Conditions and Logic

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

Manager: Recorder:

Presenter: Reflector:

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 Skill Goals

During the 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., `100 < 200`) will yield a ***Boolean*** value of either `True` or `False`. 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

Consider the following lines that were entered into a Python Shell. Write an asterisk (*) next to any row your team has questions about.

Python code	Shell output
`type(True)`	<class 'bool'>
`type(true)`	NameError
`type(3 < 4)`	<class 'bool'>
`print(3 < 4)`	True
`three = 3`	
`four = 4`	
`print(three == four)`	False
`check = three > four`	
`print(check)`	False
`type(check)`	<class 'bool'>
`print(three = four)`	TypeError
`three = four`	
`print(three == four)`	True

Questions (10 min) Start time:

1. What is the name of the data type for Boolean values?

2. Do the words `True` and `False` need to be capitalized? Explain how you know.

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

   a) Boolean variables:

   b) Boolean operators:

   c) Boolean expressions:

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

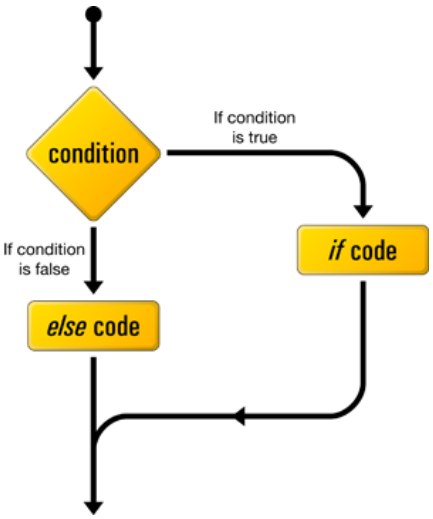
5. What is the difference between the = operator and the == operator?

6. Write a Boolean expression that uses the `!=` operator and evaluates to `False`.

Model 2 `if`/`else` Statements

`if`/`else` statements control what code will be executed, based on a condition.

```
number = int(input("Enter an integer: "))
if number < 0:
    print(number, "is negative")
else:
    print(number, "is not negative")
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:

7. What is the Boolean expression in Model 2?

8. Enter the code from Model 2 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?

9. In an `if` statement, what syntax differentiates between (1) statements that are executed based on the condition and (2) statements that are always executed?

10. Enter the line `␣␣␣␣print("Hello")` into a Python Editor (where ␣ is a space), save the file as `hello.py`, and run the program. What happens if you indent code incorrectly?

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

12. Write an `if` statement that first determines whether `number` is even or odd, and then prints the message `"(number) is even"` or `"(number) is odd"`. (Hint: use the `%` operator.)

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

Model 3 Boolean Operations

Expressions may include Boolean operators to implement 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:

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

15. 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.

16. Based on the variables in #15, what is the value of a < b? What is the value of b < c?

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

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

19. 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		

20. Assume that two Boolean expressions are combined 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.

21. Assume that two Boolean expressions are combined 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.

22. Suppose you wanted to print a result only when both `x` and `y` are positive. Determine the appropriate operators, and write a single Boolean expression for the `if`-statement condition.

23. 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.

24. Suppose that your team needs to print a result, except for when both `x` and `y` are positive. Write a Boolean expression for this condition. How is this different from the previous question?