Overview

Python has a feature called "dynamic typing" which means that variables aren't created with a specific data type in mind. A variable could start out as a number, and then change to a String, and then become equal to nothing.

The largest benefit to dynamic typing is also one of it's largest weaknesses. By being generaous with allowing different types of data to be stored in variables, writing code is beginner friendly because your code will always compile. However, some data types don't play nice together, which can make it challenging to identify bugs.

The next few lessons will introduce different variable types and give you some frameworks for using them.

Important Information

When we create a variable we always give it a name and a value. This is called **declaring** the variable. A variable cannot be used until it has been declared.

Below are variables that use 4 primitive data types in python. A primitive data type in any programming language is a basic data type. Complex data types and data structures are beyond the scope of this course, but can be useful if you take the initiative to learn them.

Boolean

A boolean variable (called just a `bool` in python) is a variable that is either `True` or `False`. It is the basis for how all math is done in computers, because it is another way of expressing binary code. With boolean values it is either `True` or `False`, and with binary it is either `1` or `0`.

Integer

An integer variable (called just an `int` in python) is a variable that stores a whole number. It can be either positive or negative.

Float

A float variable (called a `float` in python) is a variable that stores a real number (a real number is a decimal number). Float is short for "floating point" because the value of the variable is floating between 2 integers.

String

A String variable (called a str in python) is a string of characters or letters. All user input is always processed as a string, even if the user typed in numbers.

Set Up

Create a new python file called `variabletypes.py`.

Copy

```
# Boolean variables are always True or False
boolean_variable = True

# Integer variables are always an integer
integer_variable = 27

# Float variables are always a decimal
float_variable = 3.14

# String variables are always strings of text
string_variable = "Hello, World!"
```

Let's see what happens if we try and make these values interact with each other. Do each of the following one at a time, and see if any errors pop up.

Experiment 1: Integer + Integer

```
# Add 2 integers together.
print(integer_variable + integer_variable)
```

Experiment 2: Float + Float

```
# Add 2 floats together.
print(float_variable + float_variable)
```

So far so good...

Experiment 3: Float + Integer

```
# Add a float and an integer together.
print(float_variable + integer_variable)
```

...as you can see, number variables do play nice with one another.

Experiment 4: Boolean + Boolean

```
# Add a boolean and a boolean together.
print(boolean_variable + boolean_variable)
```

Not what you expected?

When we created `boolean_variable` we set the value to `True`. When we add `True` + `True` we get... `2`? What does this tell you about the way `True` is stored as a value? Try and figure out what number `False` represents.

Experiment 5: String + String

```
# Add a string and a string together.
print(string_variable + string_variable)
```

This one hopefully wasn't too surprising, but what happened when we added the strings together?

Experiment 6: Number + Boolean

```
# Add a boolean and an integer together.
print(boolean_variable + integer_variable)
```

Experiment 7: Number + String

```
# Add an integer and a string together.
print(integer_variable + string_variable)
```

Would you look at that! A `TypeError`! Most code editors are super helpful at telling you exactly what is wrong. It will show you something called a `Traceback` of the most recent call the program made before the error occured.

Here we can see that integer + string is not allowed in python and it will tell you that the operation of addition `+` is not allowed between these 2 types. You will get the same result if you try and add our boolean, float, or integer to a string.

Best Practices

Some code editors, like PyCharm, have built in type checking, which means you can use type hints. Type hints allow you to tell your coding environemnt that you want to keep a variable as the same type for the entire program. However, your code will still run even if you don't follow your hints.

If I'm being honest, using type hints in python is a personal preference of mine. I believe that it clarifies code and makes it easer to work with, so you will also have to use type hints.

Type hints use a `:` followed by the type to help the editor know what type the variable is meant to be.

```
# bool is used for booleans.
my_boolean : bool = True

# int is used for integers.
my_integer : int = 5

# float is used for floats.
my_float : float = 3.14

# str is used for strings.
my_string : str = "Hello, World!"
```