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:

- Describe the responsibility of each team role.
- Evaluate boolean expressions with comparison operators (<, >, <=, >=, ==, !=).
- 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)

Meta Activity: Team Roles

Decide who will be what role for today; we will rotate the roles each week. If you have only three people, one should have two roles. If you have five people, two may share the same role.

Manager:
Presenter:
Recorder:
Reflector:

Questions (12.5 min) Start time:

1. What is the difference between **bold** and *italics* on the role cards?

2. Manager: invite each person to explain their role to the team. Recorder: take notes of the discussion by writing down key phrases on the *Recorder's Report*.

3. What responsibilities do two or more roles have in common?

4. For each role, give an example of how someone observing your team would know that a person is not doing their job well.

- Manager:
- Presenter:
- Recorder:
- Reflector:

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`, and `str`) 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 (12.5 min) Start time:

5. Manager: lead a discussion about Model 1. Recorder: summarize (on the *Recorder's Report*) what the team has learned so far. Presenter: be prepared to share one insight with the class.

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

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

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

a) Boolean variables:

b) Boolean operators:

c) Boolean expressions:

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

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

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

Model 2 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:

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

13. 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.

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

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

16. Using the variables defined in #13, write an expression that will combine two `False` Boolean expressions using the `or` operator. Check your work using a Python Shell.

17. 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		

18. Consider two Boolean expressions that 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.

19. Consider two Boolean expressions that 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.

20. 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.

21. Rewrite the expression from #20 using the `not` operator. Your answer should yield the same result as in #20, not the opposite. Describe in words what the new expression means.

22. 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?