Back **Module 3 Graded Assessment**
Graded Quiz • 50 min • 10 total points

English **Due** Dec 17, 11:59 PM EST

1. Fill in the blanks to print the numbers 1 through 7. **1 point**

```
number = 0 # Initialize the variable
while number < 8: # Complete the while loop condition
    print(number, end=" ")
    number += 1 # Increment the variable

# Should print 1 2 3 4 5 6 7
```

Run
Reset

```
0 1 2 3 4 5 6 7
```

2. Find and correct the error in the for loop. The loop should print every number from 5 to 0 in descending order. **1 point**

```
for number in range(5,-1,-1):
    print(number)

# Should print:
# 5
# 4
# 3
# 2
# 1
# 0
```

Run
Reset

```
5
4
3
2
1
0
```

3. Fill in the blanks to complete the function "digits(n)" to count how many digits the given number has. For example: 25 has 2 digits and 144 has 3 digits. **1 point**

Tip: you can count the digits of a number by dividing it by 10 once per digit until there are no digits left.

```
def digits(n):
    count = 0
    if n == 0:
      count += 1
    while n >= 1: # Complete the while loop condition
        # Complete the body of the while loop. This should include
        # performing a calculation and incrementing a variable in the
        # appropriate order.
        n = n/10
        count += 1
    return count

print(digits(25))   # Should print 2
print(digits(144))  # Should print 3
print(digits(1000)) # Should print 4
print(digits(0))    # Should print 1
```

Run
Reset

```
2
3
4
1
```

4. Fill in the blanks to complete the "rows_asterisks" function. This function should print rows of asterisks (*), where the number of rows is equal to the "rows" variable. The number of asterisks per row should correspond to the row number (row 1 should have 1 asterisk, row 2 should have 2 asterisks, etc.). Complete the code so that "row_asterisks(5)" will print: **1 point**

```
*

* *

* * *

* * * *

* * * * *
```

```
def rows_asterisks(rows):
    # Complete the outer loop range to control the number of rows
    for x in range(1, rows+1):
        # Complete the inner loop range to control the number of
        # asterisks per row
        for y in range(x):
        #     # Prints one asterisk and one space
            print("*", end=" ")
        # An empty print() function inserts a line break at the
        # end of the row
        print()


rows_asterisks(5)
# Should print the asterisk rows shown above

```

Run

Reset

```
*
* *
* * *
* * * *
* * * * *
```

5. Fill in the blanks to complete the "countdown" function. This function should begin at the "start" variable, which is an integer that is passed to the function, and count down to 0. Complete the code so that a function call like "countdown(2)" will return the numbers "2,1,0".

1 point

```
def countdown(start):
    x = start
    if x >= 0:
        return_string = "Counting down to 0: "
        while x >= 0: # Complete the while loop
            return_string += str(x) # Add the numbers to the "return_string"
            if x >= 0:
                return_string += ","
            x -= 1 # Decrement the appropriate variable
    else:
        return_string = "Cannot count down to 0"
    return return_string


print(countdown(10)) # Should be "Counting down to 0: 10,9,8,7,6,5,4,3,2,1,0"
print(countdown(2)) # Should be "Counting down to 0: 2,1,0"
print(countdown(0)) # Should be "Cannot count down to 0"

```

Run

Reset

```
Counting down to 0: 10,9,8,7,6,5,4,3,2,1,0,
Counting down to 0: 2,1,0,
Counting down to 0: 0,
```

6. Fill in the blanks to complete the "odd_numbers" function. This function should return a space-separated string of all odd positive numbers, up to and including the "maximum" variable that's passed into the function. Complete the for loop so that a function call like "odd_numbers(6)" will return the numbers "1 3 5".

1 point

```
def odd_numbers(maximum):

    return_string = "" # Initializes variable as a string

    # Complete the for loop with a range that includes all
    # odd numbers up to and including the "maximum" value.
    for x in range(0,maximum+1):
        if x % 2 != 0 and x > 0 :
        # Complete the body of the loop by appending the odd number
        # followed by a space to the "return_string" variable.
            return_string += str(x) + " "

    # This .strip command will remove the final " " space
    # at the end of the "return_string".
    return return_string.strip()


print(odd_numbers(6))  # Should be 1 3 5
print(odd_numbers(10)) # Should be 1 3 5 7 9
print(odd_numbers(1))  # Should be 1
print(odd_numbers(3))  # Should be 1 3
print(odd_numbers(0))  # No numbers displayed

```

Run

Reset

```
1 3 5
1 3 5 7 9
1
1 3
```

7. The following code is supposed to add together all numbers from x to 10. The code is returning an incorrect answer, what is the reason for this?

1 point

```
x = 1
sum = 5
while x <= 10:
    sum += x
    x += 1
```

```python
print(sum)
# Should print 55
```

- ○ The code is not inside of a function
- [✓ The "sum" variable is initialized with the wrong value]
- ○ Not incrementing the iterator (x)
- ○ Should use a for loop instead of a while loop

8. How many numbers will this loop print? Your answer should be only one number.

1 point

```python
for sum in range(5):
    sum += sum
    print(sum)
```

5

9. What is the initial value of the “outer_loop” variable on the first iteration of the nested "inner_loop"? Your answer should be only one number.

1 point

```python
for outer_loop in range(2, 6+1):
    for inner_loop in range(outer_loop):
        if inner_loop % 2 == 0:
            print(inner_loop)
```

2

10. The following code causes an infinite loop. Can you figure out what’s missing and how to fix it?

1 point

```python
def count_numbers(first, last):
  # Loop through the numbers from first to last
  x = first
  while x <= last:
    print(x)


count_numbers(2, 6)
# Should print:
# 2
# 3
# 4
# 5
# 6
```

- ○ Missing the break keyword
- [✓ Variable x is not incremented]
- ○ Missing an **if-else** block
- ○ Wrong comparison operator is used

Upgrade to submit