Activity 3: Decisions and Loops

Computer programs make decisions based on logic: if some condition applies, do something, otherwise, do something else.

Content Learning Objectives

After completing this activity, students should be able to:

- Evaluate boolean expressions with comparison operators (<, >, <=, >=, ==, !=).
- Explain the syntax and meaning of `if/else` statements and indented blocks.
- Evaluate boolean expressions that involve comparisons with `and`, `or`, and `not`.

Process Learning Objectives

After completing this activity, students should make progress toward:

- Evaluating complex logic expressions based on operator precedence. (Critical Thinking)

Model 1 Comparison Operators

In Python, a comparison (e.g., `1 < 2`) will yield a ***Booelan*** value of either `True` or `False`. (Note: the capitalization of the first letter is required.) Most data types (including `int`, `float`, `str`, `list`, and `tuple`) can be compared using the following operators:

Operator	**Meaning**
`<`	less than
`<=`	less than or equal
`>`	greater than
`>=`	greater than or equal
`==`	equal
`!=`	not equal

Type the following code, one line at a time, into a Python Shell. Record the output for each line (if any) in the second column.

Python code	**Shell Output**
`type(True)`	
`type(3 < 4)`	
`print(3 < 4)`	
`three = 3`	
`four = 4`	
`print(three == four)`	
`check = three > four`	

`print(check)`	
`type(check)`	
`print(three = four)`	
`three = four`	
`print(three == four)`	

Questions (10 min)

start time:

1. For each of the following terms, identify examples from the table in Model 1:
   a. Boolean variables:

   b. Boolean operators:

   c. Boolean expressions:

2. Explain why the same expression `three == four` had two different results.

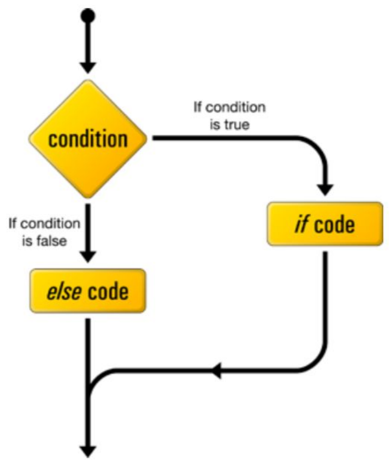
3. What is the difference between the = operator and the == operator?

4. Write a Boolean expression that uses the `!=` operator and is `False`.

Model 2 `if/else` Statements

An `if` statement makes it possible to control what code will be run in a program, based on a condition. For example:

```
number = int(input("Enter an integer: "))
if number < 0:
    print(number, "is negative")
else:
    print(number, "is a fine number")
print("Until next time...")
```



Python uses ***indentation*** to define the structure of programs. The line indented under the `if` statement is executed only when `number < 0` is `True`. Likewise, the line indented under the `else` statement is executed only when `number < 0` is `False`. The flowchart on the right illustrates this behavior.

Questions (15 min)

start time:

5. What is the Boolean expression in Model 2?

6. Enter this short program into a Python Editor. What is the output when the user enters the number 5? What is the output when the user enters the number -5?

7. After an if-condition, what syntax is different between (1) statements that are ran based on the condition and (2) statements that are always executed?

8. Enter the line `␣␣␣␣print("Hello")` into a Python Shell (where ␣ is a space). What happens if you dont indent code the same?

9. Based on the program in Model 2, what must each line preceding an indented block of code end with?

10. Does an `if` statement always need to be followed by an `else` statement? Why or why not? Given an example.

Model 3 Boolean Operations

Expressions may include Boolean operators to implement basic logic. If all three operators appear in the same expression, Python will evaluate `not` first, then `and`, and finally `or`. If there are multiple of the same operator, they are evaluated from left to right.

Do not type anything yet! Read the questions first!

Python code	Predicted Output	Actual Output
`print(a < b and b < c)`		
`print(a < b or b < c)`		
`print(a < b and b > c)`		
`print(a < b or b > c)`		
`print(not a < b)`		
`print(a > b or not a > c and b > c)`		

Questions (20 min)

start time:

11. What data type is the result of `a < b`? What data type is the result of `a < b and b < c`?

12. Predict the output of each print statement, based on the variables `a = 3`, `b = 4`, and `c = 5`. Then execute each line in a Python Shell to check your work.

13. Based on the variables in #14, what is the value of a < b? What is the value of b < c?

14. If two `True` Boolean expressions are compared using the `and` operator, what is the resulting Boolean value?

15. Using the variables defined in #14, write an expression that will compare two `False` Boolean expressions using the `or` operator. Check your work using a Python Shell.

16. Assuming P and Q each represent a Boolean expression that evaluates to the Boolean value indicated, complete the following table. Compare your team's answers with another team's, and resolve any inconsistencies.

`P`	`Q`	`P and Q`	`P or Q`
`False`	`False`		
False	True		
True	False		
True	True		

17. Assume that two Boolean expressions are compared using the `and` operator. If the value of the first expression is `False`, is it necessary to determine the value of the second expression? Explain why or why not.

18. Assume that two Boolean expressions are compared using the `or` operator. If the value of the first expression is `True`, is it necessary to determine the value of the second expression? Explain why or why not.

19. Examine the last row of the table in #14. Evaluate the Boolean expression following the order of precedence rules explained in Model 3. Show your work by rewriting the line at each step and replacing portions with either `True` or `False`.

a > b or not a > c and b > c

20. Suppose you wanted to execute the statement `sum = x + y` only when both `x` and `y` are positive. Determine the appropriate operators, and write a single Boolean expression for the if-condition.

21. Rewrite the expression from #22 using the `not` operator. Your answer should yield the same result as in #22, not the opposite. Describe in words what the new expression means.

22. Suppose that your team needs to execute the statement `sum = x + y` except when both `x` and `y` are positive. Write a Boolean expression for this condition. How is it different from the previous question?

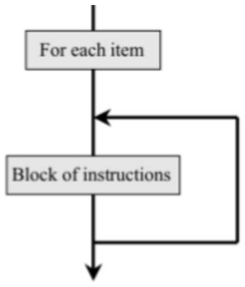
Loops

Model 1 `for` Statements

A `for` loop will execute the same block of code ``for each item in a sequence''. Create a new file named `loops.py`, and enter the following code:

```
def model_one():
    for x in [2, 7, 1]:
        print("the number is", i)
    print("goodbye")

model_one()
```



Questions (15 min)

start time:

1. When you execute the function, how many times does the indented line of code happen inside the `for` loop?

2. How many times does the line of code NOT indented happen after the for loop?

3. Identify the value of `x` each time the indented line of code is executed.

   a. 1st time:

   b. 2nd time:

   c. 3rd time:

4. Modify the list `[2, 7, 1]` in the following ways, and rerun the program each time. Indicate how many times the `for` loop executes.
   a. non-consecutive numbers: `[5, -7, 0]`

   b. numbers decreasing in value: `[3, 2, 1, 0]`

   c. all have the same value: `[4, 4]`

5. What determines the number of times that the loop repeats?

6. What determines the value of the variable `x`? Explain your answer in terms of what is assigned `(x = ...)` each time the loop runs.

7. Describe how to modify the function so that the loop executes five times. Check your answer by editing your `loops.py` program.

Model 2 The `range` Function

The Python `range` function will generate a list of numbers. The `range` function can take up to three numbers as arguments. Fill in the table below by typing the code into a Python Shell:

Python code	**Shell Output**
`range(5)`	
`list(range(5))`	
`x = range(3)`	
`x`	
`print(x)`	
`print(list(x))`	
`list(range(5, 10))`	
`list(range(-3, 4))`	
`list(range(4, 10, 2))`	

`for i in range(5):`
`print(i)`	

Questions (15 min)

start time:

8. Explain the difference in output between the first two lines of code (with and without the `list` function).

9. If the argument of the `range` function specifies a single number (x):
   a. What will be the first number listed?

   b. What will be the last number listed?

c. How many numbers will be in the list?

d. Use the range function to generate the numbers 0, 1, 2, 3.

10. If the argument of the `range` function specifies two numbers (x,y):
    a. What will be the first number listed?

    b. What will be the last number listed?

    c. How many numbers will be in the list?

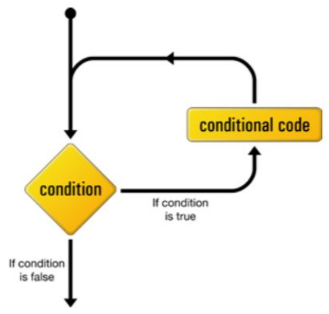
    d. Use the range function to generate the sequence 1, 2, 3, 4.

Model 3 `while` Statements

A more general looping structure is the `while` loop. Add the code below to your current `loops.py` program:

```
def model_three():
    i = 0
    while i < 3:
        print("the number is", i)
        i = i + 1
    print("goodbye")

model_three()
```



Questions (15 min)

start time:

11. What must the value of the Boolean expression (after the `while`) be in order for the first print statement to execute?

12. Circle the statement that changes the variable `i` in the above code.

13. What happens to the value of the variable `i` during the execution of the loop?

14. Explain why the loop body does not execute again after it outputs “the number is 2”.

15. Reverse the order of the statements in the loop body:
```
while i < 3:
        i = i + 1
        print("the number is", i)
```
    a. How does the order impact the output displayed by the `print` function?

    b. Does the order impact the total number of lines that are output?

16. Identify different ways to modify the code so that the loop only executes twice.

17. Describe the three parts of a `while` loop that control the number of times the loop executes.

18. Comment out the statement i = i + (To make it a comment put a # in front of the line), and run the module. Then press Ctrl-C (hold down the Ctrl key and press C). Describe the behavior you see, and explain why you think it happened.

When writing a `while` loop, it's helpful to answer a few questions before you start:
- *What needs to be initialized before the loop?*
- *What condition must be true for the loop to repeat?*
- *What will change so that the loop eventually ends?*

19. Consider the function `add(n)` that prompts the user for n numbers and returns the sum of these values. For example, when `add(5)` is called, the user is asked to input five numbers. If the user inputs 3, 1, 5, 2, and 4, the function would return the value 15.
    a. Describe the variable that needs to be initialized before the loop begins.

b. Describe the Boolean expression that must be true for the loop to continue.

c. Describe what will need to change so that the loop will eventually end.

d. Now list what else needs to happen inside the body of the loop for you to calculate the sum of the user input.

e. Given your previous answer, are there any other values that need to be initialized before the start of the loop?