Aruna Meshram

Task 4 - Iteration

Course: DfE - Data Science (Fundamentals)

Reviewed By: Abodunde Ojo

Date Submitted: March 24, 2024, 5:28 p.m.

Date Reviewed: March 29, 2024, 11:38 a.m.

Student number: AM24010013629

Scores

Completeness: 4 / 4

Efficiency: 4 / 4

Style: 4 / 4

Documentation: 4 / 4

Positive

Excellent submission overall, Aruna!

In your while.py file, you have implemented the code appropriately and have demonstrated a good understanding of the key objectives of this task.
The code you've provided accomplishes the task of continually asking the user to enter a number until they input "-1," which then calculates the average of the entered numbers (excluding the "-1"). Using the inequality sign as well as the break syntax to break the loop is very well done. That is a good understanding of the task at hand as well as the use of the while loop.

You have demonstrated a good understanding of python arithmetic and comparison operators. Generally, using comments is beneficial for explaining the purpose of sections and providing clarity and you have done a good job in integrating comments to your work. This is a good practice that you should keep up with.

Likewise, in you pattern.py file, you have implemented the code appropriately and have demonstrated an advanced understanding of the for loop syntax in python! Your code has achieved the aim of outputting a star pattern using only one for loop statement! You have achieved this by using the combination of conditional blocks (if-else statement) as well as arithmetic and comparison operator (*, <=) , and it seems to be working correctly. Excellent work all round!

Improve

If you're aiming for a more concise and efficient way to achieve the same triangular star pattern, you can use a single loop with conditional statements to determine the number of stars to print on each line.
Here are some alternatives:

```
rows = 5 # Adjust the number of rows as needed

for x in range(rows * 2):
num_stars = x + 1 if x < rows else 2 * rows - x - 1
print("* " * num_stars)
```

OR

```
for x in range(10):
num_stars = x if x < 5 else 2 * 5 - x
print("* " * num_stars)
```

OR

```
for x in range(10):
print( x * '* ' if x < 5 else (2 * 5 - x) * '* ')
```

This eliminates the need for nested loops and uses a single loop to determine the number of stars to print on each line. The num_stars variable is calculated based on the conditions, and the result is then used to print the corresponding number of stars on the current line. This approach can be more concise and may be considered more efficient in terms of code readability and simplicity.

You can aim to expand your understanding of the for loop by visiting
https://www.dataquest.io/blog/tutorial-advanced-for-loops-python-pandas/

Overall

This was awesome, Aruna. I'm looking forward to the next submission!