

Get started learning Python with [DataCamp's](#) free [Intro to Python tutorial](#). Learn Data Science by completing interactive coding challenges and watching videos by expert instructors. [Start Now!](#)

Ready to take the test? Head onto [LearnX](#) and get your Python Certification!

Pandas Basics

Pandas DataFrames

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

There are several ways to create a DataFrame. One way way is to use a dictionary. For example:

script.py

```
1 dict = {"country": ["Brazil", "Russia", "India", "China",
2 "South Africa"],
3 "capital": ["Brasilia", "Moscow", "New Dehli",
4 "Beijing", "Pretoria"],
5 "area": [8.516, 17.10, 3.286, 9.597, 1.221],
6 "population": [200.4, 143.5, 1252, 1357, 52.98] }
7
8 import pandas as pd
9 brics = pd.DataFrame(dict)
10 print(brics)
```

IPython Shell

	area	capital	country	population
0	8.516	Brasilia	Brazil	200.40
1	17.100	Moscow	Russia	143.50
2	3.286	New Dehli	India	1252.00
3	9.597	Beijing	China	1357.00
4	1.221	Pretoria	South Africa	52.98

In [1]: |

Run

Powered by DataCamp

As you can see with the new `brics` DataFrame, Pandas has assigned a key for each country as the numerical values 0 through 4. If you would like to have different index values, say, the two letter country code, you can do that easily as well.

script.py

```
1 # Set the index for brics
2 brics.index = ["BR", "RU", "IN", "CH", "SA"]
```

IPython Shell

	area	capital	country	population
BR	8.516	Brasilia	Brazil	200.40
RU	17.100	Moscow	Russia	143.50
IN	3.286	New Dehli	India	1252.00
CH	9.597	Beijing	China	1357.00
SA	1.221	Pretoria	South Africa	52.98

In [1]: |

Solution

Run

Powered by DataCamp

Another way to create a DataFrame is by importing a csv file using Pandas. Now, the csv `cars.csv` is stored and can be imported using `pd.read_csv`:

script.py

```
1 # Import pandas as pd
2 import pandas as pd
3
4 # Import the cars.csv data: cars
5 cars = pd.read_csv('cars.csv')
6
7 # Print out cars
8 print(cars)
```

IPython Shell

<script.py> output:

	Unnamed: 0	cars_per_cap	country	drives_right
0	US	809	United States	True
1	AUS	731	Australia	False
2	JAP	588	Japan	False
3	IN	18	India	False
4	RU	200	Russia	True
5	MOR	70	Morocco	True
6	EG	45	Egypt	True

In [1]: |

Solution

Run

Powered by DataCamp

Indexing DataFrames

There are several ways to index a Pandas DataFrame. One of the easiest ways to do this is by using square bracket notation.

In the example below, you can use square brackets to select one column of the `cars` DataFrame. You can either use a single bracket or a double bracket. The single bracket will output a Pandas Series, while a double bracket will output a Pandas DataFrame.

script.py

```
1 # Import pandas and cars.csv
2 import pandas as pd
3 cars = pd.read_csv('cars.csv', index_col = 0)
4
5 # Print out country column as Pandas Series
6 print(cars['cars_per_cap'])
7
8 # Print out country column as Pandas DataFrame
9 print(cars[['cars_per_cap']])
10
11 # Print out DataFrame with country and drives_right
```

Great job!

IPython Shell

```
IN      18
RU      200
MOR      70
EG       45
cars_per_cap  country
US      809  United States
AUS      731  Australia
JAP      588  Japan
IN       18  India
RU      200  Russia
MOR      70  Morocco
EG       45  Egypt
```

In [1]: |

Solution

Run

Powered by DataCamp

Square brackets can also be used to access observations (rows) from a DataFrame. For example:

script.py

```
1 # Import cars data
2 import pandas as pd
3 cars = pd.read_csv('cars.csv', index_col = 0)
4
5 # Print out first 4 observations
6 print(cars[0:4])
7
8 # Print out fifth and sixth observation
9 print(cars[4:6])
```

IPython Shell

```
In [1]: |
```

Solution

Run

Setting Up Workspace

Powered by DataCamp

You can also use `loc` and `iloc` to perform just about any data selection operation. `loc` is label-based, which means that you have to specify rows and columns based on their row and column labels. `iloc` is integer index based, so you have to specify rows and columns by their integer index like you did in the previous exercise.

script.py

```
1 # Import cars data
2 import pandas as pd
3 cars = pd.read_csv('cars.csv', index_col = 0)
4
5 # Print out observation for Japan
6 print(cars.iloc[2])
7
8 # Print out observations for Australia and Egypt
9 print(cars.loc[['AUS', 'EG']])
```

Great job!

IPython Shell

```
<script.py> output:
cars_per_cap    588
country         Japan
drives_right    False
Name: JAP, dtype: object
cars_per_cap  country drives_right
AUS          731  Australia      False
EG           45    Egypt        True
```

In [1]: |

Solution

Run

Powered by DataCamp

script.py

```
1 |
```

IPython Shell

```
In [1]: |
```