

Data 8 Summer 2021

Discussion: Iteration and Conditionals (Discussion 06)

Welcome to Discussion 6! This week we will be discussing conditional statements and iteration, which are powerful computational tools that we will use throughout the course. Conditional statements allow data scientists to make more complex decisions with their code, while for loops allow us to repeat the same action many times.

Question 1. What does the following function do? Fill out the docstring description for the function to include what the inputs should be and what the function does. *Hint: try to figure out what the function would do on different inputs.*

```
def mystery_function(n1, n2):  
    """  
    if n2 - n1 > 0:  
        return n2 - n1  
    elif n2 - n1 < 0:  
        return n1 - n2  
    else:  
        return 0  
    """
```

Question 2. The instructor of a lower division statistics class has assigned you a task: make a function that takes in a student's score on a scale from 0 to 100 and assigns a letter grade based on the following grade boundaries.

Score	Letter Grade
0 - 69	F
70- 79	C
80 - 89	B
90 - 100+	A

Complete the function `compute_letter_grades`. It takes in a student's score and returns the letter grade they should receive.

```
def compute_letter_grades(score):  
    """  
    compute_letter_grades(10)  
    >>> "F"  
    compute_letter_grades(99)  
    >>> "A"  
    """  
    if _____:  
        return _____  
    elif _____:  
        return _____  
    elif _____:  
        return _____  
    else:  
        return _____
```

Question 3. Skeleton code for the function `count_evens` is below. The function takes in an array of numbers and returns the number of even numbers in the array.

a. If a number `n` is odd, what will `n%2` return:

Hint: the `%` operator returns the remainder if you divide by a certain number! Example: `11 % 5 = 1`

b. Use a combination of iteration and conditionals to complete the function below.

```
def count_evens(n_array):  
    num_evens = _____  
  
    for _____:  
        if _____:  
            _____  
  
    return _____
```

c. Use array operations to complete the function below.

```
def count_evens(n_array):  
    remainder_array = _____  
    return _____
```

Question 4. Complete the function `separate_numbers`, which takes in an array of numbers and a boolean value. It should return the number of even values in the array if the argument `return_even` is True, or the number of odd values in the array if `return_even` is False.

Hint: Use the `count_evens` function you defined above!

```
def separate_numbers(n_array, return_even):  
    num_evens = _____  
    if _____:  
        return _____  
    else:  
        return _____
```

Question 5. Match the following table method calls to the resulting descriptions of tables:
The table `chocolates` is listed here:

Color	Shape	Amount	Price (\$)
Dark	Round	4	1.30
Milk	Rectangular	6	1.20
White	Rectangular	12	2.00
Dark	Round	7	1.75
Milk	Rectangular	9	1.40
Milk	Round	2	1.00

And the nutrition table is listed here:

Type	Calories
Dark	120
Milk	130
White	115
Ruby	120

Letter	Function Call
A	<code>chocolates.group("Shape")</code>
B	<code>chocolates.group("Shape", max)</code>
C	<code>chocolates.pivot("Color", "Shape")</code>
D	<code>chocolates.group(["Shape", "Color"], max)</code>
E	<code>chocolates.pivot("Color", "Shape", "Price", max)</code>
F	<code>chocolates.join("Color", nutrition, "Type")</code>
G	<code>chocolates.group(["Shape", "Color"])</code>

Number	Columns	# of Rows
1	Shape, Color max, Amount max, Price (\$) max	2
2	Shape, Dark, Milk, White	2
3	Shape, Color, Price max, Amount max	4

4	Color, Shape, Amount, Price (\$), Calories	6
5	Shape, count	2
6	Shape, Color, count	4
7	Shape, Dark, Milk, White	2

A:
B:
C:
D:
E:
F:
G: