Discussion-4 Python Loops

Discussion Topic:

What are the key constructs required to create a loop? Identify two scenarios that may require two different types of loops. Be sure to provide specific details for each scenario that illustrate why the different types of loops are required. Include a brief pseudocode example to share that includes at least one loop. Actively participate in this discussion by providing constructive feedback on the scenarios and details posted by your peers.

Be sure to post an initial, substantive response by Thursday at 11:59 p.m. MT, and respond to two or more classmates or the instructor with substantive responses by Sunday at 11:59 p.m. MT.

A substantive initial post answers the question presented completely and/or asks a thoughtful question pertaining to the topic. Be sure your post is unique. Do not repeat what other students have said.

Substantive peer responses ask thoughtful questions pertaining to the topic, and/or answer a question (in detail) posted by another student or the instructor. Note: The following are not examples of substantive contributions:

Thanking, agreeing with, or complimenting a classmate.
Providing irrelevant commentary.

My Post:

Hello Class,

Loops are a great tool in programming; they are a fundamental component, with recursion, to create efficient algorithms. A loop is essentially an automated task that processes collections of data or computations, which helps to manage program flow without having to write the same code over and over again.

The key constructs required to create a loop are basically initialization, condition, body, update, and termination.

1. The initialization sets the initial value (e.g., `counter=0`, `i=0`, an iterator for a collection) used in a condition check. In other words, it sets the starting point.
2. The condition is a Boolean expression used to evaluate whether the loop needs to run again (e.g., `i < n` or while `input_var == “continue”`).
3. The body is the code block that needs to be repeatedly computed by the loop.
4. The update is the process of updating the variable that is used in the Condition construct (e.g., `i+=1`).
5. The termination is the process of terminating the loop when a condition in the Conditional construct is evaluated to false, or by using a statement such as `break`, used to terminate the loop without using the condition construct.

Python has two primary loop constructs: the `for` loop and the `while` loop.
On a side note, Python does not have a `do-while` loop Guido van Rossum's (the creator of Python), rejected PEP 315 (proposed changes to Python), which proposed adding `do-while` loops to Python. He

argued that it was against the philosophy of Python as it breaks the “Zen of Python” (PEP 20) principles, which state that Python code needs to be simple, readable, and explicit, and a `do-while` loop would undermine PEP 20 for minimal gain, as the `do-while` loop functionality can be implemented using the `while` loop.

The for loop and the `while loop` use the constructs list previously. However, they differ on how iteration is applied. For example, the `for loop` can iterate over an iterable data type (such as a list, range, or file), where the number of elements, or the source size of the elements, is known and can be handled implicitly by the iterator. On the other hand, the `while loop` is more bound to its Boolean condition (being true); this is better suited for situations where the number of iterations is unknown (e.g., waiting for user input, network state). Nonetheless, both loops can be used to iterate over data, and when the number of elements is unknown or at what point the set condition will return false. They are interchangeable and can accomplish the same tasks, with one being more efficient than the other, depending on the nature of the task. Below are two scenarios illustrating the use of each loop:

This scenario is a collection processing using a for-each loop - `for loop`.
The example is an average positive number calculator with a validator (non-None, non-negative) reading

```
readings = [21.0, None, 20.5, -1.0, 22.3, 19.9]

total = 0.0
count = 0

for r in readings:
    if r is not None and r >= 0:
        total += r
        count += 1

avg = (total / count) if count > 0 else None
print("Average:", avg if avg is not None else "not a valid data")
```

This scenario menu usage - `while` loop.
This example is a simple menu that ends when the user types 'q'

```
user_choice = ""

while command != "q":  # condition-only termination

    print("\nMenu: [add] [list] [q=quit]")
    user_choise = input("Select: ").strip().lower()

    if user_choise == "add":
        print("Added item.")
    elif user_choise == "list":
```

```
        print("Listing items...")
    elif user_choise == "q":
        print("Goodbye!")
    else:
        print("Invalid option.")
```

The same example, but simulating a `do-while` loop, is my preferred choice for implementing menus.

```
while True:  # Simulates a do-while loop

    print("\nMenu: [add] [list] [q=quit]")
    user_choise = input("Select: ").strip().lower()

    if user_choise == "add":
        print("Added item.")
    elif user_choise == "list":
        print("Listing items...")
    elif user_choise == "q":
        print("Goodbye!")
        break
    else:
        print("Invalid option.")
```

-Alex