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:

- Recognize the value of developing process skills.
- 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)

Copyright © 2021 C. Mayfield, T. Shepherd, and H. Hu. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

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

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(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:

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

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

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

a) Boolean variables:

b) Boolean operators:

c) Boolean expressions:

~~8. Explain why the same expression three == four had two different results.~~

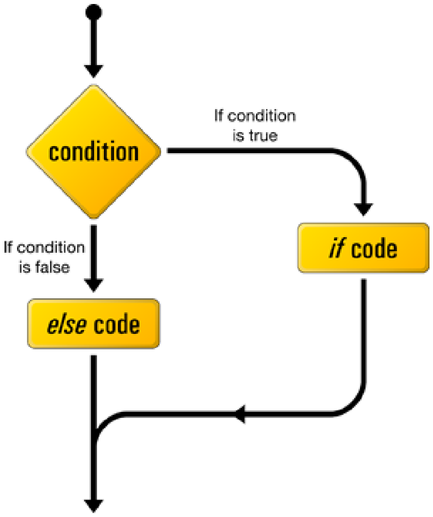
9. What is the difference between the = operator and the == operator?

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

Model 2 if/else Statements

An if statement makes it possible to control what code will be executed 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:

11. What is the Boolean expression in Model 2?

12. 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?

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

~~**14**. 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 inconsistently?~~

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

16. 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.)

17. 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 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:

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

19. 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.

20. Based on the variables in #19, what is the value of a < b? What is the value of b < c?

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

22. Using the variables defined in #19, write an expression that will combine two `False` Boolean expressions using the `or` operator. Check your work using a Python Shell.

23. 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		

24. 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.

25. 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.

26. Examine the last row of the table in #19. 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
```

27. 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.

28. Rewrite the expression from #27 using the `not` operator. Your answer should yield the same result as in #27, not the opposite. Describe in words what the new expression means.

29. 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?