1. Tabular Data

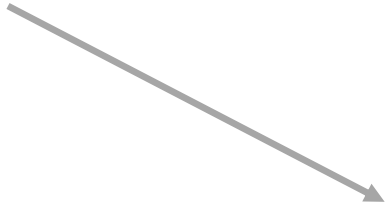
Tabular Data

The primary way that we will organize data is called the tabular data model. Data are arranged into a grid of rows and columns.

Tabular Data

The primary way that we will organize data is called the tabular data model. Data are arranged into a grid of rows and columns.

the individual entries are called **values**



breed	weight	height
Shih-Tzu	5.5	24
Labrador	33	56
Beagle	10.2	34
Newfoundland	70	69
Chihuahua	1.3	20
Affenpinscher	9.6	27

Tabular Data

The primary way that we will organize data is called the tabular data model. Data are arranged into a grid of rows and columns.

the individual entries are called **values**

breed	weight	height
Shih-Tzu	5.5	24
Labrador	33	56
Beagle	10.2	34
Newfoundland	70	69
Chihuahua	1.3	20
Affenpinscher	9.6	27

Rows ⇒ Observations

each row represents one observation

Tabular Data

The primary way that we will organize data is called the tabular data model. Data are arranged into a grid of rows and columns.

Columns ⇒ Variables

each column has name and a data type

the individual entries are called **values**

Rows ⇒ Observations

each row represents one observation

breed	weight	height
Shih-Tzu	5.5	24
Labrador	33	56
Beagle	10.2	34
Newfoundland	70	69
Chihuahua	1.3	20
Affenpinscher	9.6	27

Tabular Data

Each feature (column) in the tabular data model has a **data type** associated with it. This is not always explicit in way the dataset is saved, but will be defined when we are working with it in Python.

The most common data types we will see are:

- ↣ **int64:** a whole number
- ↣ **float64**: a number with a decimal point
- ↣ **object:** usually, an arbitrary sequences of any characters

There is a single type for each feature; we cannot mix and match data types. We will see other data types as they arise in our work.

Tabular Data

Implied data types in our example:

character <object>	numeric <float64>	numeric <int64>
breed	**weight**	**height**
Shih-Tzu	5.5	24
Labrador	33	56
Beagle	10.2	34
Newfoundland	70	69
Chihuahua	1.3	20
Affenpinscher	9.6	27

2. Python Basics

T. ARNOLD

Objects

Everything in Python is an object. We can save new objects by assigning a value to a name with an equals sign:

```
almost_pi = 3.14
almost_e = 2.718
almost_phi = 1.618
```

Functions

Functions are used to take a number of input objects and generate an output. These range from very simple mathematical functions (abs()) to functions to run long, complex modelling computations.

The inputs to a functions can be set by name or position. For example, these are the same:

```
pow(base=2, exp=5)
pow(2, 5)
```

You can see the arguments, names, and default values looking at the documention in the Python help pages.

Methods

A *method* is a special kind of function that is attached to an object. We call it by using a dot followed by the name of the function. Then, the function works just as with any other.

```
my_string = "DATA SCIENCE!"
my_string.count("A")
```

Most of the functions we will be using will be methods associated with data frame objects.

Modules

Only a limited number of functions are available when we start Python. To get access to others, we need to first import the modules (collections of additional functions and objects) that we want to work with. For example, if we start by running:

```
import pathlib
```

We then will be able to run the following code:

```
pathlib.Path("file.txt")
```

Note that we start with the module name followed by a dot an then the name of the function.

Modules, cont.

Alternatively, if we only need a few functions, we can import them directly:

```
from pathlib import Path
```

Which then lets us write the previous code like this:

```
Path("file.txt")
```

Modules, cont.

Other ways of importing modules include giving the module a different name:

```
import numpy as np
```

Which then treats the module as being nammed **np**:

```
np.array([1, 1])
```

Note: We generally only do this in a few cases like numpy and pandas that have standardized renaming values.