

Importing Text Files I

```
open(file_name, 'r') # open the file
file.read() # read the file
file.close() # close the file
file.closed() # check if the file is closed
```

It is a good practice to close the file after reading it when using 'open'

Importing Text Files II

```
with open(file_name) as file:
    # open the file

file.read() # read the file
file.readline() # read line by line
```

When using the 'with' statement there is no need to close the file

Importing Flat Files with Numpy I

```
import numpy as np

np.loadtxt(file_name, delimiter=',')

skiprows=1

usecols=[0, 2]
```

dtype = str

loadtxt only works with numeric data

Importing Flat Files with Numpy II

```
import numpy as np

np.recfromcsv(file, delimiter=',', names=True, dtype=None)

np.genfromtxt(file, delimiter=',', names=True, dtype=None)
```

with the functions `recfromcsv()` and `genfromtxt()` we are able to import data with different types

Importing Stata Files

```
import pandas as pd

df = pd.read_stata('disarea.dta')
```

Importing Flat Files With Pandas

```
import pandas as pd

pd.read_csv(file)

nrows=5 # argument for the number of rows to load

header=None # argument for no header

importing the file # argument to set delimiter

comment='#' # argument takes characters that comments occur after in the file

showna='drop' # argument to only show specific columns

na_values=['N othi ng'] # argument to recognize a string as a NaN Value

to import the data as string
```

Import pickled files

```
import pickle

import numpy

open(file_name, 'r') # open the file

open(file_name, 'rb') # open the file

pickle.load(file)
```

Importing Spreadsheet Files

```
import pandas as pd

pd.ExcelFile(file) # opening the file

xl.sheet_names # exporting the sheet names

xl.parse(sheet_name='Sheet1') # loading sheet to dataframe

skiprows=[index] # skipping specific row

names=[List of Names] # naming the sheet's columns

usecols=[0,] # parse specific columns
```

skiprows, names and usecols are all arguments of the function `parse()`

Importing SAS Files

```
from sas7bdat import SAS7BDAT

import pandas as pd

with SAS7BDAT('file_name') as file:
    file.to_dataframe()
```

Importing HDF5 files

```
import numpy as np      import numpy
import h5py              importing the
                          h5py library
h5py.File(file, 'r')    reading the
                          file
```

Importing MATLAB files

```
import scipy.io          importing
                          scipy.io
scipy.io.loadmat('file_ reading
name')                   the file
```

Relational databases I

```
import pandas as pd
from sqlalchemy import create_engine
```

```
engine = create_engine('database://name:password@host:port/database')
con = engine.connect()
```

```
rs = con.execute('SELECT * FROM Album')
```

```
df = pd.DataFrame(rs.fetchall())
```

```
df.columns = rs.keys
```

```
con.close()
```

The best practice is to close the connection

Relational databases II

```
engine = create_engine('database://name:password@host:port/database')
```

```
with engine.connect() as con:
```

```
rs = con.execute('sql code')
```

```
df = pd.DataFrame(rs.fetchmany(size=3))
```

With 'open' you don't have to close the connection at the end

Relational databases III

```
engine = create_engine('database://name:password@host:port/database')
```

```
df = pd.read_sql_query('SQL code', engine)
```

Fastest way to connect to a database engine and perform query

importing pandas
importing the necessary library

creating an engine
performing query

connecting to the engine
performing query
save as a dataframe
set columns names
close the connection



By **issamdb**
cheatography.com/issamdb/

Published 16th August, 2019.
Last updated 16th August, 2019.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>