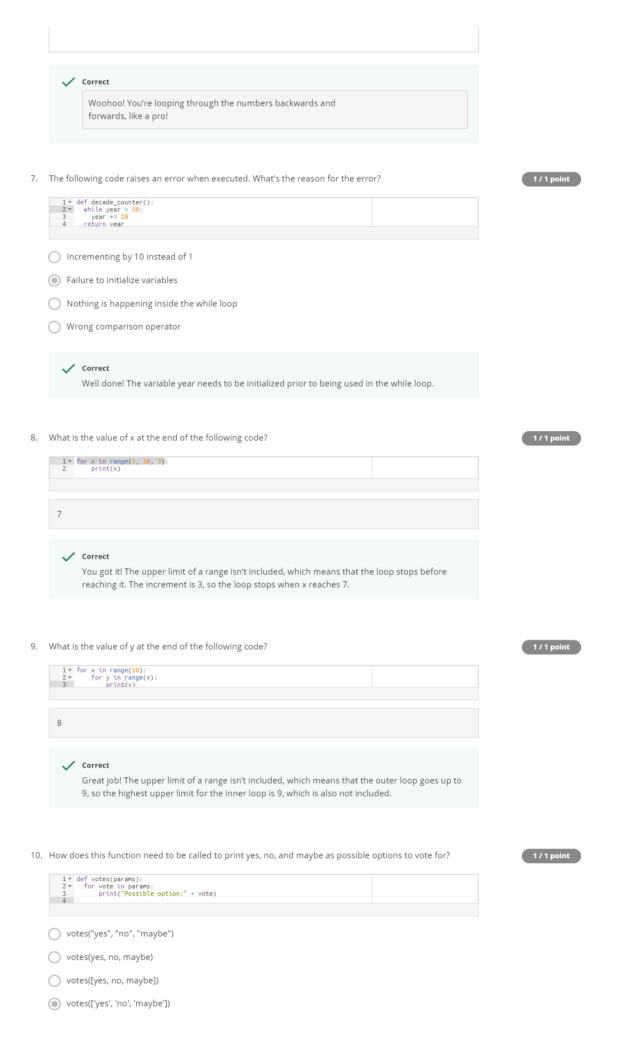
6. The loop function is similar to range(), but handles the parameters somewhat differently: it takes in 3 parameters: the starting point, the stopping point, and the increment step. When the starting point is greater than the stopping point, it forces the steps to be negative. When, instead, the starting point is less than the stopping point, it forces the step to be positive. Also, if the step is 0, it changes to 1 or -1. The result is returned as a one-line, space-separated string of numbers. For example, loop(11,2,3) should return 11 85 and loop(1,5,0) should return 12 34. Fill in the missing parts to make that happen.

1 / 1 point

```
1 * def loop(start, stop, step):
    return_string = ""
3 * if step == 0:
    step == 0:
    step == 15 * if start > stop:
    step = abs(step) * -1
    else:
        step = abs(step) * -1
        return_string += str(count) + " "
        return_string += str(count) + " "
        return_string.strip()

2    print(loop(11,2,3)) # Should be 11 8 5
    print(loop(1,5,0)) # Should be 12 3 4
    print(loop(1,2,0)) # Should be 10 12 14 16 18 20 22 24
        Reset

11 8 5
    1 2 3 4
    -1
    10 12 14 16 18 20 22 24
```



✓ Correct

 ${\bf Excellent!}\ {\bf This}\ {\bf function}\ {\bf is}\ {\bf looking}\ {\bf for}\ {\bf one}\ {\bf argument}, \ {\bf and}\ {\bf the}\ {\bf list}\ {\bf of}\ {\bf strings}\ {\bf is}\ {\bf just}\ {\bf one}\ {\bf argument}.$