Introduction to Variables: Numeric Types and Booleans

Introduction

So, we know that we have a type for representing text, the String. But what if we want to represent other types of data, like Numbers? We are Data Scientists after all, and we'll be working with numbers a lot. In this lesson, we'll introduce both Numbers and Booleans, data types we will frequently use in Python.

Objectives

You will be able to:

- Use different numeric data types
- Distinguish the difference between numeric data types
- Perform basic mathematical operations with numeric data types
- Use Boolean data types

What Are Numeric Data Types?

README.md

All of us are familiar with numbers. `1492` is a number. So is `34.75`. If we think about what the common operations are with numbers, we get a pretty good idea for what Python allows us to do with numbers.

> **Note:** *to see the output of the following operations, press the shift + enter keys and run each cell*

```
1 + 1
```

```
2 * 5
```

If we look at a number for its type, we find something slightly different.

```
type(10)
```

```
type(10.2)
```

Python is simply indicating that a number without a decimal is called an `int` for ***integer***, and a number with a decimal is called a **float**. Both floats and integers are numeric data types, but for now, we can think of both of these simply as numbers.

```
10 + 0.75
```

As we can see in the above example, the `float` , `0.75` , plus the `int` , `10` , resolves to the `float` , `10.75` . So, a `float` combined with an `int` returns a number that is a `float` .

What is a Boolean?

A Boolean (`bool`) has two possible values: **True** or **False**.

```
type(True)
```

It's fairly rare for programmers to explicitly write the word `True` or `False` . Instead our programs can respond to questions for us in this form. We have already seen one such example:

```
"Homer Simpson".endswith("Simpson")
```

```
"Homer Simpson".endswith("Homer")
```

And as you might imagine, a boolean can be returned from a math operation as well.

```
3 * 5 < 10
```

You will see later on that by utilizing these returned booleans, we can make decisions with our code. For example: send this email if a user's last name is Simpson, or send an invite if the user is in a target age group. We aren't there yet, but we'll get there!

Data Types as a Choice

> "Bad programmers worry about the code. Good programmers worry about data structures and their relationships." -- Linus Torvalds

For now, it's interesting to think of how methods allow us to change between data types and to think of when we may want our data to be in one data type versus another. We started this lesson by saying that 34 is a number. But what if it's not?

```
"west 34th street"
```

We could make the argument that in certain contexts, like an address, 34 is text. In others, when we are judging distance in blocks it feels like a number. So how do we decide?

We find the answer to that by thinking about what we want to do with the data. If we want to capitalize all of the words in the string, to mail a letter (which I admit, sounds awful), we should keep the data in the format of a string.

```
"34th street".title()
```

What if we are trying to ask if a number in that string is larger than another number? For example, a restaurant that only delivers food below 22nd street might use a program to write something like:

```
34 < 22
```

But would a method like less than (`<`) work with a string? Does it make sense for a string or text to answer whether it is less than or greater than a number? Trying things is free, so let's give it a shot.

```
"34th street" < 22
```

Well, now we know for sure. So, if we want to help our restaurant with deliveries, we should convert our number from a string to a number and then make the comparison.

```
int('34') < 22
# False
```

```
False
```

And if we want to go from a number to a string, for example to produce an address, we again need to pay attention to the type.

```
str(34) + 'th Street'
```

Here we saw our first method for switching between types: simply write the name of the type followed by parentheses and the data on which we want to operate. After introducing this pattern, we can start to explore with others types such as Boolean.

```
bool(100)
```

```
bool(0)
```

Great, so we can coerce a number to a boolean as well. And we are beginning to think about keeping our data in one form or another based on what we want to do with that data.

Summary

In this section, we introduced two new types of data: numbers and booleans. We saw that numbers allow us to perform standard math operations and we saw that booleans answer whether something is True or False, and serve as a way our program or different methods can respond to questions.

We have seen almost all of our Python data types. We talked about how to choose a data type, and how to switch between data types. We said that we choose a data type based on the capabilities that we want to give to that data: should it answer whether it is larger or smaller, or does it make sense to capitalize? The goal of this discussion is to begin thinking about why we decide to put data in specific types (i.e. string, number, boolean). We also introduced coercion methods like `bool` and `str` that switch between data types.