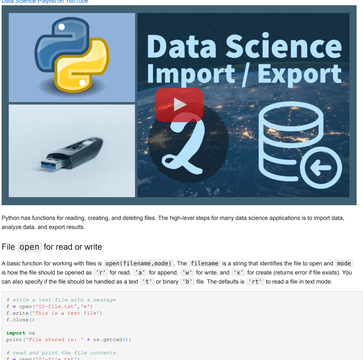
2. Import and Export Data

Data Science Playlist on YouTube



Python has functions for reading, creating, and deleting files. The high-level steps for many data science applications is to import data, analyze data, and export results.

File `open` for read or write

A basic function for working with files is `open(filename,mode)`. The `filename` is a string that identifies the file to open and `mode` is how the file should be opened as `'r'` for read, `'a'` for append, `'w'` for write, and `'x'` for create (returns error if file exists). You can also specify if the file should be handled as a text `'t'` or binary `'b'` file. The defaults is `'rt'` to read a file in text mode.

In [1]:
```
# write a test file with a message
f = open('02-file.txt','w')
f.write('This is a test file')
f.close()

import os
print('File stored in: ' + os.getcwd())

# read and print the file contents
f = open('02-file.txt')
print(f.read())
f.close()
```

```
File stored in: /home/curtis/classes/dynamic_optmization/data_science-master
This is a test file
```

Write Data Files

A common data file form is the Comma Separated Value (CSV) where entries are delimited (separated) by a comma. There is some data `n` that we would like to write to a CSV file with headers in `clist`. This example shows how to write the CSV file with several modules.

```
x,y,z
1,2,3
4,5,6
7,8,9
```

After running each cell, open the file in your current run directory with either Excel or a text editor.

`open` and `csv` module

The `with` command automatically closes the file when the commands inside the block are completed. The `newline=''` is only required for Windows. The `writerow` function writes one row of the CSV file.

In [2]:
```
clist = ['x','y','z']
n = [[1,2,3],\
     [4,5,6],\
     [7,8,9]]

import csv
with open("02-data1.csv",mode="w",newline='') as f:
    cw = csv.writer(f)
    cw.writerow(clist)
    for i in range(len(n)):
        cw.writerow(n[i])
```

`numpy` writes CSV

The numerical Python `numpy` package is used throughout this course. The `np.savetxt` function requires the file name, data `n`, the type of delimiter `','`, and header. If `comments=''` is omitted then the header has a `#` sign in front.

In [3]:
```
import numpy as np
np.savetxt('02-data2.csv',n,delimiter=',',comments='',header='x,y,z')
```

`pandas` writes CSV

The module `pandas` requires that the data be in `DataFrame` form for writing.

In [4]:
```
import pandas as pd
df = pd.DataFrame(n,columns=clist)
df.to_csv('02-data3.csv',index=False)
```

`pandas` writes XLSX and JSON

`pandas` can also write other files such as json or Excel files. You may need to install openpyxl to write the Excel file. You can do this in a cell with `!pip install openpyxl` and include `--user` if you do not have administrative privilege. You may need to restart the IPython notebook kernel after pip installs the `openpyxl` package.

In [5]:
```
df.to_json('02-data3.json',orient='table',index=False)
df.to_excel('02-data3.xlsx',index=False)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
/tmp/ipykernel_33006/1125059017.py in <module>
      1 df.to_json('02-data3.json',orient='table',index=False)
----> 2 df.to_excel('02-data3.xlsx',index=False)

~/.local/lib/python3.8/site-packages/pandas/core/generic.py in to_excel(self, excel_writer, sheet_name, na_rep, float_format, columns, header, index, index_label, startrow, startcol, engine, merge_cells, encoding, inf_rep, verbose, freeze_panes, storage_options)
   2202             inf_rep=inf_rep,
   2203         )
-> 2204         formatter.write(
   2205             excel_writer,
   2206             sheet_name=sheet_name,

~/.local/lib/python3.8/site-packages/pandas/io/formats/excel.py in write(self, writer, sheet_name, startrow, startcol, freeze_panes, engine, storage_options)
    832                 # error: Cannot instantiate abstract class 'ExcelWriter' with abstract
    833                 # attributes 'engine', 'save', 'supported_extensions' and 'write_cells'
--> 834                 writer = ExcelWriter(  # type: ignore[abstract]
    835                     writer, engine=engine, storage_options=storage_options
    836                 )

~/.local/lib/python3.8/site-packages/pandas/io/excel/_openpyxl.py in __init__(self, path, engine, date_format, datetime_format, mode, storage_options, if_sheet_exists, engine_kwargs, **kwargs)
     46         if_sheet_exists: str | None = None,
     47         # Use the openpyxl module as the Excel writer.
---> 48         from openpyxl.workbook import Workbook
     49
     50         engine_kwargs = combine_kwargs(engine_kwargs, kwargs)

ModuleNotFoundError: No module named 'openpyxl'
```

Write Activity

Use `numpy` to create `51` equally spaced values for `x` between `0` and `100`. Calculate `y=x**2` and `z=x**3` that are derived from `x`. Store `x`, `y`, and `z` in a CSV file with headings in file `02-test.csv`.

In [7]:
```
x = np.linspace(0, 100, 51)
y = x**2
z = x**3

data = {"x":x, "y":y,"z":z}
df = pd.DataFrame(data)
df.to_csv('02-test.csv')
```

Read Data Files

Just like writing the CSV files, there are modules for reading data files.

Use `numpy` to read CSV

The `np.loadtxt` function reads the CSV data file with option `skiprows=1` to skip the header row. Numpy does not label the rows or columns and only stores the CSV values.

In [ ]:
```
data = np.loadtxt("02-data1.csv",delimiter=',',skiprows=1)
print(data)
```

Use `pandas` to read CSV

The `pd.read_csv` function reads the CSV data file the the header row to label the columns. The `data.head()` and `data.tail()` functions prints up to the first or last 5 values, respectively.

In [ ]:
```
data = pd.read_csv("02-data1.csv")
data.head()
```


Read Activity

Use `pandas` to read the `02-test.csv` file created above. Display the first 5 rows of the file.

In [10]:
```
data = pd.read_csv("02-test.csv")
data.head()
```

Out[10]:

	Unnamed: 0	x	y	z
0	0	0.0	0.0	0.0
1	1	2.0	4.0	8.0
2	2	4.0	16.0	64.0
3	3	6.0	36.0	216.0
4	4	8.0	64.0	512.0

Delete Data Files

It is also possible to delete files using the `os` (operating system) module.

```
import os
os.remove('02-data1.csv')
```

The `glob` module builds a list of files that start with `02-data` and end with `.csv`. It uses the wildcard character `*` to select any files that match the first and last parts.

```
['02-data1.csv', '02-data2.csv', '02-data3.csv']
```

If the user's first letter of the answer is `y`, then it deletes these files.

In [ ]:
```
import os
import glob
filelist = glob.glob('02-data*.csv')

if filelist==[]:
    print("No files to delete")
    answ='no'
else:
    ans = input('Delete files '+str(filelist)+'? ')

if ans[0].lower()=='y':
    for f in filelist:
        os.remove(f)
```


Delete Activity

Delete the file `02-test.csv` with Python.

In [11]:
```
os.remove("02-test.csv")
```


TCLab Activity

Write data file `02-tclab.csv` with 5 columns that include time in seconds (`t`), heater levels (`Q1` and `Q2`), and temperatures (`lab.T1` and `lab.T2`). Include a data row every second for 20 seconds. The starting script only prints those values to the screen but they also need to be saved to a file.

In [15]:
```
import tclab
import time
n = 20
Q1 = 30; Q2 = 70
t_hist = []
q1 = []
q2 = []
t1 = []
t2 = []
with tclab.TCLab() as lab:
    for (t,Q1,Q2) in lab.Q1Q2():
        print('t Q1 Q2 T1 T2')
        for i in range(n+1):
            print(t,Q1,Q2,lab.T1,lab.T2)
            t_hist.append(t)
            q1.append(Q1)
            q2.append(Q2)
            t1.append(lab.T1)
            t2.append(lab.T2)
            time.sleep(1)

data = ["t","t_hist","q1":q1,"q2":q2,"t1":t1,"t2":t2}
pd.DataFrame(data).to_csv("02-tclab.csv")
```

```
TCLab version 0.4.9
Arduino Leonardo connected on port /dev/ttyACM1 at 115200 baud.
TCLab Firmware 2.0.1 Arduino Leonardo/Micro.
t Q1 Q2 T1 T2
0 30 70 27.312 27.2115
1 30 70 27.344 27.2212
2 30 70 27.344 27.3164
3 30 70 27.437 27.376
4 30 70 27.312 27.376
5 30 70 27.344 27.408
6 30 70 27.344 27.376
7 30 70 27.344 27.279
8 30 70 27.344 27.344
9 30 70 27.344 27.344
10 30 70 27.344 27.344
11 30 70 27.44 27.505
12 30 70 27.634 27.569
13 30 70 27.634 27.795
14 30 70 27.666 27.892
15 30 70 27.763 27.985
16 30 70 27.892 28.053
17 30 70 27.988 28.246
18 30 70 27.988 28.278
19 30 70 28.246 28.278
TCLab disconnected successfully.
```

Read the `02-tclab.csv` file and print the first 5 rows. If you do not have a TCLab device, read the data file from the `url` with `data=pd.read_csv(url)`

In [16]:
```
# read this file if you do not have a TCLab
url = 'http://apmonitor.com/pdc/uploads/Main/tclab_data2.txt'

pd.read_csv("02-tclab.csv").head()
```

Out[16]:

	Unnamed: 0	t	q1	q2	t1	t2
0	0	0	30	70	27.312	27.344
1	1	1	30	70	27.344	27.2115
2	2	2	30	70	27.279	27.118
3	3	3	30	70	27.247	27.054
4	4	4	30	70	27.247	27.118

In [ ]:
```

```