Discussion-4: Python Repetition

Discussion Topic:

What are the key constructs required to create loops in Python? 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. Actively participate in this discussion by providing constructive feedback on the scenarios and details posted by your peers. Provide at least one reference to support your findings.

Select only one of the following questions and offer a good, substantive reply. Some repetition is inevitable, but please aim to reply to a question that has not already been answered. Please include the question number you respond to, so we all can better understand your post, such as “#1”, “#2”, etc.:

1. Most algorithms in programming have within them some looping mechanism... Let's start with the basics: what is a loop?
2. Which are the statements required to create loops in Python? How do they work? Why have several and not just one?

This week we investigate the repetition control structure, also called a "loop" in programming. Loops allow programmers to provide iterative commands that execute a specific number of times or until a condition is met. Loops are an essential component of nearly any computer program!

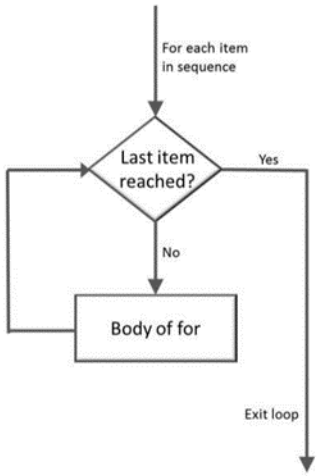
My Post:

Hello class,

For this discussion, I chose question #2, What is "Which are the statements required to create loops in Python? How do they work? Why have several and not just one?"
In Python, the major statements required to create loops are for and while.
The for statement is mostly used to iterate over iterable objects (such as a string, tuple or list). Additionally, like other coding languages (Python Software Foundation (a), n.d.). The 'while' loop, on the other hand, is used for repeated execution as long as an expression is true. (Python Software Foundation (b), n.d.).

In other words, both the 'for' and the 'while' loops are algorithmic, meaning they perform repetitive tasks until a condition is met or a condition remains true. To be more specific, the 'for' iterates over sequences executing a set of instructions until a condition is met, for example, until the end of the sequence is reached. In comparison, the 'while' will execute a set of instructions as long a condition is true. The loops complement each other and when nested within each other they can be a powerful tool for solving complex problems. This is the main reason Python has more than one loop statement.

The 'for' statement
The 'for' statement goes through each item in the sequence or iterable, one by one, and executes the block of code for each element. The flow chart below depicts the algorithmic nature of the 'for' loop.
Figure 1
The 'for' loop



Note: *from 4.3 For Loops in Python, by Colorado State University Global (2024a)*

A scenario of iterating over a sequence using a 'for' loop could be similar to the following:

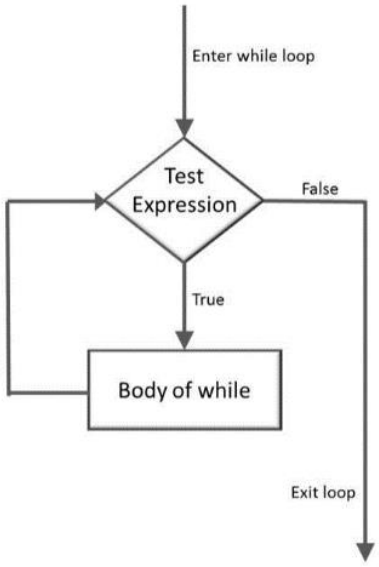
```
user_ids = [101, 102, 103, 104]
for user_id in user_ids:
        print (user_id)
```

The ‘while’ statement

The ‘while’ statement, before each iteration, evaluates the condition; if the condition is true, the loop's body is executed. If the condition becomes false, the loop stops. The flow chart below depicts the algorithmic nature of the ‘while’ loop.

Figure 2

The ‘while’ loop



Note: *from 4.2 While Loops in Python, by Colorado State University Global (2024b)*

A scenario of iterating of using a ‘while’ loop as long a condition is true could be similar to the following:

```
coffee = 0
homework_num = 100
while coffee < 100:
   coffee += 1
   print(f"Drinking coffee number {coffee} ...")
   if coffee < 100:
      print(f"Doing homework {homework_num } …")
      homework_num -= 1
      if homework_num == 0:
         break
    else:
         print("Rest in peace")
```

The ‘break’ will exit the loop. The ‘break’, ‘continue’, ‘pass’, and ‘else’ statements can be used in conjunction with loops to control their execution.

- The ‘break’ statement is used within loops to exit the loop.
- The ‘continue’ statement allows the loop to skip the rest of its code block and proceed directly to the next iteration.

- The ‘pass’ statement acts as a placeholder and does nothing really. It is often used by programmers as a placeholder to bypass blocks of code that are under construction or not yet implemented.
- The ‘else’ statement executes a block of code after the loop completes normally. In other words, the code within the ‘else’ block runs only if the loop is not terminated by a ‘break’ statement.

For example, the ‘while’ loop example could be rewritten as follows:

```
coffee = 0
homework_num = 100
while coffee < 100:
   coffee += 1
   print(f"Drinking coffee number {coffee} ...")
   if coffee < 100:
      print(f"Doing homework {homework_num } …")
      homework_num -= 1
      if homework_num == 0:
         break
 else:
    print("Rest in peace")
```

Here the ‘else’ statement is part of the ‘while’ loop, the code within the ‘else’ would be executed if the loop is not terminated by the ‘break’ statement. In this case, the code within the ‘else’ statement will run.

The examples above
-Alex

References:

Colorado State University Global (2024a) 4.3 For Loops in Python. Module 4: Python. Repetition. In ITS320: Basic Programming. https://csuglobal.instructure.com/courses/88479/pages/4-dot-3-for-loops-in-python?module_item_id=4620803

Colorado State University Global (2024b) 4.2 While Loops in Python. Module 4: Python. Repetition. In ITS320: Basic Programming. https://csuglobal.instructure.com/courses/88479/pages/4-dot-2-while-loops-in-python?module_item_id=4620802

Python Software Foundation (a). (n.d.). 4. More Control Flow Tools. *The Python Tutorial*. python.org. https://docs.python.org/3/tutorial/controlflow.html#index-0

Python Software Foundation (b). (n.d.). 8. Compound statements. *The Python Language Reference*. python.org. https://docs.python.org/3/tutorial/controlflow.html#index-0