Set 6: Booleans

Skill 6.1: Interpret a Boolean expression

Skill 6.1 Concepts

- A ***Boolean value*** is simply a computer science-y term that means **a true/false value**.
- A ***Boolean expression*** is a statement that *evaluates* to a *Boolean value* (a single true/false).

To determine whether two values are the same or not the same, or whether one value is greater or less than another value requires a *comparison operator*.

Below is a list of comparison operators commonly used in computer science.

== is equal to != is not equal to > is greater than < is less than >= is greater than *or* equal to <= is less than *or* equal to	To the left are 6 common **comparison operators**. Each compares a value on the left with a value on the right and returns a Boolean value – **true** or **false**. Most of these do what you would expect. Why these symbols: **==**, **!=**, **<=**, and **>=**? 1. We use `==` because the single equal sign `=` is the assignment operator. We need something different to indicate we want to compare two values instead of assigning one to the other. 2. We use `!=`, `<=`, and `>=` because they only require ASCII symbols. Historically the mathematical symbols `≠`, `≤` and `≥` were hard or impossible to produce on some computer systems. The `!` is universally read as "not".

Skill 6.1 Exercise 1

Skill 6.2: Use truth tables to evaluate whether a statement is true of false

Skill 6.2 Concepts

Truth tables are helpful for showing how combinations of Boolean expressions are evaluated,

a	b	a AND b		a	b	a OR b
False	False	False		False	False	False
False	True	False		False	True	True
True	False	False		True	False	true
True	True	True		True	True	true

Skill 6.2 Exercise 1

Skill 6.3: Write Boolean statements using proper Python syntax

Skill 6.3 Concepts

The previous examples illustrate how true and false statements are evaluated, however, the syntax is not correct. In order for Python to correctly read the syntax the "AND" and "OR" statements must be replaced with the correct symbols as shown below,

Statement	True or False
((x < 10) & (y = = 6)	AND is replaced with &, the = is replaced with "= ="
((x < 10) & (y = = 5)	AND is replaced with &, the = is replaced with "= ="
((x > 10) & (y != -3)	AND is replaced with &, the ≠ is replaced with "!="
((x < 10) | (y = = 5))	OR is replaced with |, the = is replaced with "= ="
((x > 10) | (y = = 5))	OR is replaced with |, the = is replaced with "= ="
Note, that in the above examples AND can also be replaced with *and* (all lower case) and OR can be replaced with *or* (all lower case)	

Skill 6.3 Exercise 1

Skill 6.4: Apply the *not* operator

Skill 6.4 Concepts

The not (or negation) operator is used to indicate the opposite. For example, what is the opposite of true?... False of course. The "not" operator is indicated with the word "not". Below are some examples.

Code	Output
print(not True)	False
print(not False)	True
print(not 3 < 5)	False
print(not 1 == 0)	True

Skill 6.4 Exercise 1

Skill 6.5: Declare Boolean variables

Skill 6.5 Concepts

Boolean variables can be declared in Python by setting the value of the variable to either True or False. Once a Boolean variable has been created it can be combined with any of the Boolean operations previously described

Code	Output

`a = True`
`b = False`
`print(a)`
`print(b)`
`print(a == b)`
`print(not a)`
`print(not b)`
	True
	False
	False
	False
	True

The result of a comparison operation can also be stored in a variable. For example,

Code	Output

`a = 5`
`b = 10`
`result1 = a < b`
`result2 = a > b`
`result3 = a == b`
`print(result1, result2, result3, sep="\n")`
	True
	False
	False

Skill 6.5 Exercise 1

Skill 6.6: Be aware of data types when combining Boolean expressions

Skill 6.6 Concepts

When comparing variables using Boolean operations, be aware that the data types being compared must be compatible. Consider the following example,

Code	Output

`guess = input("Guess my number")`
`secret = 10`
 	
`print(guess > secret)`
	TypeError: '<' not supported between instances of 'str' and 'int'

The error occurs because c and y are different data types. To fix the error, you could convert the string portion of the comparison to an integer.

Code	Output

`guess = input("Guess my number")`
`secret = 10`
`print(int(guess) > secret)`
	True

Skill 6.6 Exercise 1

Skill 6.7: Apply operator precedence

Skill 6.7 Concepts

Just like math operations follow an order of precedence, so too do comparison and logical operators. Consider a problem like,

```
print( False and True or True )
```

Depending on which part we evaluate first, we get a different result. This is illustrated below,

Code	Output
`print( False and (True or True) )`	False
`print( (False and True) or True )`	True

As it turns out the original expression prints True because *and* takes precedence over *or*.

The following table summarizes the operator precedence from highest to lowest.

Level	Category	Operators
7(high)	exponent	**
6	multiplication	*,/,//,%
5	addition	+,-
4	relational	==,!=,<=,>=,>,<
3	logical	not
2	logical	and
1(low)	logical	or

Skill 6.7 Exercise 1