Conditions and Logic I

Warm-up

Last class, we looked at sequences. Test your knowledge by answering the following questions:

Questions (10 min) **Start time:**

1. Evaluate each expressions in order and write down what it evaluates to. If an error occurs, write what type of error. Place an asterisk (*) next to any output for which you are unsure.

Python code	Output
`seq1 = "one two"`	
`type(seq1)`	<class 'str'>
`len(seq1)`	7
`seq1[3] = ','`	Runtime Error (str is immutable)
`seq1.replace(' ',',')`	
`print(seq1)`	`'one two'`
`seq2 = ["one", "two"]`	
`type(seq2)`	<class 'list'>
`seq2[1] = 1`	
`print(seq2)`	['one', 1]
`seq2.append(1)`	
`print(seq2)`	['one', 1, 1]
`seq2.count(1)`	2
`seq3 = 'abcdefg'`	
`seq3[:]`	'abcdefg'
`seq3[::-1]`	'gfedcba'
`seq3[:-3]`	'abcd'
`seq3[:-3:-1]`	'gf'
`seq3[-3:]`	'efg'

2. What are similarities and differences between `lists` and `strings`?

Answers may vary; they are both sequences, but are declared differently, strings with quotes while lists with brackets and commas between elements. Lists are mutable while strings are immutable.

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

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` and `list`) can be compared using the following operators:

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

Evaluate each expression in order and record the output for each line (if any) in the second column.

Python code	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
`three in range(0,17,4)`	True

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

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

Yes, because `type(true)` resulted in NameError: name 'true' is not defined.

5. For each of the following terms, identify examples from the table:

   a) Boolean variables: check

   b) Boolean operators: <, ==, >

   c) Boolean expressions: 3 < 4, three == four, three > four

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

The two variables were initially different values, so the first comparison was `False`. But later on, the value of `four` was assigned to `three`, so the second comparison was `True`.

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

The = operator assigns a value to a variable, and the == operator compares two values.

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

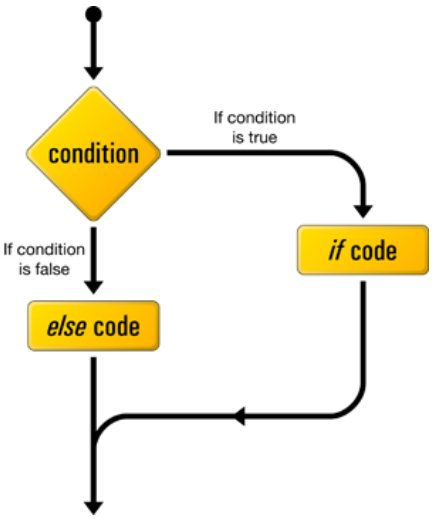
5 != 5

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:

```
def main() -> None:
    number = int(input("Enter an integer: "))
    if number < 0:
        print(number, "is negative")
    else:
        print(number, "is a fine number")
    print("Until next time...")

main()
```



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.

9. What is the Boolean expression in the code snippet above?

number < 0

10. What is the output when the user enters the number 5? What is the output when the user enters the number -5? **STOP HERE** The instructor will trace the code with you using the debugger.

```
5 is a fine number              -5 is negative
Until next time...              Until next time...
```

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

The indentation; statements that are indented under the if are based on the condition, and statements indented at the same level (later in the program) are always executed.

12. What happens if you indent code inconsistently? For example, what would happen if you entered ␣␣␣␣`print("Hello")` into a Python Editor (where ␣ is a space), save the file as `hello.py`, and run the program?

SyntaxError: unexpected indent

13. Based on the program above, what must each line preceding an indented block of code end with?

A colon.

14. 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.)

```
if number % 2 == 0:
    print(number, "is even")
else:
    print(number, "is odd")
```

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

No; you can have an if statement without an else. For example, you could determine that a number is even and print a message, without printing a different message if it's odd.

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. Evaluate the following expressions based on the variables `a = 3`, `b = 4`, and `c = 5`:

Python code	Output
`print(a < b and b < c)`	True
`print(a < b or b < c)`	True
`print(a < b and b > c)`	False
`print(a < b or b > c)`	True
`print(not a < b)`	False
`print(a > b or not a > c and b > c)`	False
`print(a not in [a,b,c])`	False
`print(a not in ['a',b,c])`	True
`print(a not in 'abc')`	TypeError (LHS: a not a str)
`print(a not in 345)`	TypeError (RHS: 345 not a sequence)

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

The type of each is `bool`; both are Boolean expressions.

17. What is the value of a < b? What is the value of b < c?

They are both true.

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

True and True is True.

19. Write an expression that will combine two `False` Boolean expressions using the `or` operator.

a > b or a > c

20. 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	False
False	True	False	True
True	False	False	True
True	True	True	True

21. 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.

It is unnecessary, because "false and anything" is false.

22. 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.

It is unnecessary, because "true or anything" is true.

23. Examine the last row of the table. Evaluate the Boolean expression following the order of precedence rules. 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
```

```
False or not a > c and b > c
False or not False and b > c
   False or True and b > c
   False or True and False
       False or False
           False
```

24. 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.

x > 0 and y > 0

25. Rewrite the expression from #24 using the `not` operator. Your answer should yield the same result as in #24, not the opposite. Describe in words what the new expression means.

not (x <= 0 or y <= 0)
In other words, "both x and y are positive" is equivalent to "neither x nor y is negative/zero".

26. 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?

not (x > 0 and y > 0)
To represent "except when" logic, we simply negate the original condition. The previous question negated each of the operators as well, which is known as De Morgan's law.