Python For Data Science *Cheat Sheet*
Importing Data
Learn Python for data science **Interactively** at www.DataCamp.com

Importing Data in Python

Most of the time, you'll use either **NumPy** or **pandas** to import your data:

```
>>> import numpy as np
>>> import pandas as pd
```

Help

```
>>> np.info(np.ndarray.dtype)
>>> help(pd.read_csv)
```

Text Files

Plain Text Files

```
>>> filename = 'huck_finn.txt'
>>> file = open(filename, mode='r')     Open the file for reading
>>> text = file.read()                  Read a file's contents
>>> print(file.closed)                  Check whether file is closed
>>> file.close()                        Close file
>>> print(text)
```

Using the context manager `with`

```
>>> with open('huck_finn.txt', 'r') as file:
        print(file.readline())          Read a single line
        print(file.readline())
        print(file.readline())
```

Table Data: Flat Files

Importing Flat Files with numpy

Files with one data type

```
>>> filename = 'mnist.txt'
>>> data = np.loadtxt(filename,
                      delimiter=',',    String used to separate values
                      skiprows=2,       Skip the first 2 lines
                      usecols=[0,2],    Read the 1st and 3rd column
                      dtype=str)        The type of the resulting array
```

Files with mixed data types

```
>>> filename = 'titanic.csv'
>>> data = np.genfromtxt(filename,
                         delimiter=',',
                         names=True,    Look for column header
                         dtype=None)
```

```
>>> data_array = np.recfromcsv(filename)
```

The default `dtype` of the `np.recfromcsv()` function is `None`.

Importing Flat Files with pandas

```
>>> filename = 'winequality-red.csv'
>>> data = pd.read_csv(filename,
                       nrows=5,         Number of rows of file to read
                       header=None,     Row number to use as col names
                       sep='\t',        Delimiter to use
                       comment='#',     Character to split comments
                       na_values=[""])  String to recognize as NA/NaN
```

Excel Spreadsheets

```
>>> file = 'urbanpop.xlsx'
>>> data = pd.ExcelFile(file)
>>> df_sheet2 = data.parse('1960-1966',
                           skiprows=[0],
                           names=['Country',
                                  'AAM: War(2002)'])
>>> df_sheet1 = data.parse(0,
                           parse_cols=[0],
                           skiprows=[0],
                           names=['Country'])
```

To access the sheet names, use the `sheet_names` attribute:

```
>>> data.sheet_names
```

SAS Files

```
>>> from sas7bdat import SAS7BDAT
>>> with SAS7BDAT('urbanpop.sas7bdat') as file:
        df_sas = file.to_data_frame()
```

Stata Files

```
>>> data = pd.read_stata('urbanpop.dta')
```

Relational Databases

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite://Northwind.sqlite')
```

Use the `table_names()` method to fetch a list of table names:

```
>>> table_names = engine.table_names()
```

Querying Relational Databases

```
>>> con = engine.connect()
>>> rs = con.execute("SELECT * FROM Orders")
>>> df = pd.DataFrame(rs.fetchall())
>>> df.columns = rs.keys()
>>> con.close()
```

Using the context manager `with`

```
>>> with engine.connect() as con:
        rs = con.execute("SELECT OrderID FROM Orders")
        df = pd.DataFrame(rs.fetchmany(size=5))
        df.columns = rs.keys()
```

Querying relational databases with pandas

```
>>> df = pd.read_sql_query("SELECT * FROM Orders", engine)
```

Exploring Your Data

NumPy Arrays

```
>>> data_array.dtype       Data type of array elements
>>> data_array.shape       Array dimensions
>>> len(data_array)        Length of array
```

pandas DataFrames

```
>>> df.head()              Return first DataFrame rows
>>> df.tail()              Return last DataFrame rows
>>> df.index               Describe index
>>> df.columns             Describe DataFrame columns
>>> df.info()              Info on DataFrame
>>> data_array = data.values   Convert a DataFrame to an a NumPy array
```

Pickled Files

```
>>> import pickle
>>> with open('pickled_fruit.pkl', 'rb') as file:
        pickled_data = pickle.load(file)
```

HDF5 Files

```
>>> import h5py
>>> filename = 'H-H1_LOSC_4_v1-815411200-4096.hdf5'
>>> data = h5py.File(filename, 'r')
```

Matlab Files

```
>>> import scipy.io
>>> filename = 'workspace.mat'
>>> mat = scipy.io.loadmat(filename)
```

Exploring Dictionaries

Accessing Elements with Functions

```
>>> print(mat.keys())            Print dictionary keys
>>> for key in data.keys():      Print dictionary keys
        print(key)
meta
quality
strain
>>> pickled_data.values()        Return dictionary values
>>> print(mat.items())           Returns items in list format of (key, value)
                                 tuple pairs
```

Accessing Data Items with Keys

```
>>> for key in data ['meta'].keys()      Explore the HDF5 structure
        print(key)
Description
DescriptionURL
Detector
Duration
GPSstart
Observatory
Type
UTCstart
>>> print(data['meta']['Description'].value)   Retrieve the value for a key
```

Navigating Your FileSystem

Magic Commands

```
!ls       List directory contents of files and directories
%cd ..    Change current working directory
%pwd      Return the current working directory path
```

`os` Library

```
>>> import os
>>> path = "/usr/tmp"
>>> wd = os.getcwd()              Store the name of current directory in a string
>>> os.listdir(wd)               Output contents of the directory in a list
>>> os.chdir(path)               Change current working directory
>>> os.rename("test1.txt",       Rename a file
              "test2.txt")
>>> os.remove("test1.txt")       Delete an existing file
>>> os.mkdir("newdir")           Create a new directory
```