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

```
dict = {"country": ["Brazil", "Russia", "India", "China",
"South Africa"],
       "capital": ["Brasilia", "Moscow", "New Dehli",
"Beijing", "Pretoria"],
       "area": [8.516, 17.10, 3.286, 9.597, 1.221],
       "population": [200.4, 143.5, 1252, 1357, 52.98] }

import pandas as pd
brics = pd.DataFrame(dict)
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

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

Great job!

IPython Shell

```
    area    capital       country  population
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

```
# Import pandas as pd
import pandas as pd

# Import the cars.csv data: cars
cars = pd.read_csv('cars.csv')

# Print out cars
print(cars)
```

Great job!

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

```
# Import pandas and cars.csv
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)

# Print out country column as Pandas Series
print(cars['cars_per_cap'])

# Print out country column as Pandas DataFrame
print(cars[['cars_per_cap']])

# 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

```
# Import cars data
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)

# Print out first 4 observations
print(cars[0:4])

# Print out fifth and sixth observation
print(cars[4:6])
```

Great job!

IPython Shell

```
<script.py> output:
     cars_per_cap        country drives_right
US            809  United States         True
AUS           731      Australia        False
JAP           588          Japan        False
IN             18          India        False
     cars_per_cap  country drives_right
RU            200   Russia         True
MOR            70  Morocco         True

In [1]:
```

Solution Run

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

```
# Import cars data
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)

# Print out observation for Japan
print(cars.iloc[2])

# Print out observations for Australia and Egypt
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]:
```

Run

Powered by DataCamp

This site is generously supported by DataCamp. DataCamp offers online interactive Python Tutorials for Data Science. Join **over a million** other learners and get started learning Python for data science today!

Ready to take the test? Head onto LearnX and get your Python Certification!

« Previous Tutorial

Take the Test »

Next Tutorial »