

Python for Data Science

Importing Data

Cheat Sheet

f616 adapted from datacamp.com

Introductory Note

This document is an adaption of the original datacamp.org cheat sheet.

- <https://www.datacamp.com/resources/cheat-sheets/importing-data-python-cheat-sheet>
- <https://github.com/f616/Python-Importing-Data-Cheat-Sheet>

1 Importing Data in Python

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

```
1 import pandas as pd
```

2 Help

```
1 np.info(np.ndarray.dtype)
2 help(pd.read_csv)
```

3 Text Files

Plain Text Files

```
1 filename = 'huck_finn.txt'
2 file = open(filename, mode='r') #Open the file for
  reading
3 text = file.read() #Read a file's contents
4 print(file.closed) #Check whether file is closed
5 file.close() #Close file
6 print(text)
```

Using the context manager with

```
1 with open('huck_finn.txt', 'r') as file:
2     print(file.readline()) #Read a single line
3     print(file.readline())
4     print(file.readline())
```

Table Data: Flat Files

Importing Flat Files with NumPy

```
1 filename = 'huck_finn.txt'
2 file = open(filename, mode='r') #Open the file
  for reading
3 text = file.read() #Read a file's contents
4 print(file.closed) #Check whether file is closed
5 file.close() #Close file
6 print(text)
```

Files with one data type

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

Files with mixed data type

```
1 filename = 'titanic.csv'
2 data = np.genfromtxt(filename, \
3     delimiter=',', \
4     names=True, \ #Look for column header
5     dtype=None)
6 data_array = np.recfromcsv(filename)
7 #The default dtype of the np.recfromcsv()
  function is None
```

Importing Flat Files with Pandas

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

4 Exploring Your Data

NumPy Arrays

```
1 data_array.dtype #Data type of array elements
2 data_array.shape #Array dimensions
3 len(data_array) #Length of array
```

Pandas DataFrames

```
1 df.head() #Return first DataFrame rows
2 df.tail() #Return last DataFrame rows
3 df.index #Describe index
4 df.columns #Describe DataFrame columns
5 df.info() #Info on DataFrame
6 data_array = data.values #Convert a DataFrame to an a NumPy
  array
```

5 SAS File

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

6 Stata File

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

7 Excel Spreadsheets

```
1 file = 'urbanpop.xlsx'
2 data = pd.ExcelFile(file)
3 df_sheet2 = data.parse('1960-1966', \
4     skiprows=[0], \
5     names=['Country', \
6     'AAM: War(2002)'])
7 df_sheet1 = data.parse(0, \
8     parse_cols=[0], \
9     skiprows=[0], \
10    names=['Country'])
```

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

```
1 data.sheet_names
```

8 Relational Databases

```
1 from sqlalchemy import create_engine
2 engine =
3     create_engine('sqlite://Northwind.sqlite')
```

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

```
1 table_names = engine.table_names()
```

Querying Relational Databases

```
1 con = engine.connect()
2 rs = con.execute("SELECT * FROM Orders")
3 df = pd.DataFrame(rs.fetchall())
4 df.columns = rs.keys()
5 con.close()
```

Using the context manager with

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

Querying Relational Databases with Pandas

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

9 Pickled Files

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

10 Matlab Files

```
1 import scipy.io
2 filename = 'workspace.mat'
3 mat = scipy.io.loadmat(filename)
```

11 HDF5 Files

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

12 Exploring Dictionaries

Querying relational databases with pandas

```
1 print(mat.keys()) #Print dictionary keys
2 for key in data.keys(): #Print dictionary keys
3     print(key)
4
5 meta
6 quality
7 strain
8
9 pickled_data.values() #Return dictionary values
10 print(mat.items()) #Returns items in list format of (key,
    value) tuple pairs
```

Accessing Data Items with Keys

```
1 for key in data ['meta'].keys(): #Explore the HDF5 structure
2     print(key)
3
4 Description
5 DescriptionURL
6 Detector
7 Duration
8 GPSstart
9 Observatory
10 Type
11 UTCstart
12
13 print(data['meta']['Description'].value) #Retrieve the value
    for a key
```

13 Navigating Your FileSystem

Magic Commands

```
1 !ls #List directory contents of files and directories
2 %cd .. #Change current working directory
3 %pwd #Return the current working directory path
```

OS Library

```
1 import os
2 path = '/usr/tmp'
3 wd = os.getcwd() #Store the name of current directory in a
    string
4 os.listdir(wd) #Output contents of the directory in a list
5 os.chdir(path) #Change current working directory
6 os.rename('test1.txt', 'test2.txt') #Rename a file
7 os.remove('test1.txt') #Delete an existing file
8 os.mkdir('newdir') #Create a new directory
```