Importing Data in Python I Cheat Sheet

by issambd via cheatography.com/88527/cs/20287/

Importing Text Files I

`open(f ile _name, 'r ')`	open the file
`file.r ead()`	read the file
`file.c lose()`	close the file
`file.c losed()`	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(f ile _name) as file :`	open the file
`file.r ead()`	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`	import numpy
`np.loa dtx t(f ile _name, delimiter= ' ')`	importing the file
`skiprows=1`	argument to skip a specific row
`usecol s=[0, 2]`	argument to only show specific columns
`dtype = str'`	argument to import the data as string

loadtxt only works with numeric data

Importing Flat Files with Numpy II

`import numpy as np`	
`np.rec fro mcs v(file, delimi ter =",", names= True, dtype = None)`	
`np.gen fro mtx t(file, delimi ter =',', 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`	importing pandas
`df = pd.rea d_s tat a(' dis are a .dta')`	reading the stata file

Importing Flat Files With Pandas

`import pandas as pd`	import pandas
`pd.rea d_c sv( file )`	open csv file
`nrows=5`	argument for the number of rows to load
`header=None`	argument for no header
`sep='\t'`	argument to set delimiter
`comment='#'`	argument takes characters that comments occur after in the file
`na_values ='N othing'`	argument to recognize a string as a NaN Value

Import pickled files

`import pickle`	import numpy
`with open(f ile _name, 'rb') as fi :`	open the file
`pickle.lo ad( file)`	open the file

Importing Spreadsheet Files

`import pandas as pd`	importing pandas
`pd.Exc elF ile (file)`	opening the file
`xl.she et_ names`	exporting the sheet names
`xl.par se( she et_ nam e/i ndex)`	loading sheet to datafram
`skipro ws= [index]`	skipping specific row
`names= [List of Names]`	naming the sheet's columns
`usecol s=[0,]`	parse spesific columns

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

Importing SAS Files

`from sas7bdat import SAS7BDAT`
`import pandas as pd`
`with SAS7BD AT( 'fi le_ name') as ile:`
`file.t o_d ata _fr ame()`

Importing Data in Python I Cheat Sheet

by issambd via cheatography.com/88527/cs/20287/

Importing HDF5 files

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

Importing MATLAB files

`import scipy.io`	importing scipy.io
`cipy.i o.l oad mat ('f ile _ name')`	reading the file

Relational databases I

`import pandas as pd`	importing pandas
`from sqlalchemy import create _engine`	importing the necessary library
`engine = create _en gin e(' dat aba set ype :// /na me.d at aba set ype')`	creating an engine
`con = engine.co nnect()`	connecting to the engine
`rs = con.ex ecu te( 'SELECT * FROM Album')`	performe query
`df = pd.Dat aFr ame (rs.fe tch all())`	save as a dataframe
`df.columns = rs.keys`	set columns names
`con.cl ose()`	close the connection

The best practice is to close the connection

Relational databases II

`engine = create _en gin e(' dat aba set ype :// /na me.d at aba set ype')`	creating an engine
`with engine.co nnect() as con:`	connecting to the engine
`rs = con.ex ecu te('sql code')`	performe query
`df = pd.Dat aFr ame (rs.fe tch man y(s ize=3))`	load a number of rows as a dataframe

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

Relational databases III

`engine = create _en gin e(' dat aba set ype :// /na me.d at aba set ype')`	creating an engine
`df = pd.rea d_s ql_ que ry('SQL code', engine)`	performe query

Fastest way to connect to a database and perform query