Conditions and Logic I

Warm-up

1. Write a function that take in a list and a count and returns a list with count elements picked at random from the input list (with replacement):

2. Write a function that take in a list and a count and returns a list with count elements picked at random from the input list (without replacement):

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)`	
`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)`	
`three in range(0,17,4)`	

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

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

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

   a) Boolean variables:

   b) Boolean operators:

   c) Boolean expressions:

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

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

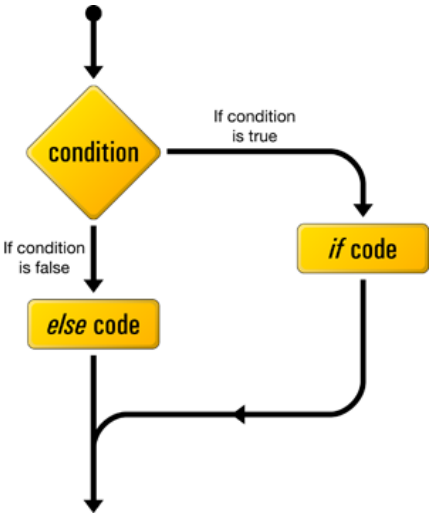
8. Write a Boolean expression that uses the `!=` operator and evaluates to `False`.

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:

```
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?

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.

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?

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?

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

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.)

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

Python code	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)`	
`print(a not in [a,b,c])`	
`print(a not in ['a',b,c])`	
`print(a not in 'abc')`	
`print(a not in 345)`	

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

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

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

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

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	True		
True	False		
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.

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.

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
```

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.

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.

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?