Discussion-3: Understanding Python Decision Control Structure

Discussion Topic:

There are various selection structures that can be used in programming. What are the different relational operators used in selection structures in the Python programming language? Also, explain what short-circuiting is with respect to compound conditional expressions in Python. Provide examples to support your response in addition to constructive feedback on the structures and circumstances 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. What is the Python "if-else" statement good for? How does it work? How is it written in detail?
2. What is the importance of code indentation in Python?
3. What is a "relational/logical operator"? What are they used for?
4. What is a "membership operator"? What are they used for?
5. What is "short-circuiting" concerning compound conditional expressions in Python?

This week we study decision statements -- the IF statement. In nearly any program, we will need to make decisions based on some data, or some user input. To implement the decision, we will use Python's IF statement and its accompanying conditional and logical expressions -- this week's discussions will be fun! Just pick one of the questions and let's go.

My Post:

Hello class,

For this discussion, I choose question #5, What is "short-circuiting" concerning compound conditional expressions in Python?
To understand the concept of short-circuiting in compound conditional expressions in Python, it is important to be familiar with the logical operators and and or.
The table below summarizes the logical outcomes for these operators.

Table 1

The 'and' and 'or' Operators

A	B	A AND B	A OR B
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Note: *From Module 3: Understanding Python decision control structure, ITS320: Basic Programming, by Colorado State University Global, 2024. Modified 2024, February 25.*

In Python, short-circuiting in the context of compounded conditional expressions is when the interpreter stops evaluating a logical expression as soon as the logical expression outcome is determined (Severance, 2016).
In other words, when in the process of reading a logical expression, if the interpreter can determine the outcome of the expression before reaching the end of it, it would stop reading the expression.
Note: the interpreter reads from left to right.

This occurs when using the operators and and or in an expression. This is called a short-circuit boolean evaluation. (Hrehirchuk et al, 2024)

For example:

When using the `and` operator:

```
a = 1
b = 2
c = 3
d = 4

if a < b and a > c and a < d:
        #--- Do something
```

Here the short-circuiting happens when the Python interpreter stops evaluating the logical expression `a < b and a > c and a < d` at step `a > c` because `a > c` returns `False`. Consequently, the expression `a < b and a > c and a < d` is `False`, it does not matter if the expression `a < d` returns `False` or `True`.

When using the `or` operator:

```
a = 1
b = 2
c = 3
d = 4

if a > b or a < c or a > d:
        #--- Do something
```

Here the short-circuiting happens when the Python interpreter stops evaluating the logical expression `a > b or a < c or a > d` at step `a < c` because `a < c` returns `True`. Consequently, the expression `a > b or a < c or a > d` is `True`, it does not matter if the expression `a > d` returns `False` or `True`.

When using a combination of `and` and `or` logical operators, the `and` operator has precedent over the `or` operator. This is similar to the arithmetic operator precedence between '+' and '*', where '*' has precedence over '+'.
The table below depicts the logical operators' precedence utilizing parentheses

Table 2
Precedence of Logical Operators

A or B and C	means	A or (B and C)
A and B or C and D	means	(A and B) or (C and D)
A and B and C or D	means	((A and B) and C) or D
!A and B or C	means	((!A) and B) and C

Note*: from Chapter 40 Boolean Expressions and Short-Circuit Operators - Precedence of Logical Operators, by Kjell, n.d. Modified 2024, February 25.*

-Alex

References:

Colorado State University Global (2024). Module 3: Understanding Python decision control structure, ITS320: Basic Programming. https://csuglobal.instructure.com/courses/88479/pages/module-3-overview?module_item_id=4620787

Hrehirchuk, M., Chalmers, E., Curtis, C., & Perri, P. (2024, January 30). *5.11 Short-circuit evaluation. Foundations of Python programming: Functions first*. Runestone Academy. https://runestone.academy/ns/books/published/foppff/conditionals_short-circuit-evaluation.htmlLinks to an external site.

Kjell, B.(n.d.). Chapter 40 Boolean expressions and short-circuit operators - Precedence of logical operators. *Introduction to computer science using Java*. Central Connecticut State University. https://chortle.ccsu.edu/java5/Notes/chap40/ch40_16.htmlLinks to an external site.

Severance, C. (2016, July 5). 4.8. Short-circuit evaluation of logical expressions. *Python for everybody - -interactive*. Runestone Academy. https://runestone.academy/ns/books/published/py4e-int/conditional/shortCircuit.html