Learning Objectives

In the second module of this course, you will learn how to import Data Frames and select information from a Data Frame.

Learning Objectives:

1. Importing a CSV and loading a Data Frame
2. Visualizing a Data Frame
3. Understanding Boolean Operators

Boolean Operators

Boolean operators are operators that return a boolean value (True or False).

Equal To

Python uses the == operator to determine equality. Beginners often confuse the = and the == operators. Remember, = is the assignment operator.

```
a = 5
b = 5
print(a == b)
```

Not Equal To

The != operator checks to see if two values are not equal.

```
a = 5
b = 5
print(a != b)
```

Less Than, Greater Than, Equal to

- The < operator is used to check if one value is less than another value.
- The <= operator is used to check if one value is less than or equal to another value.
- The > operator is used to check if one value is greater than another value.
- The >= operator is used to check if one value is greater than or equal to another value.

```
a = 5
b = 5
print(a >= b)
```

The and Operator

The `and` operator allows for compound (more than one) boolean expressions. All boolean expressions **must** be true in order for the whole thing to be true. If only one boolean expression is false, then the whole thing is false.

```
a = True
b = True
c = False
print(a and b)
```

challenge

What happens if you:

- Change `print` to `print(a and c)`?
- Change `print` to `print(c and b)`?

Multiple `and` Statements

You can chain several `and` statements together. They are evaluated in a left-to-right manner.

```
a = True
b = True
c = False
print(a and b and c)
```

challenge

What happens if you:

- Change `print` to `print(a and b and a and b and a)`?
- Change `print` to `print(a and b and a and b and c)`?

The `or` Operator

The `or` operator allows for compound (more than one) boolean expressions. If only one boolean expressions is true, then the whole thing is true. To be false, all boolean expressions **must** be false.

```
a = True
b = True
c = False
d = False
print(a or b)
```

Multiple `or` Statements

Similar to `and` Statements, You can chain several `or` statements together. They are evaluated in a left-to-right manner.

```
a = True
b = True
c = False
print(a or b or c)
```

The `not` Operator

The `not` operator produces the opposite of the boolean expression that it modifies.

```
print(not True)
```

challenge

What happens if you:

- Change `print` to `print(not True and False)`?
- Change `print` to `print(not (True and False))`?
- Change `print` to `print(not not True)`?

Arithmetic Operations

The Addition (+) Operator

The addition operator works as you would expect with numbers.

```
print(7 + 3)
```

You can also add two variables together.

```
a = 7
b = 3
c = a + b
print(c)
```

---

challenge

What happens if you:

- Make `a` of type float (e.g. `a = 7.0`)?
- Make `b` a negative number (e.g. `b = -3`)?
- Make `b` an explicitly positive number (e.g. `b = +3`)

Incrementing Variables

Incrementing a variable means changing the value of a variable by a set amount. You will most often have a counting variable, which means you will increment by 1.

```
a = 0
a = a + 1
print(a)
```

Subtraction

```
a = 10
b = 3
c = a - b
print(c)
```

▼ **Subtracting a boolean?**
In Python, boolean values are more than just true and false. False has the numerical value of 0, while True has the numerical value of 1. This is why doing math with a boolean does not give you an error message.

Division

Division in Python is done with the `/` operator

```
a = 25
b = 5
print(a / b)
```

Floor Division

Normal division in Python always returns a float. If you want a whole number, use floor division (`//`). Floor division does not round up, nor round down. It removes the decimal value from the answer.

images/floor-division

```
a = 5
b = 2
print(a // b)
```

Modulo

Modulo is the mathematical operation that performs division but returns the remainder. The modulo operator is `%`. The modulo operation happens during the multiplication and division step of the order of operations.

images/modulo

```
modulo = 5 % 2
print(modulo)
```

Multiplication

Python uses the * operator for multiplication.

```
a = 5
b = 10
print (a * b)
```

Multiplication & Strings

Python allows you to multiply a string by a number.

```
a = 3
b = "Hello!"
print(a * b)
```