Reading data

Why data matters

"Companies are vacuuming up data **to make better decisions** about everything from product development and advertising to hiring. In their 2012 feature on big data, Andrew McAfee and Erik Brynjolfsson describe the opportunity and report that 'companies in the top third of their industry **in the use of data-driven decision making were, on average, 5% more productive and 6% more profitable than their competitors**' even after accounting for several confounding factors." (Walter Frick (2014), HBR)

https://hbr.org/2014/05/an-introduction-to-data-driven-decisions-for-managers-who-dont-like-math

Sidenote: Data comes from different sources and has different shapes and sizes

Data source	Data format examples	suffix	Package to read data
Flat files	Comma separated values	.csv	pandas
Declarative Languages (similar to a data format)	JSON	.json	json
	XML	.xml	ElementTree
Foreign data formats	Stata data files	.dta	pandas
	SPSS data files	.sav	savReaderWriter
	SAS data files	.sas	Pandas
Relational databases	SQLite		Sqlite3
	MySQL		MySQLdb

… and many more!

Sidenote: Data comes from different sources and has different shapes and sizes

Data source	Data format examples	suffix	Package to read data
Flat files	Comma separated values	.csv	pandas
Declarative Languages (similar to a data format)	JSON	.json	json
	XML	.xml	ElementTree
Foreign data formats	Stata data files	.dta	pandas
	SPSS data files	.sav	savReaderWriter
	SAS data files	.sas	Pandas
Relational databases	SQLite		Sqlite3
	MySQL		MySQLdb

… and many more!

Sidenote: Data comes from different sources and has different shapes and sizes

Data source	Data format examples	suffix	Package to read data
Flat files	Comma separated values	.csv	pandas
Declarative Languages (similar to a data format)	JSON	.json	json
	XML	.xml	ElementTree
Foreign data formats	Stata data files	.dta	pandas
	SPSS data files	.sav	savReaderWriter
	SAS data files	.sas	Pandas
Relational databases	SQLite		Sqlite3
	MySQL		MySQLdb

… and many more!

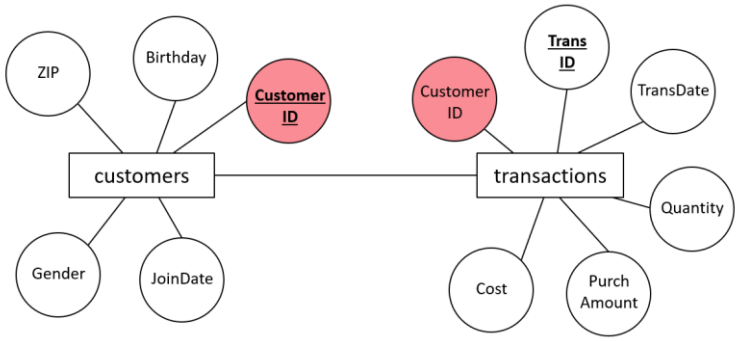
Comma separated values (CSV) and relational databases are the most common forms of data

CSV files are loaded directly into Python ❶

Data is read from an existing database ❷

transactionData.csv

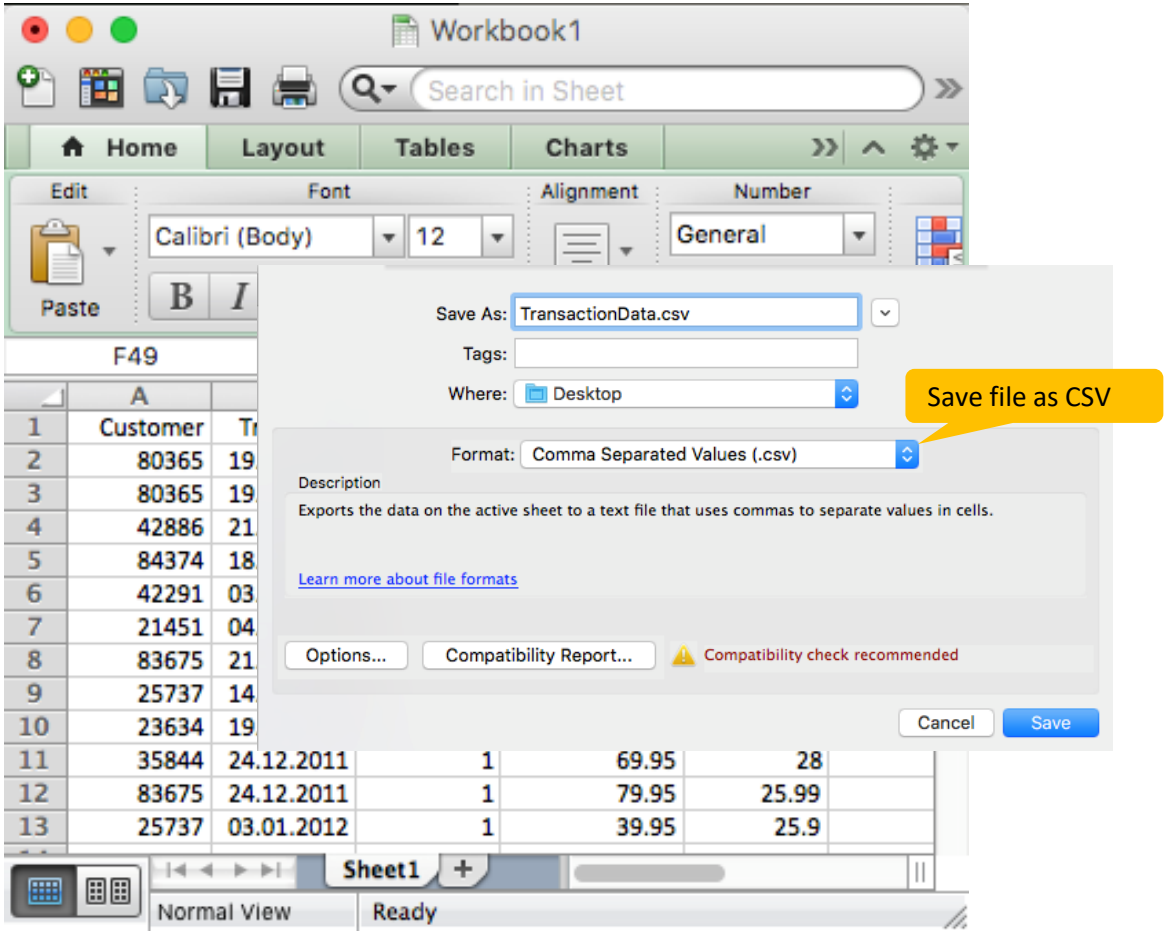
```
Customer,TransDate,Quantity,PurchAmount,Cost,TransID
149332,15/11/05,1,1.9995E2,107,127998739
172951,29/08/08,1,1.9995E2,108,128888288
120621,19/10/07,1,9.995E1,49,125375247
149236,14/11/05,1,3.995E1,1.895E1,127996226
149236,12/06/07,1,7.995E1,35,128670302
140729,19/11/09,1,1.2995E2,59,127637750
140729,19/11/09,1,7.995E1,3.918E1,127637750
140729,19/11/09,1,2.495E1,1.15E1,127637750
140729,19/11/09,1,7.95,2.96,127637750
140729,28/04/12,1,8.995E1,35,129377737
140729,28/04/12,1,6.995E1,2.64E1,129377737
180970,19/06/09,1,1.1995E2,5.846E1,129195647
180970,19/06/09,1,2.795E1,1.251E1,129195647
149332,13/12/05,1,4.995E1,2.487E1,129878743
149332,05/10/06,1,2.495E1,1.25E1,129883508
182927,11/02/09,2,5.99E1,1.592E1,129290963
182927,11/02/09,1,3.495E1,1.575E1,129290963
182927,11/02/09,1,1.2995E2,6.308E1,129290963
182927,11/02/09,1,2.495E1,8,129290963
```



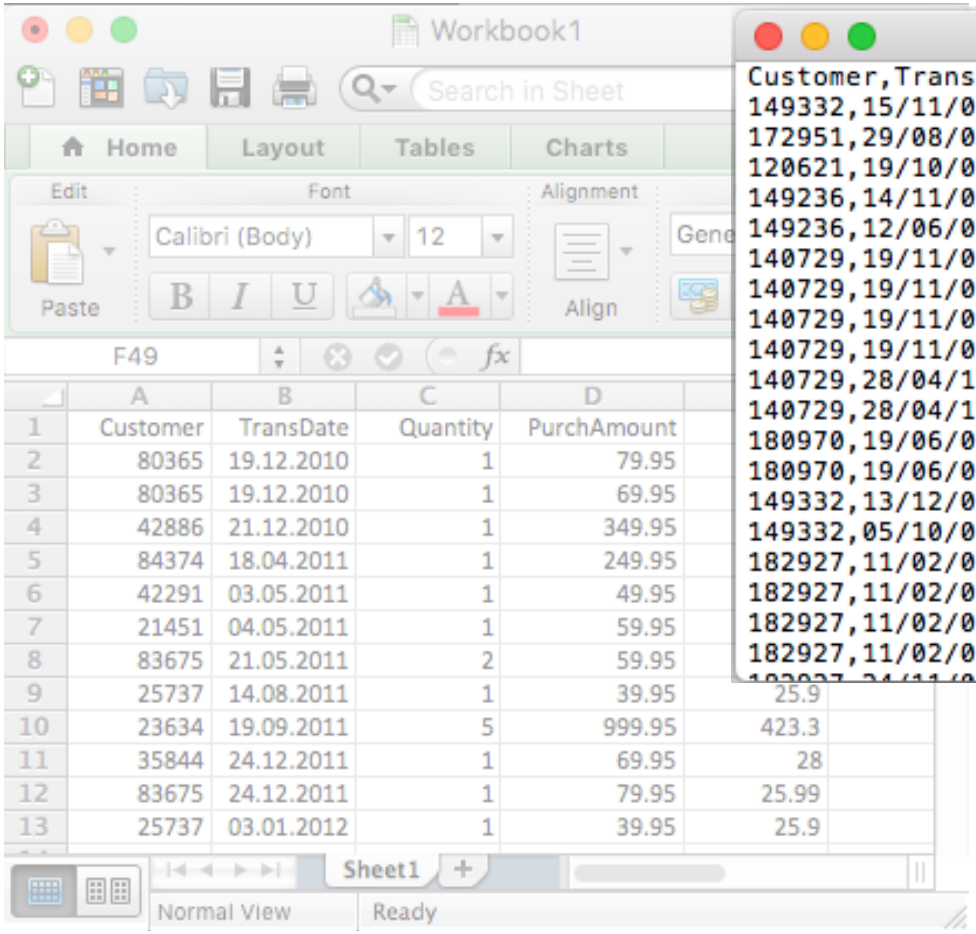
Reading CSV data in 3 simple steps

1. Ensure your data is in CSV format or create a CSV file out of data.
2. Load data into Python.
3. Make data available for processing.

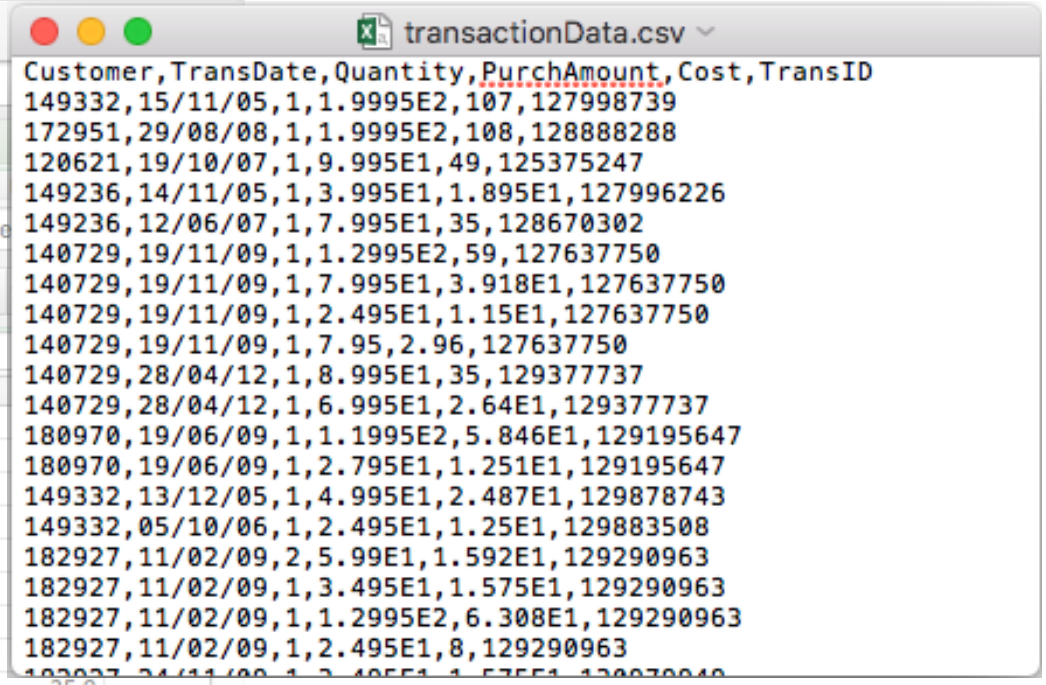
Step 1:
Create CSV out of spreadsheet



Step 1:
Create CSV out of spreadsheet

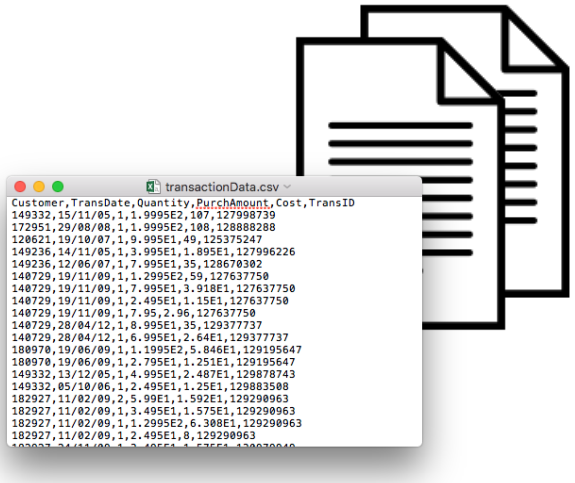


	A	B	C	D	
1	Customer	TransDate	Quantity	PurchAmount	
2	80365	19.12.2010	1	79.95	
3	80365	19.12.2010	1	69.95	
4	42886	21.12.2010	1	349.95	
5	84374	18.04.2011	1	249.95	
6	42291	03.05.2011	1	49.95	
7	21451	04.05.2011	1	59.95	
8	83675	21.05.2011	2	59.95	
9	25737	14.08.2011	1	39.95	25.9
10	23634	19.09.2011	5	999.95	423.3
11	35844	24.12.2011	1	69.95	28
12	83675	24.12.2011	1	79.95	25.99
13	25737	03.01.2012	1	39.95	25.9



Values are separated by commas (but you also may see semicolons ";")

Step 2:
How to load data into Python from a CSV file



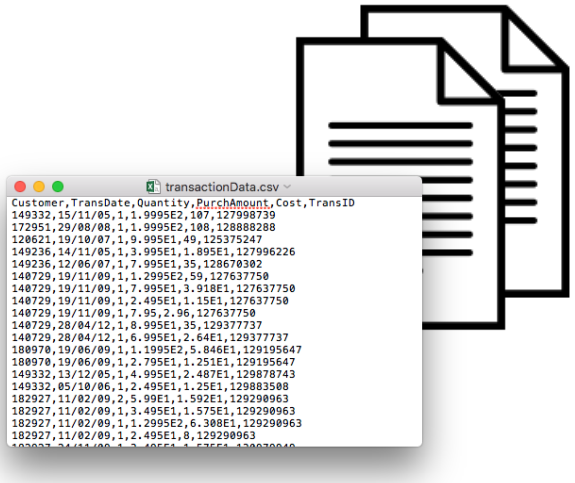
Customer	TransDate	Quantity	PurchAmount	Cost
149332	15/11/05	1	199.95	107.00
172951	29/08/08	1	199.95	108.00
120621	19/10/07	1	99.95	49.00
149236	14/11/05	1	39.95	18.95
149236	12/06/07	1	79.95	35.00
...	...	...	...	...

```
pd.read_csv(input, sep=",", ...)
```

1. We have already imported the `pandas` library

2. Name of CSV file

Step 2:
How to load data into Python from a CSV file



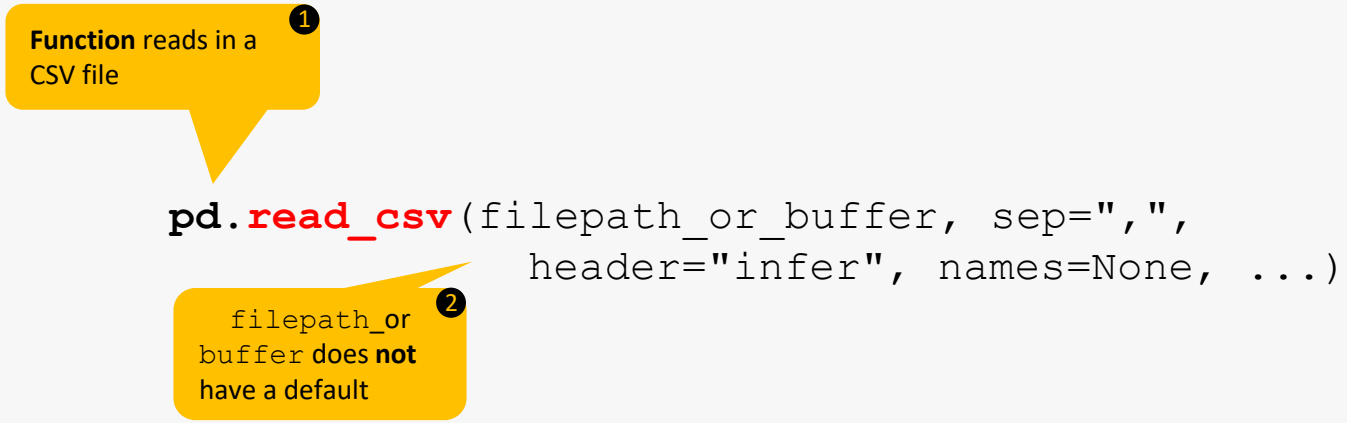
Customer	TransDate	Quantity	PurchAmount	Cost
149332	15/11/05	1	199.95	107.00
172951	29/08/08	1	199.95	108.00
120621	19/10/07	1	99.95	49.00
149236	14/11/05	1	39.95	18.95
149236	12/06/07	1	79.95	35.00
...	...	...	...	...

```
pd.read_csv(input, sep=",", ...)
```

1. We have already imported the `pandas` library

2. Name of CSV file

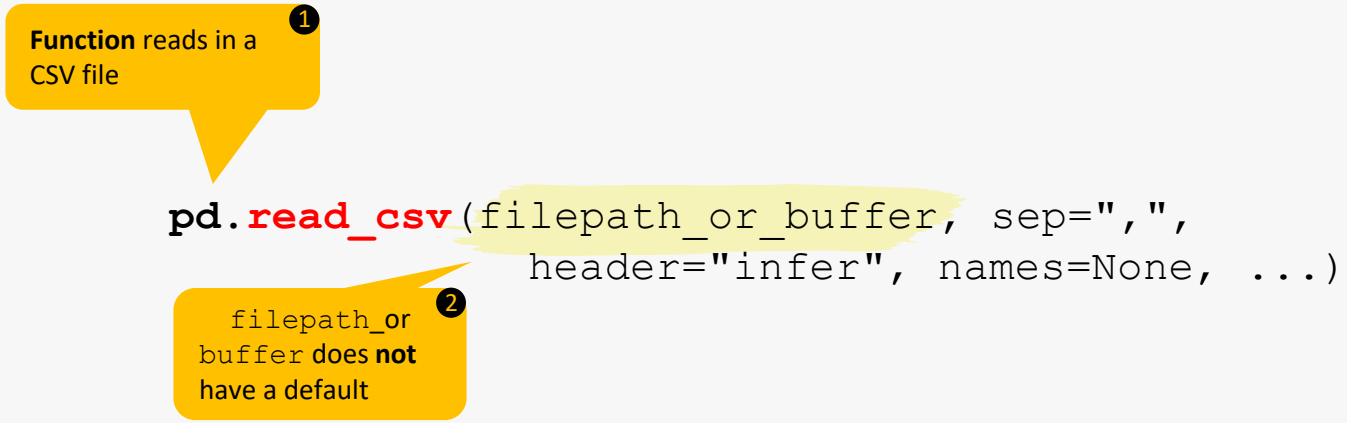
Python Basics: A function performs a specific action and is controlled through arguments



Differentiate two types of function arguments:

- **No default** if argument is not specified, error is returned
- **Default** if argument is not specified, default is used

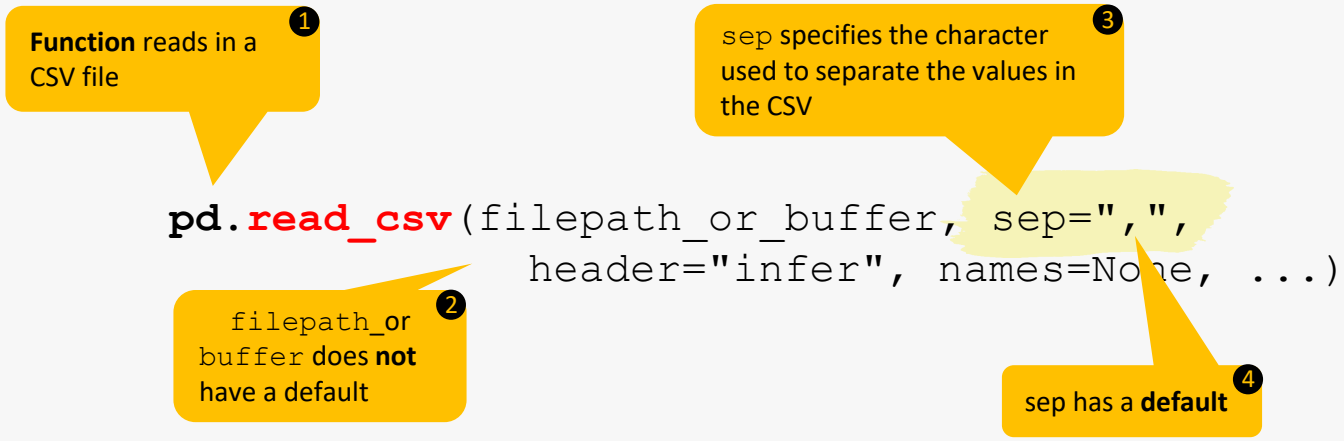
Python Basics: A function performs a specific action and is controlled through arguments



Differentiate two types of function arguments:

- **No default** if argument is not specified, error is returned
- **Default** if argument is not specified, default is used

Python Basics: A function performs a specific action and is controlled through arguments



Differentiate two types of function arguments:

- **No default** if argument is not specified, error is returned
- **Default** if argument is not specified, default is used

Why pandas?

- The **pandas** library speeds up data manipulations:
  - fast file reader
  - add/modify/delete columns by using groups without copying
  - data aggregation
  - ordered joins etc.
- Especially useful for large data (>1 GB in RAM)

Python Basics: Specifying your working directory

- To find the current working directory use:

```
os.getcwd()
```

1. The `os` - module is responsible for Mac **and** Windows. Review lecture 1 to see how to load a module.

- To set a new working directory use:

```
os.chdir("~/path/to/my/directory/")
```

2. Plug in your intended path here

Python Basics: Specifying your working directory

- To find the current working directory use:

```
os.getcwd()
```

1. The `os` - module is responsible for Mac **and** Windows. Review lecture 1 to see how to load a module.

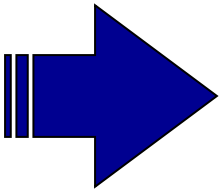
- To set a new working directory use:

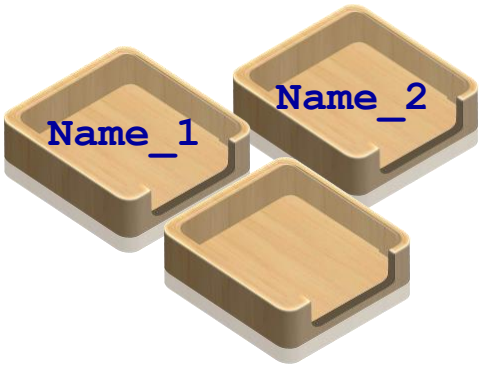
```
os.chdir("~/path/to/my/directory/")
```

2. Plug in your intended path here

Step 3:
Make data available for use

Customer	TransDate	Quantity	PurchAmount	Cost
149332	15/11/05	1	199.95	107.00
172951	29/08/08	1	199.95	108.00
120621	19/10/07	1	99.95	49.00
149236	14/11/05	1	39.95	18.95
149236	12/06/07	1	79.95	35.00
...	...	...	...	...





1 Name of new variable

```
variableName = (...)
```

2 Use equal sign to store object to `variableName`

Sidenote: Choose variable names wisely

- Do **not** name your variables after existing variables or functions.

Reading data