Lab 05: Iterations, Conditionals

Data 8 Discussion Worksheet

Welcome to the Lab 5 Discussion Worksheet! 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.

1. What does the following function do? Write a sentence below describing what the function's inputs should be and what the function does.
Hint: Try out the function on a few different inputs and see what happens!

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

`The function takes in two different numbers and computes the absolute difference between them.`

2. The instructor of a lower division statistics class has assigned you a task: define 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_grade`. It takes in a student's score and returns the letter grade they should receive. Assume that all scores are integers (i.e. you cannot have scores with decimals).

```
def compute_letter_grade(score):
     """Takes a numerical score and returns the corresponding letter
     grade"""
     if ______________________________:
          return ______________________________
     elif ____________________________:
          return ______________________________
     elif ____________________________:
          return ______________________________
     else:
          return ____________________________

     if score <= 69:
          return "F"
     elif score <= 79:
          return "C"
     elif score <= 89:
          return "B"
     else:
          return "A"

     # We could also do it this way!

     if score > 89:
          return "A"
     elif score > 79:
          return "B"
     elif score > 69:
          return "C"
     else:
          return "F"
```

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. The `%` operator returns the remainder if you divide by a certain number. (e.g. `11%5=1`) If a number `n` is odd, what will `n%2` return?

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

```
def count_evens(n_array):
    num_evens = 0

    for _____________________________________:
      if _____________________________________:
           _____________________________________

    return _____________________________________
```

```
def count_evens(n_array):
    num_evens = 0
    for num in n_array:
     if num % 2 == 0:
         num_evens = num_evens + 1
    return num_evens
```

c. Use array operations to complete the function below.

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

```
def count_evens(n_array):
    remainder_array = n_array % 2
    return np.count_nonzero(remainder_array == 0)

# We could also do it this way!

def count_evens(n_array):
    remainder_array = n_array % 2
    return len(remainder_array) - sum(remainder_array)
```

4. 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	`chocolates.group(“Shape”)`
B	`chocolates.group(“Shape”, max)`
C	`chocolates.group([“Shape”, “Color”], max)`
D	`chocolates.pivot(“Color”, “Shape”, “Price($)”, max)`
E	`chocolates.join(“Color”, nutrition, “Type”)`
F	`chocolates.group([“Shape”, “Color”])`

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

A.
B.
C.
D.
E.
F.

A-5: group with no function yields a table back with just the group column and count.
B-1: group with a function yields back the group column and the other columns with the function name added at the end.
C - 3: group with multiple columns and a function gives you those columns and the remaining columns with the function added
D-2: pivot yields the group column and then the unique values from the pivot column as column names
E-4: join gives you all the columns from the two tables except for the extra column that's joined in
F-6: group with multiple columns and no argument gives you those columns back and a count column