Learning Objectives

- **Learn some of the basic Python data types**
- **Create a variable**
- **Store numerical data in a list**
- **Print the data stored in variables/list**

definition

Limitations

- This section will primarily cover variables and lists. Data frames and functions will be covered in a later section.

Data Types

Data Types - Strings

Strings

A string is a collection of text, numbers, or symbols. Strings are always surrounded by quotation marks.

```
string_variable = "This is a string"
second_string = 'This is a string also'
print(string_variable)
print(second_string)
```

Notice that when you print a string, the quotation marks are not printed.

challenge

What happens if you:

Mix single (') and double (") quotation marks?

▼ **Solution:**
This causes an error because Python requires that you be consistent with quotation marks. If you start with a single quote (') you must end with a single quote. The same is true for double quotes ("). You may use either style of quotation marks, just be consistent.

Forget one of the quotation marks?

▼ **Solution:**
This causes an error because Python requires that quotation marks be used in pairs.

Forget both quotation marks?

▼ **Solution:**
This causes an error because to Python a string without quotes appears to be a series of variables that have not been defined.

Boolean

Boolean values mean `True` or `False`. You will see how boolean values are used when we talk about conditionals and while loops.

```
boolean_variable = True
print(boolean_variable)
```

challenge

What happens if you:

- Change the variable to `False`?
- Change the variable to `true`?
- Change the variable to `false`?

Integers

Integers (often called ints) are whole numbers. They can be positive or negative. Do not use a comma when typing large numbers

▼ **5 vs. '5'**
`5` is not the same thing as `'5'`. The first one is an integer, the second is a string. You will see in a later lesson the different operations you can perform on strings and numbers. Treating a string as a number can cause errors.

```
integer_variable = 50
print(integer_variable)
```

challenge

What happens if you:

- Change the variable to `5000`?
- Change the variable to `5,000`?
- Change the variable to `050`?

Floating Point numbers

Floating point numbers (often called floats) are numbers with a decimal. They can be positive or negative.

```
float_variable = 50.0
print(float_variable)
```

challenge

What happens if you:

- Change the variable to `50.`?
- Change the variable to `.001`?

▼ **What does `type` mean?**

The `type` command returns the data type of the value stored in a variable. Python abbreviates these types: `int` is an integer, `float` is a floating point number, `str` is a string, and `bool` is a boolean.

```
a = 3
print(type(a))
```

List

What is a List?

Lists are a built-in data structure that groups information together. Lists do not have to be of the same data type, but often are. Lists are declared with the `[]` brackets, and a comma separates each item in a list. Lists are mutable, which means you can alter them in a variety of ways. Effective data-driven science and computation requires understanding how data is stored and manipulated. Variables and List are some of the keys way we store and manipulate data.

```
int_list = [1, 2, 3, 4, 5]
string_list = ["John", "Paul", "George", "Ringo"]
mixed_list = [0.87, "hello", True, 17]

print(type(int_list))
print(int_list)
print(string_list)
print(mixed_list)
```

▼ **The Empty List**

There is a special list called an empty list. This is a list that has no elements. An empty list looks like this: `my_list = []`. We will see how to add elements to an empty list in a later lesson.

Populating a List

You can use the `range` function to create a sequence of numbers for a list. The syntax is slightly different from a for loop. There is an extra `i` before the for loop.

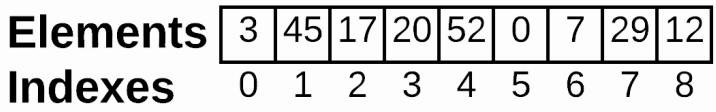
```
my_list = [i for i in range(1, 51)]

print(my_list)
```

List and its Index

Each piece of data in a list is called an element. You can access elements of a list with an index, which is like an address for each element. Start counting with index 0. The first element in the image below would be

`my_list[0]`.



images/list-and-index

```
my_list = [5, 10, 15, 20]

print(my_list[0])
```

challenge

What happens if you:

- Change the index to 2 : `print(my_list[2])`?
- Change the index to 4 : `print(my_list[4])`?
- Change the index to -1 : `print(my_list[-1])`?
- Change the index to 1.5 : `print(my_list[1.5])`?