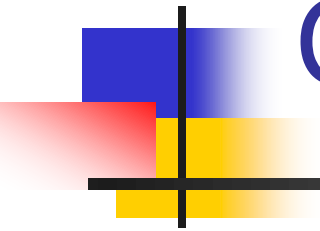
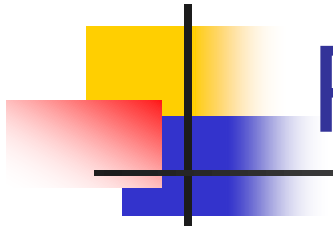


Computer Methods (MAE 3403)

Chapter 3

I/O, Files



Reading/Writing Data

- Storing data and results of programs are important.
- When Python closes, all variables in the memory are lost.
- Data must be stored, readable by or written in a form that can be used by other programs.

Open a file

```
f = open(filename, mode)
```

Mode:

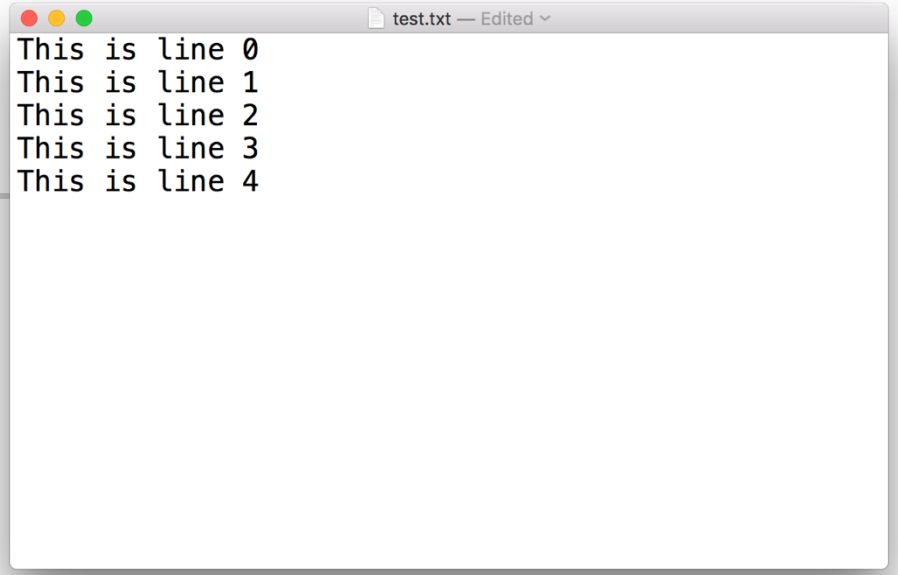
- ‘r’, this is the default mode, which opens a file for reading
- ‘w’, this mode opens a file for writing, if the file does not exist, it creates a new file.
- ‘a’, open a file in append mode, append data to end of file. If the file does not exist, it creates a new file.
- ‘b’, open a file in binary mode.
- ‘r+’, open a file (do not create) for reading and writing.
- ‘w+’, open or create a file for writing and reading, discard existing contents.
- ‘a+’, open or create file for reading and writing, and append data to end of file.

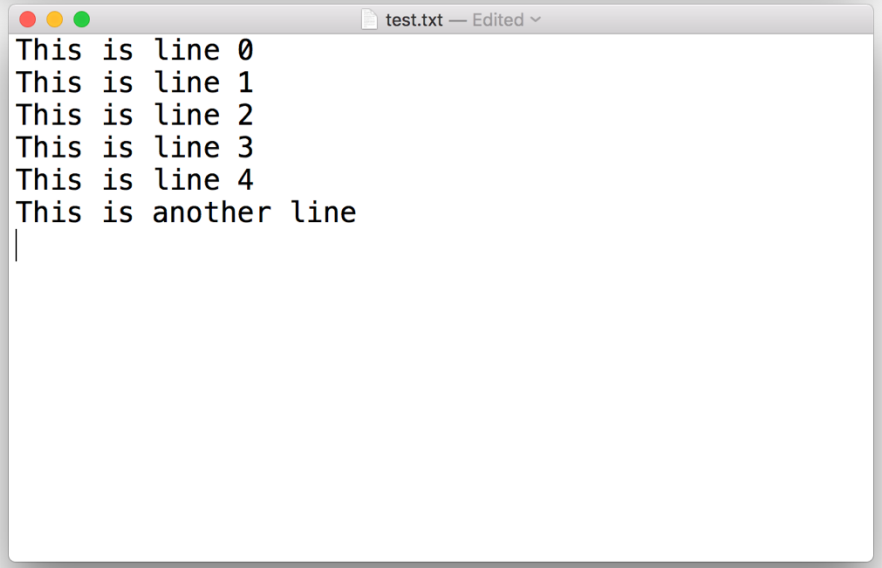
After all operations, DO NOT forget: `f.close()`

Write to a file

```
f = open('test.txt', 'w')
for i in range(5):
    f.write(f"This is line {i}\n")
f.close()
```

```
f = open('test.txt', 'a')
f.write(f"This is another line\n")
f.close()
```





Read a file

```
f = open('./test.txt', 'r')
content = f.read()
f.close()
print(content)
```

```
This is line 0
This is line 1
This is line 2
This is line 3
This is line 4
This is another line
```

```
type(content)
```

```
str
```

```
f = open('./test.txt', 'r')
contents = f.readlines()
f.close()
print(contents)
```

```
['This is line 0\n', 'This is line 1\n', ...]
```

```
type(content)
```

```
list
```

Operation on each line of a file

- Read only the current line

```
f.readline(n)
```

read n characters from the line.

- Operation on each line:

```
for line in f:
    do something with line
```

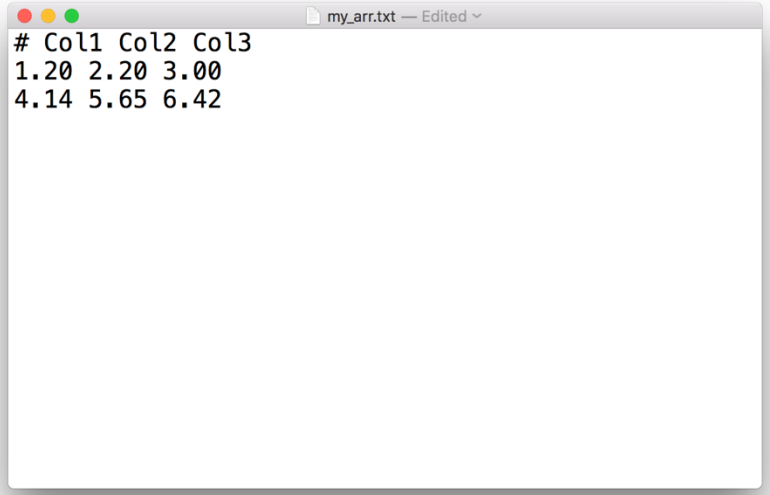
```
x = []
data = open('sunspots.txt', 'r')
for line in data:
    x.append(eval(line.split()[3]))
data.close()
```

'sunspots.txt'

```
1896 05 26 40.94
1896 05 27 40.58
1896 05 28 40.20
...
```

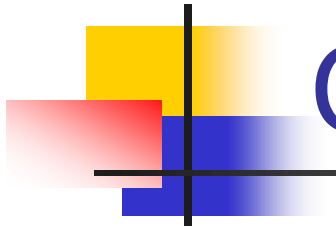
Dealing with numbers and arrays

```
import numpy as np
arr = np.array([[1.20, 2.20, 3.00],
[4.14, 5.65, 6.42]])
np.savetxt('my_arr.txt', arr,
fmt='%.2f', header = 'Col1 Col2 Col3')
```




```
my_arr = np.loadtxt('my_arr.txt')
```

```
array([[1.2 , 2.2 , 3. ],
[4.14, 5.65, 6.42]])
```



CSV (Comma-separated values) files

- Each line (row) is one data record and each record contains one or more fields, separated by commas.
- Useable by MS Excel.

- Python has its own csv module.
- We can also use numpy package to deal with csv files.

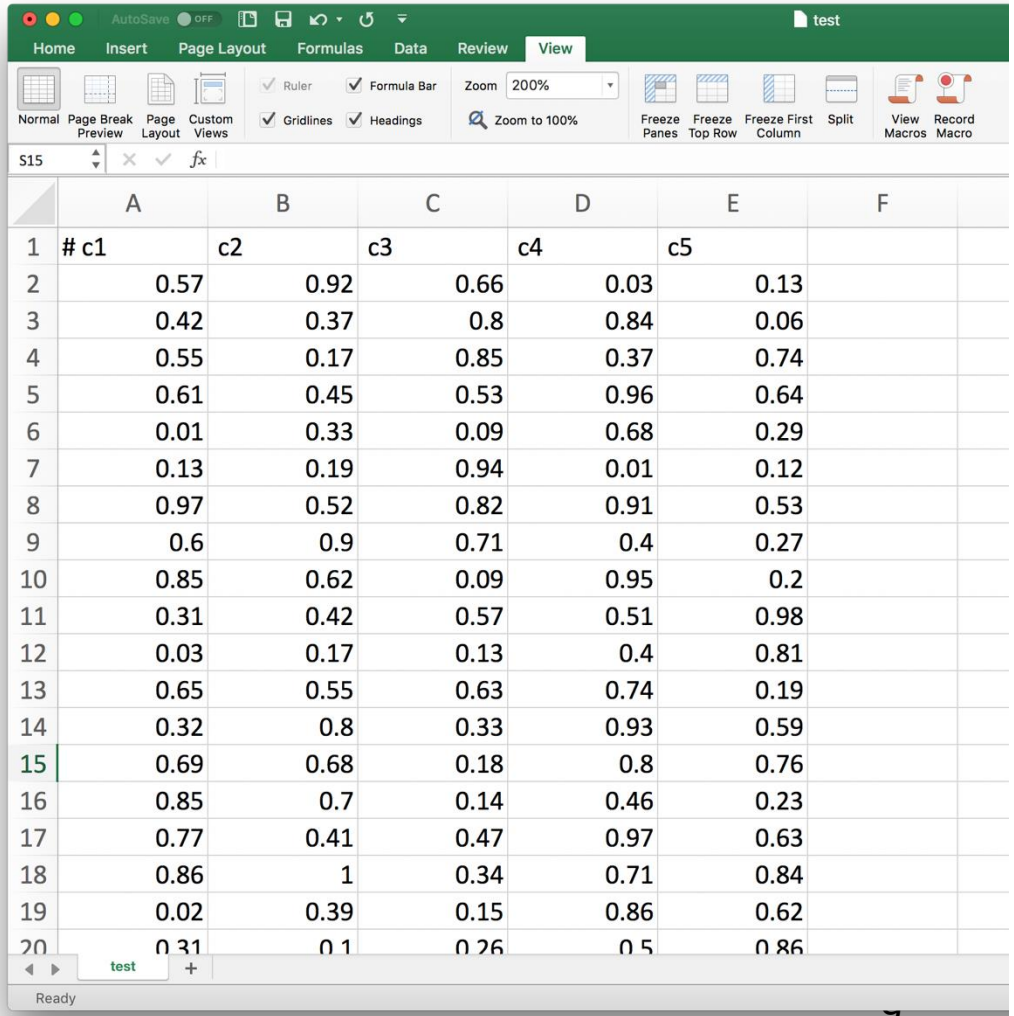
Write and read a CSV file

- Write

```
import numpy as np
data = np.random.random((100,5))
np.savetxt('test.csv', data, fmt = '%.2f',
delimiter=',', header = 'c1, c2, c3, c4,
c5')
```

- Read

```
my_csv = np.loadtxt('./test.csv',
delimiter=',')
```



	A	B	C	D	E	F
1	# c1	c2	c3	c4	c5	
2	0.57	0.92	0.66	0.03	0.13	
3	0.42	0.37	0.8	0.84	0.06	
4	0.55	0.17	0.85	0.37	0.74	
5	0.61	0.45	0.53	0.96	0.64	
6	0.01	0.33	0.09	0.68	0.29	
7	0.13	0.19	0.94	0.01	0.12	
8	0.97	0.52	0.82	0.91	0.53	
9	0.6	0.9	0.71	0.4	0.27	
10	0.85	0.62	0.09	0.95	0.2	
11	0.31	0.42	0.57	0.51	0.98	
12	0.03	0.17	0.13	0.4	0.81	
13	0.65	0.55	0.63	0.74	0.19	
14	0.32	0.8	0.33	0.93	0.59	
15	0.69	0.68	0.18	0.8	0.76	
16	0.85	0.7	0.14	0.46	0.23	
17	0.77	0.41	0.47	0.97	0.63	
18	0.86	1	0.34	0.71	0.84	
19	0.02	0.39	0.15	0.86	0.62	
20	0.31	0.1	0.26	0.5	0.86	

Python can work with other files

- Pickle file (store dictionaries, tuples, etc, serialize python objects)

```
import pickle
dict_a = {'A':0, 'B':1, 'C':2}
pickle.dump(dict_a, open('test.pkl', 'wb'))


my_dict = pickle.load(open('./test.pkl', 'rb'))
```

- JSON (JavaScript Object Notation)
  - language-independent
  - less space, faster than pickle
  - import json
- HDF5 (Hierarchical data format)
  - large amount of data
  - data and group: folder-like
  - import h5py