EASC2410 Lecture 9

Python Basics: File System, Paths, Pandas

Dr. Binzheng Zhang
Department of Earth Sciences

Review of Lecture 8

In Lecture 8, we learned:

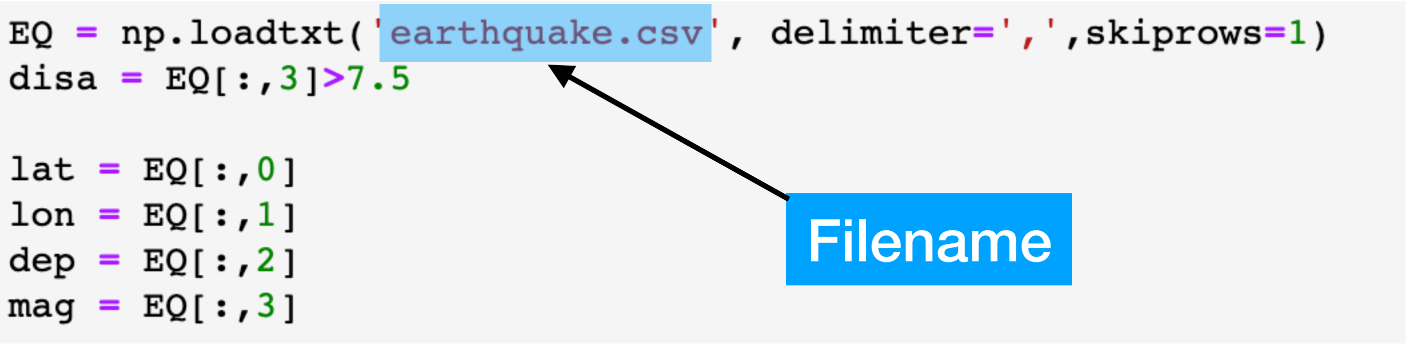
- **Creating maps using the *basemap* module**
- **plotting spatial data points on your maps**

In Lecture 9, you will learn:

- **Basics concepts of file system (since you're gonna work with data files)**
- **Intro to Pandas - load data**

Recall in Exercise #8

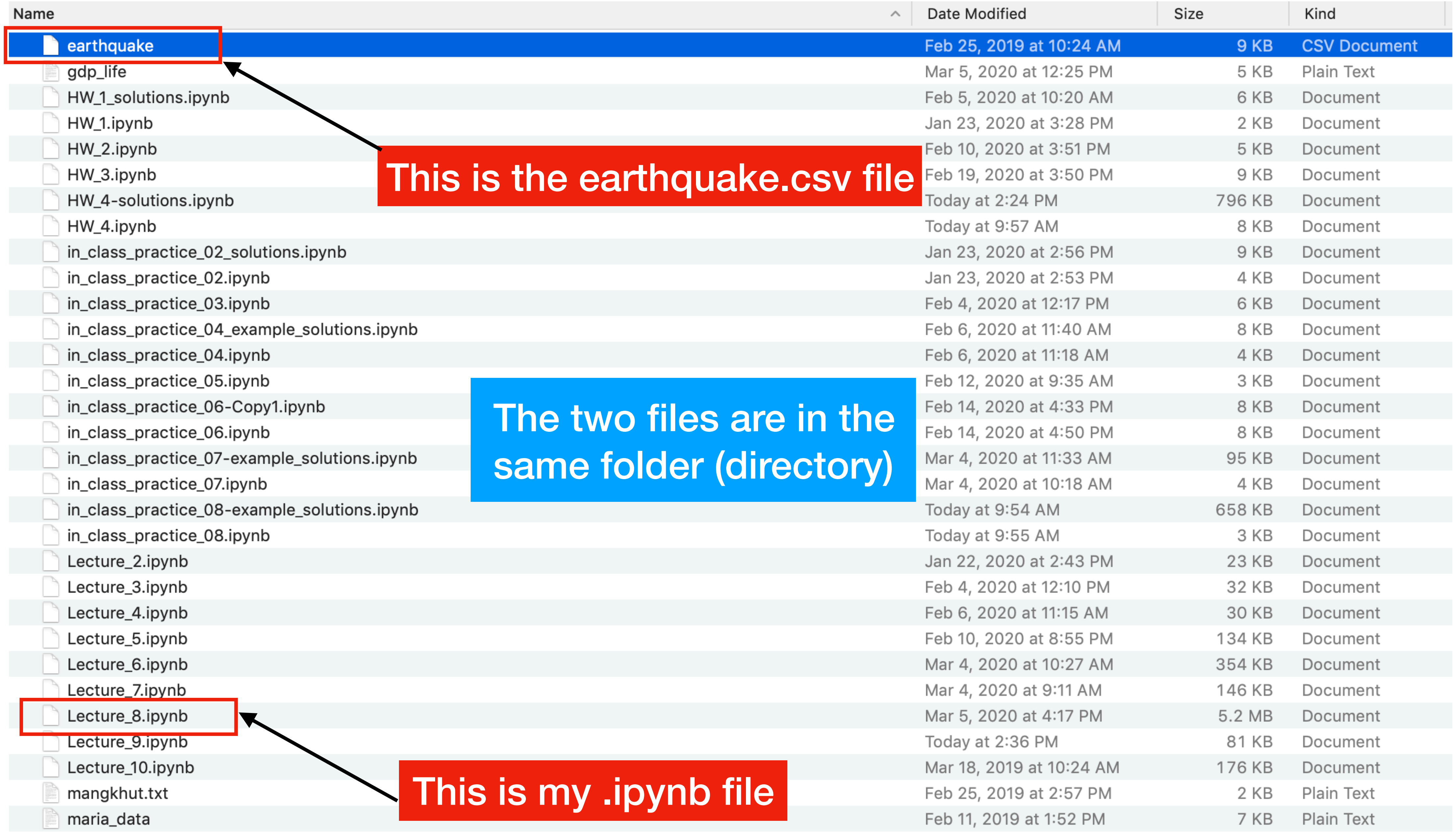
When you were trying to load the data file using the NumPy function:



```
EQ = np.loadtxt('earthquake.csv', delimiter=',',skiprows=1)
disa = EQ[:,3]>7.5

lat = EQ[:,0]
lon = EQ[:,1]
dep = EQ[:,2]
mag = EQ[:,3]
```

Filename

The question is - How do Python know where to find the file?



Name	Date Modified	Size	Kind
earthquake	Feb 25, 2019 at 10:24 AM	9 KB	CSV Document
gdp_life	Mar 5, 2020 at 12:25 PM	5 KB	Plain Text
HW_1_solutions.ipynb	Feb 5, 2020 at 10:20 AM	6 KB	Document
HW_1.ipynb	Jan 23, 2020 at 3:28 PM	2 KB	Document
HW_2.ipynb	Feb 10, 2020 at 3:51 PM	5 KB	Document
HW_3.ipynb	Feb 19, 2020 at 3:50 PM	9 KB	Document
HW_4-solutions.ipynb	Today at 2:24 PM	796 KB	Document
HW_4.ipynb	Today at 9:57 AM	8 KB	Document
in_class_practice_02_solutions.ipynb	Jan 23, 2020 at 2:56 PM	9 KB	Document
in_class_practice_02.ipynb	Jan 23, 2020 at 2:53 PM	4 KB	Document
in_class_practice_03.ipynb	Feb 4, 2020 at 12:17 PM	6 KB	Document
in_class_practice_04_example_solutions.ipynb	Feb 6, 2020 at 11:40 AM	8 KB	Document
in_class_practice_04.ipynb	Feb 6, 2020 at 11:18 AM	4 KB	Document
in_class_practice_05.ipynb	Feb 12, 2020 at 9:35 AM	3 KB	Document
in_class_practice_06-Copy1.ipynb	Feb 14, 2020 at 4:33 PM	8 KB	Document
in_class_practice_06.ipynb	Feb 14, 2020 at 4:50 PM	8 KB	Document
in_class_practice_07-example_solutions.ipynb	Mar 4, 2020 at 11:33 AM	95 KB	Document
in_class_practice_07.ipynb	Mar 4, 2020 at 10:18 AM	4 KB	Document
in_class_practice_08-example_solutions.ipynb	Today at 9:54 AM	658 KB	Document
in_class_practice_08.ipynb	Today at 9:55 AM	3 KB	Document
Lecture_2.ipynb	Jan 22, 2020 at 2:43 PM	23 KB	Document
Lecture_3.ipynb	Feb 4, 2020 at 12:10 PM	32 KB	Document
Lecture_4.ipynb	Feb 6, 2020 at 11:15 AM	30 KB	Document
Lecture_5.ipynb	Feb 10, 2020 at 8:55 PM	134 KB	Document
Lecture_6.ipynb	Mar 4, 2020 at 10:27 AM	354 KB	Document
Lecture_7.ipynb	Mar 4, 2020 at 9:11 AM	146 KB	Document
Lecture_8.ipynb	Mar 5, 2020 at 4:17 PM	5.2 MB	Document
Lecture_9.ipynb	Today at 2:36 PM	81 KB	Document
Lecture_10.ipynb	Mar 18, 2019 at 10:24 AM	176 KB	Document
mangkhut.txt	Feb 25, 2019 at 2:57 PM	2 KB	Plain Text
maria_data	Feb 11, 2019 at 1:52 PM	7 KB	Plain Text

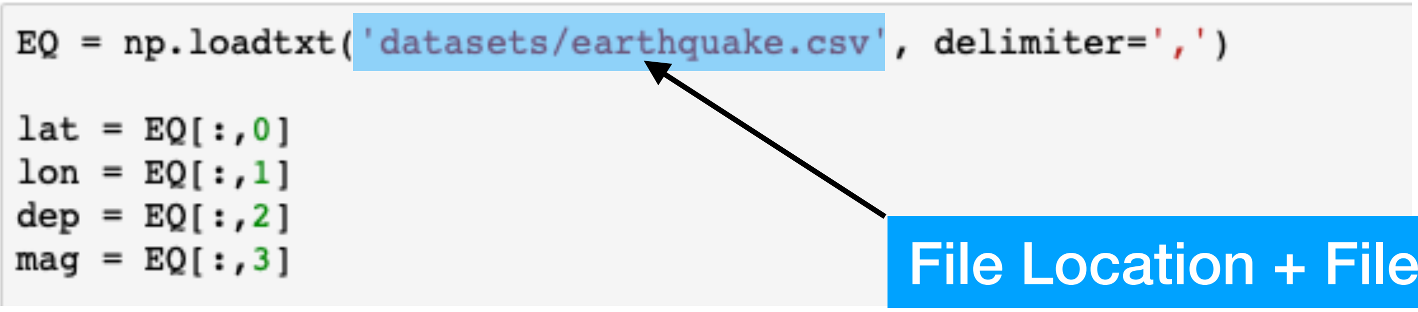
This is the earthquake.csv file

The two files are in the same folder (directory)

This is my .ipynb file

Try another example

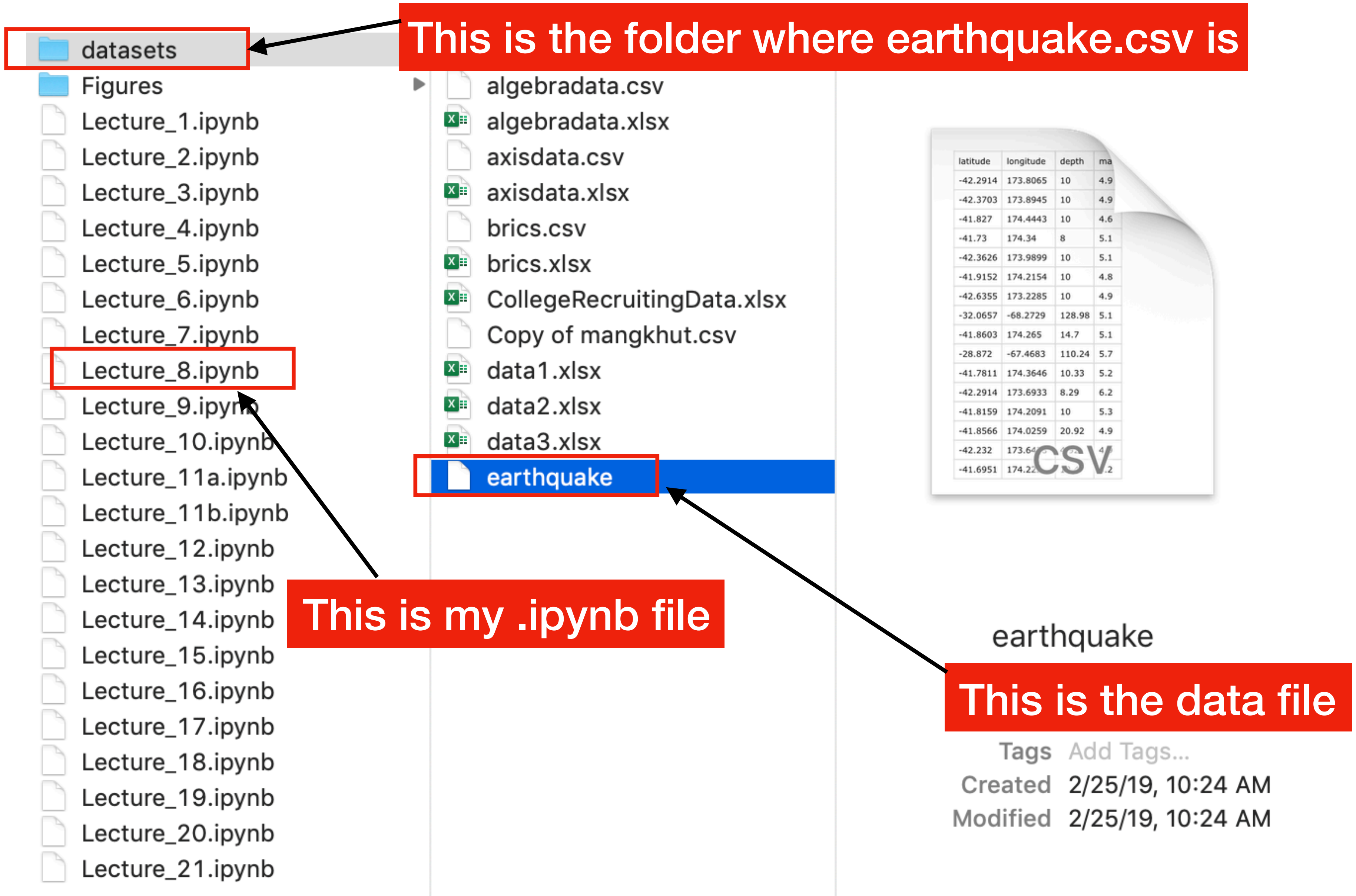
What if you were trying to load the data file using the following code?



```
EQ = np.loadtxt('datasets/earthquake.csv', delimiter=',')

lat = EQ[:,0]
lon = EQ[:,1]
dep = EQ[:,2]
mag = EQ[:,3]
```

File Location + Filename

Now the question is - How do Python know where to find the file?



This is the folder where earthquake.csv is

- datasets
- Figures
- Lecture_1.ipynb
- Lecture_2.ipynb
- Lecture_3.ipynb
- Lecture_4.ipynb
- Lecture_5.ipynb
- Lecture_6.ipynb
- Lecture_7.ipynb
- Lecture_8.ipynb
- Lecture_9.ipynb
- Lecture_10.ipynb
- Lecture_11a.ipynb
- Lecture_11b.ipynb
- Lecture_12.ipynb
- Lecture_13.ipynb
- Lecture_14.ipynb
- Lecture_15.ipynb
- Lecture_16.ipynb
- Lecture_17.ipynb
- Lecture_18.ipynb
- Lecture_19.ipynb
- Lecture_20.ipynb
- Lecture_21.ipynb

This is my .ipynb file

- algebradata.csv
- algebradata.xlsx
- axisdata.csv
- axisdata.xlsx
- brics.csv
- brics.xlsx
- CollegeRecruitingData.xlsx
- Copy of mangkhut.csv
- data1.xlsx
- data2.xlsx
- data3.xlsx
- earthquake

latitude	longitude	depth	ma
-42.2914	173.8065	10	4.9
-42.3703	173.8945	10	4.9
-41.827	174.4443	10	4.6
-41.73	174.34	8	5.1
-42.3626	173.9899	10	5.1
-41.9152	174.2154	10	4.8
-42.6355	173.2285	10	4.9
-32.0657	-68.2729	128.98	5.1
-41.8603	174.265	14.7	5.1
-28.872	-67.4683	110.24	5.7
-41.7811	174.3646	10.33	5.2
-42.2914	173.6933	8.29	6.2
-41.8159	174.2091	10	5.3
-41.8566	174.0259	20.92	4.9
-42.232	173.64	[illegible]	[illegible]
-41.6951	174.22	[illegible]	.2

CSV

earthquake

This is the data file

Tags Add Tags...
Created 2/25/19, 10:24 AM
Modified 2/25/19, 10:24 AM

Summary

The two files are in the same folder (directory)

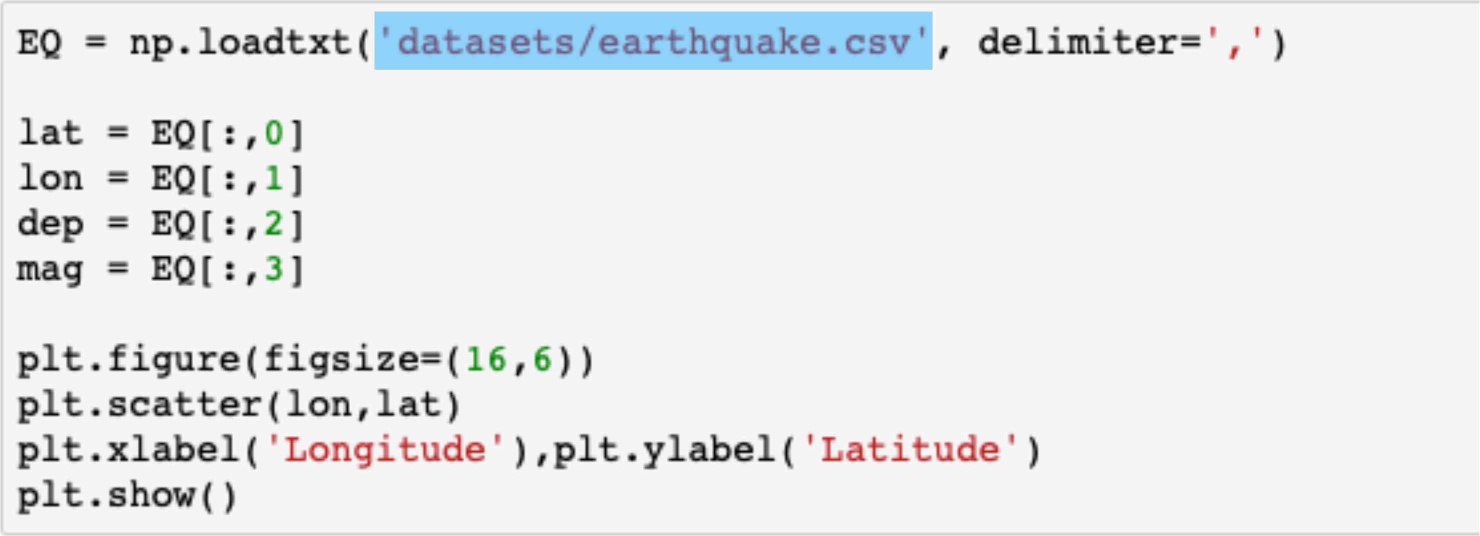
Python looks at the current directory to find the file

The two files are NOT in the same folder (directory)

Python goes to a given location (directory) to find the file

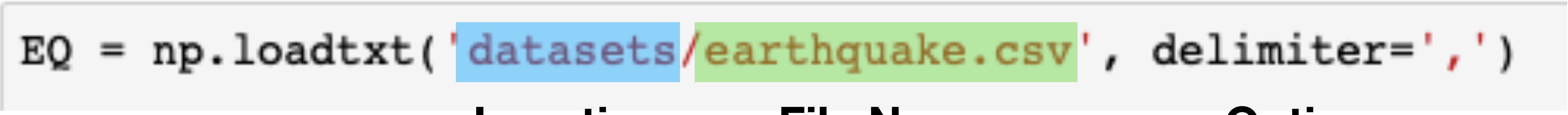
Specify the location and file name of a data file

When you were trying to load the data file using the NumPy function:



```
EQ = np.loadtxt('datasets/earthquake.csv', delimiter=',')

lat = EQ[:,0]
lon = EQ[:,1]
dep = EQ[:,2]
mag = EQ[:,3]

plt.figure(figsize=(16,6))
plt.scatter(lon,lat)
plt.xlabel('Longitude'),plt.ylabel('Latitude')
plt.show()
```

Basically, what needed in the `loadtxt()` function is the location, the name of the data file, together with options



```
EQ = np.loadtxt('datasets/earthquake.csv', delimiter=',')
```

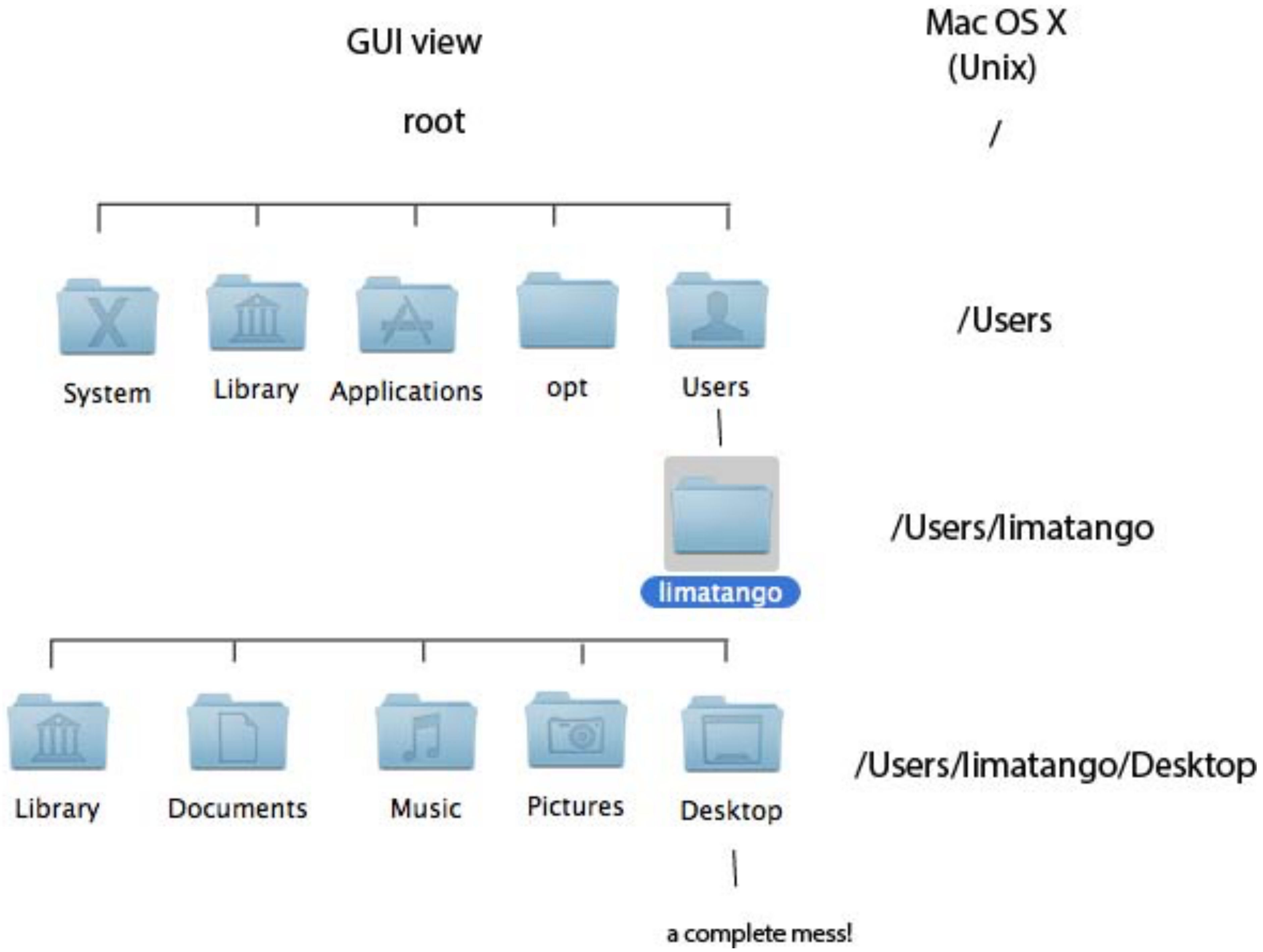
Location **File Name** **Options**

If the file is in the same location as the ipynb file, then use the filename directly

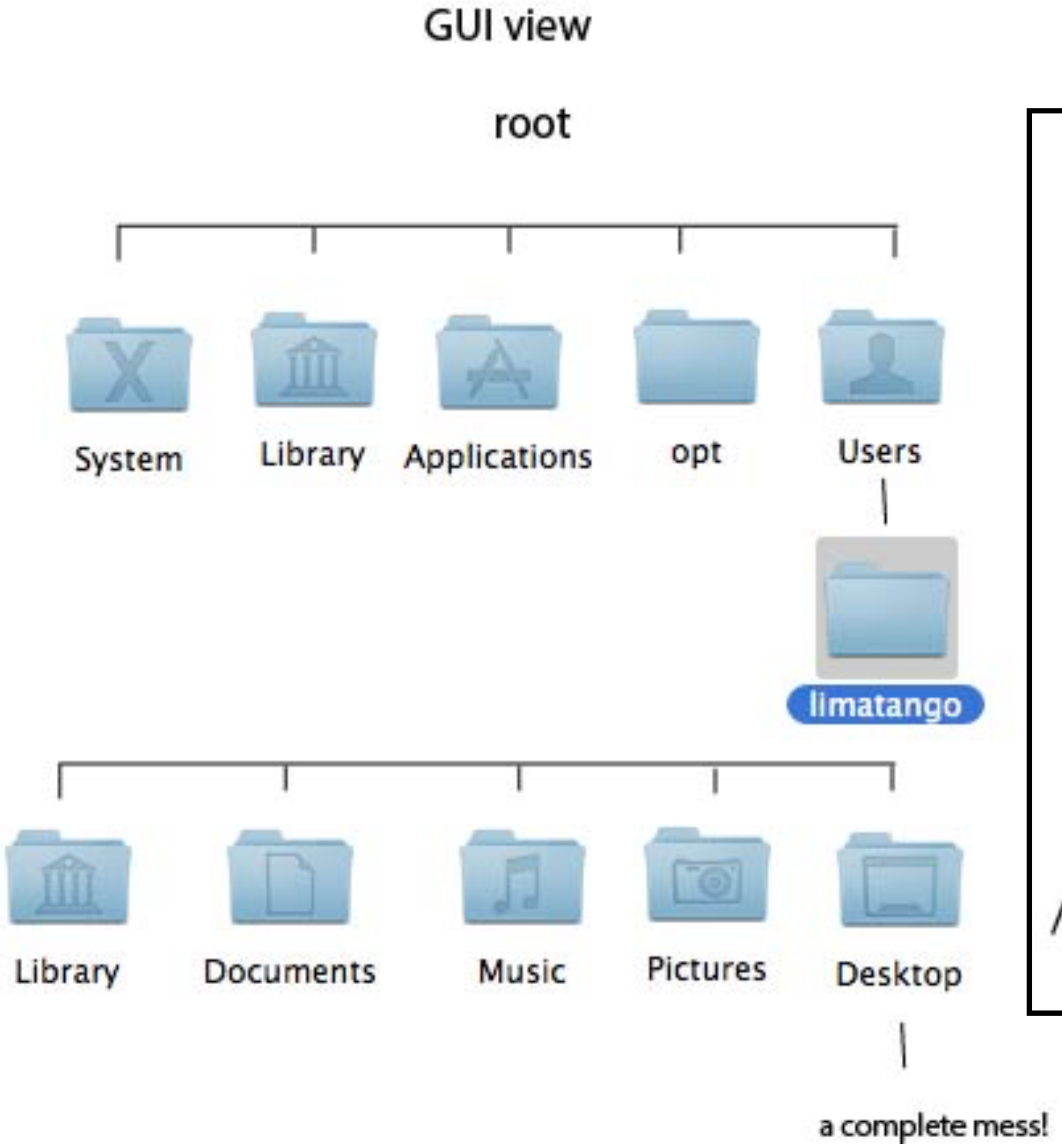
Now let’s learn how to specify locations of your data files (the concept of file systems)

File System of a mac/linux/unix computer

The organization of computers is based on a *file system*. The file system is hierarchical, so at the top you'll find the *root directory* or for Mac and PC users, a *folder*. The root directory contains files and other folders which may also contain files and folders and etc. This continues, resulting in a tree of files and folders that make up the file system. The following figure is an example of a computer's file system:



Concept of Path



Mac OS X
(Unix)

/

/Users

/Users/limatango

/Users/limatango/Desktop

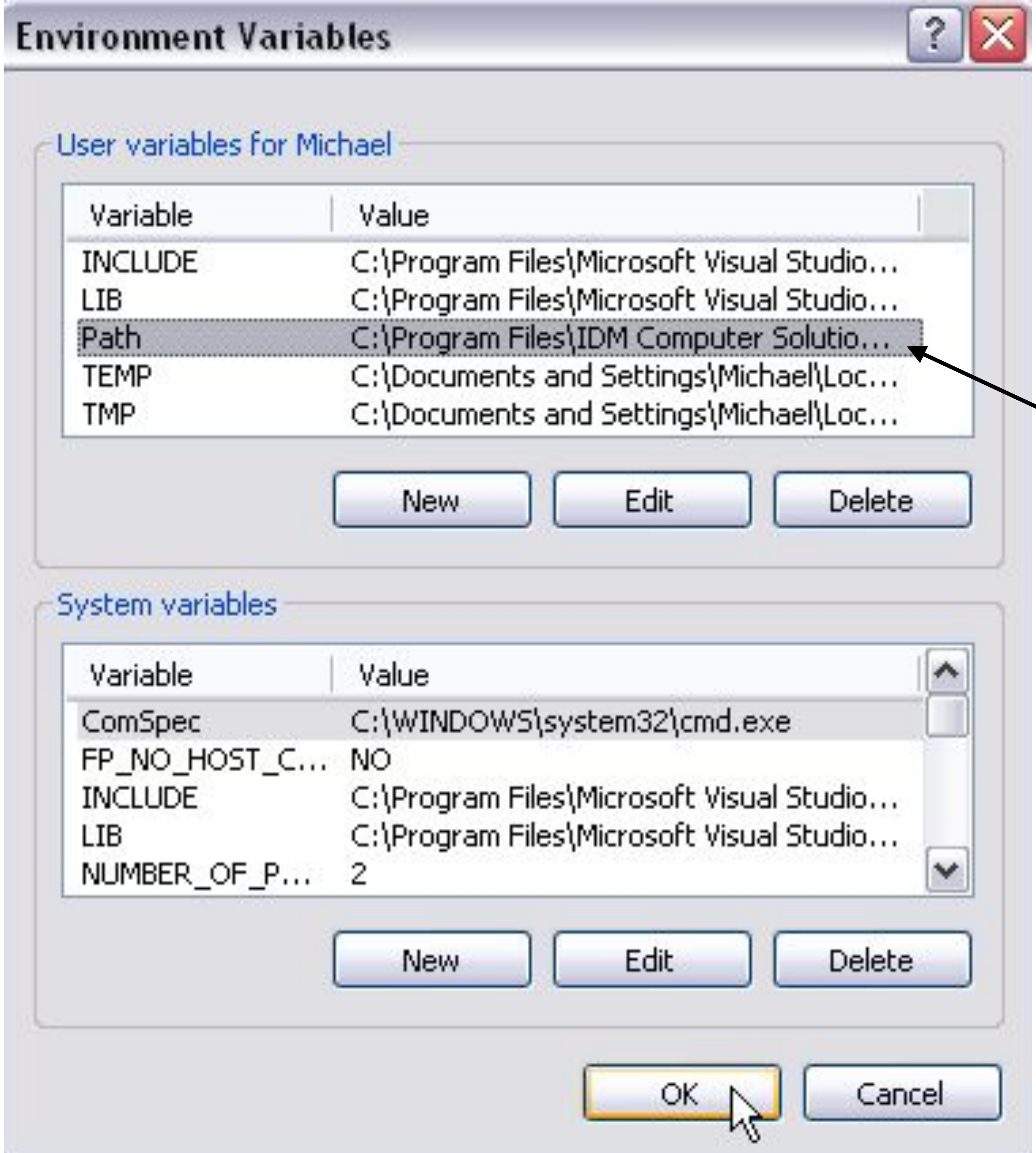
Absolute paths

uniquely define the location of the file or directory from anywhere on the computer.

Relative paths

The *relative paths* are handy short cuts. For example, we can refer to a directory above the current directory without knowing what that is necessarily, we use these conventions:
./ is the current directory
../ is the one above
../../ is the one above that
and so on.

File System of a windows computer



Environment Variables

User variables for Michael

Variable	Value
INCLUDE	C:\Program Files\Microsoft Visual Studio...
LIB	C:\Program Files\Microsoft Visual Studio...
Path	C:\Program Files\IDM Computer Solutio...
TEMP	C:\Documents and Settings\Michael\Loc...
TMP	C:\Documents and Settings\Michael\Loc...

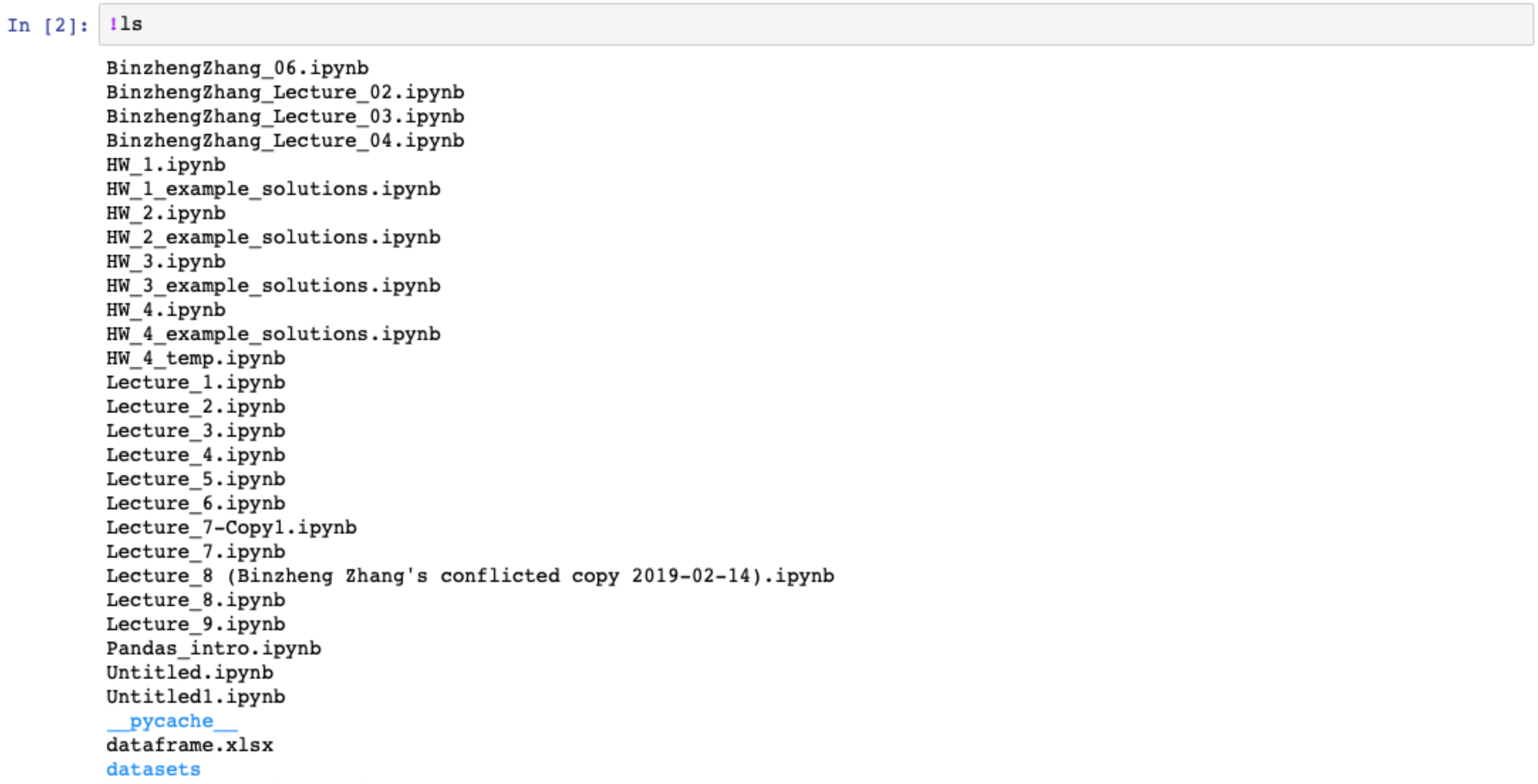
System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
INCLUDE	C:\Program Files\Microsoft Visual Studio...
LIB	C:\Program Files\Microsoft Visual Studio...
NUMBER_OF_P...	2

Absolute Windows Paths

Survival UNIX commands: `ls`

- Macs and PCs both have functions that can be called from a *command line*, such as listing the contents of a folder or file, creating new folders, changing permissions on files or folders, combining the contents of files, moving files and folders around, and so on. These commands are directed to the operating system instead of the Python interpreter.
- Before we begin using commands, we can execute many operating system commands from within a Jupyter notebook. To signal to Jupyter that your commands are not for Python but for the operating system, you may type a "!" (bang) in front of the command.
- Let's learn our first UNIX command, which lists the contents of a directory, **`ls`**. [Note that your output will be different.]



```
In [2]: !ls
```

```
BinzhengZhang_06.ipynb
BinzhengZhang_Lecture_02.ipynb
BinzhengZhang_Lecture_03.ipynb
BinzhengZhang_Lecture_04.ipynb
HW_1.ipynb
HW_1_example_solutions.ipynb
HW_2.ipynb
HW_2_example_solutions.ipynb
HW_3.ipynb
HW_3_example_solutions.ipynb
HW_4.ipynb
HW_4_example_solutions.ipynb
HW_4_temp.ipynb
Lecture_1.ipynb
Lecture_2.ipynb
Lecture_3.ipynb
Lecture_4.ipynb
Lecture_5.ipynb
Lecture_6.ipynb
Lecture_7-Copy1.ipynb
Lecture_7.ipynb
Lecture_8 (Binzheng Zhang's conflicted copy 2019-02-14).ipynb
Lecture_8.ipynb
Lecture_9.ipynb
Pandas_intro.ipynb
Untitled.ipynb
Untitled1.ipynb
__pycache__
dataframe.xlsx
datasets
```

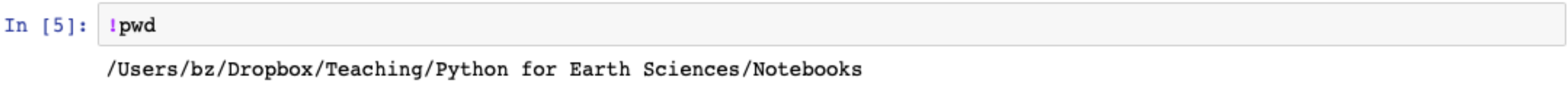
Survival UNIX command to show the path: `pwd`

- A useful command for showing the path of your working directory is **`pwd`** which gives the absolute path of the current directory, pwd means print out working director

Syntax: `pwd`

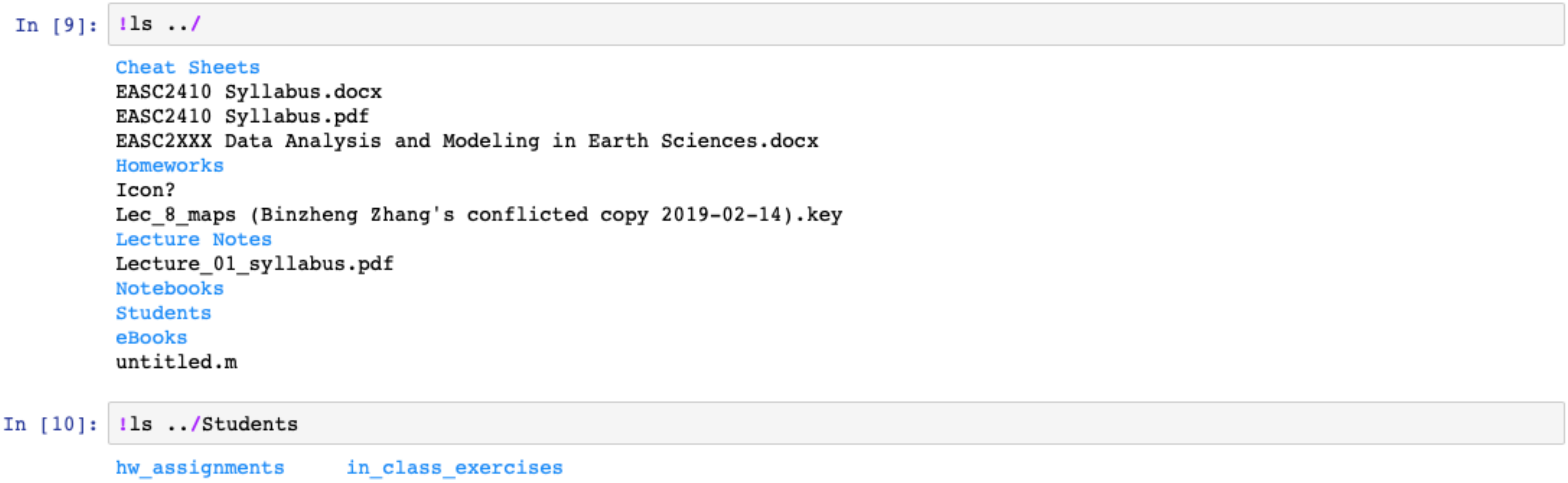
Example: `pwd`

Function: show absolute path of the current working directory



```
In [5]: !pwd

        /Users/bz/Dropbox/Teaching/Python for Earth Sciences/Notebooks
```

Examples of relative directory:



```
In [9]: !ls ../

        Cheat Sheets
        EASC2410 Syllabus.docx
        EASC2410 Syllabus.pdf
        EASC2XXX Data Analysis and Modeling in Earth Sciences.docx
        Homeworks
        Icon?
        Lec_8_maps (Binzheng Zhang's conflicted copy 2019-02-14).key
        Lecture Notes
        Lecture_01_syllabus.pdf
        Notebooks
        Students
        eBooks
        untitled.m

In [10]: !ls ../Students

        hw_assignments      in_class_exercises
```

Survival UNIX commands: `mkdir, rmdir`

- Another useful command is **`mkdir`** which creates a new directory. Please note that *directory* means the same thing as *folder*. It is just that in a graphical operating system with icons, the term *folder* makes sense. They look like folders. Whereas to the operating system, they are traditionally referred to as *directories*.

Syntax: `mkdir DIRECTORY_NAME`

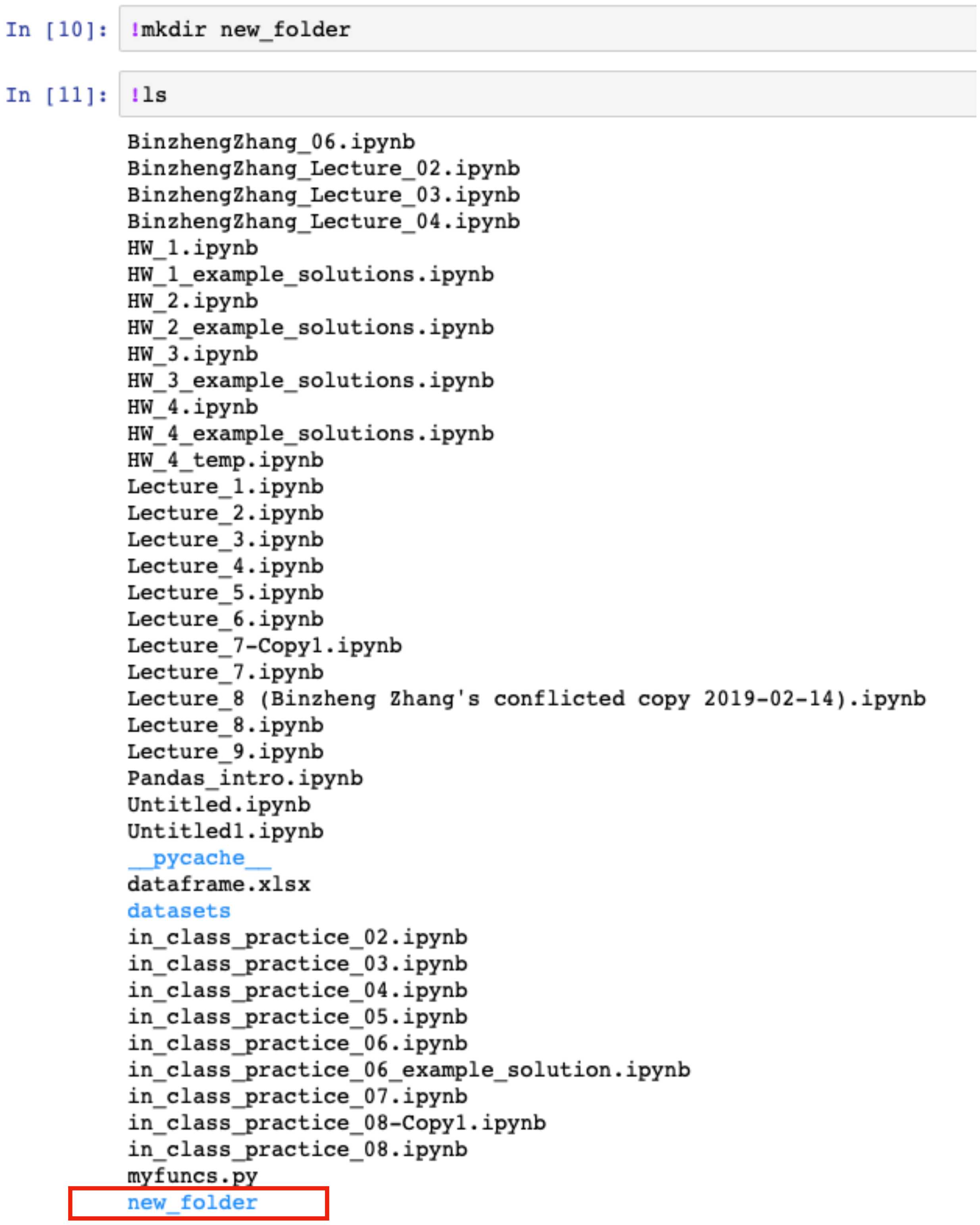
Example: `mkdir new_folder`

Function: make a new director named “new_folder”

Syntax: `rmdir DIRECTORY_NAME`

Example: `rmdir new_folder`

Function: delete a director named “new_folder”



```
In [10]: !mkdir new_folder

In [11]: !ls

         BinzhengZhang_06.ipynb
         BinzhengZhang_Lecture_02.ipynb
         BinzhengZhang_Lecture_03.ipynb
         BinzhengZhang_Lecture_04.ipynb
         HW_1.ipynb
         HW_1_example_solutions.ipynb
         HW_2.ipynb
         HW_2_example_solutions.ipynb
         HW_3.ipynb
         HW_3_example_solutions.ipynb
         HW_4.ipynb
         HW_4_example_solutions.ipynb
         HW_4_temp.ipynb
         Lecture_1.ipynb
         Lecture_2.ipynb
         Lecture_3.ipynb
         Lecture_4.ipynb
         Lecture_5.ipynb
         Lecture_6.ipynb
         Lecture_7-Copy1.ipynb
         Lecture_7.ipynb
         Lecture_8 (Binzheng Zhang's conflicted copy 2019-02-14).ipynb
         Lecture_8.ipynb
         Lecture_9.ipynb
         Pandas_intro.ipynb
         Untitled.ipynb
         Untitled1.ipynb
         __pycache__
         dataframe.xlsx
         datasets
         in_class_practice_02.ipynb
         in_class_practice_03.ipynb
         in_class_practice_04.ipynb
         in_class_practice_05.ipynb
         in_class_practice_06.ipynb
         in_class_practice_06_example_solution.ipynb
         in_class_practice_07.ipynb
         in_class_practice_08-Copy1.ipynb
         in_class_practice_08.ipynb
         myfuncs.py
         new_folder
```

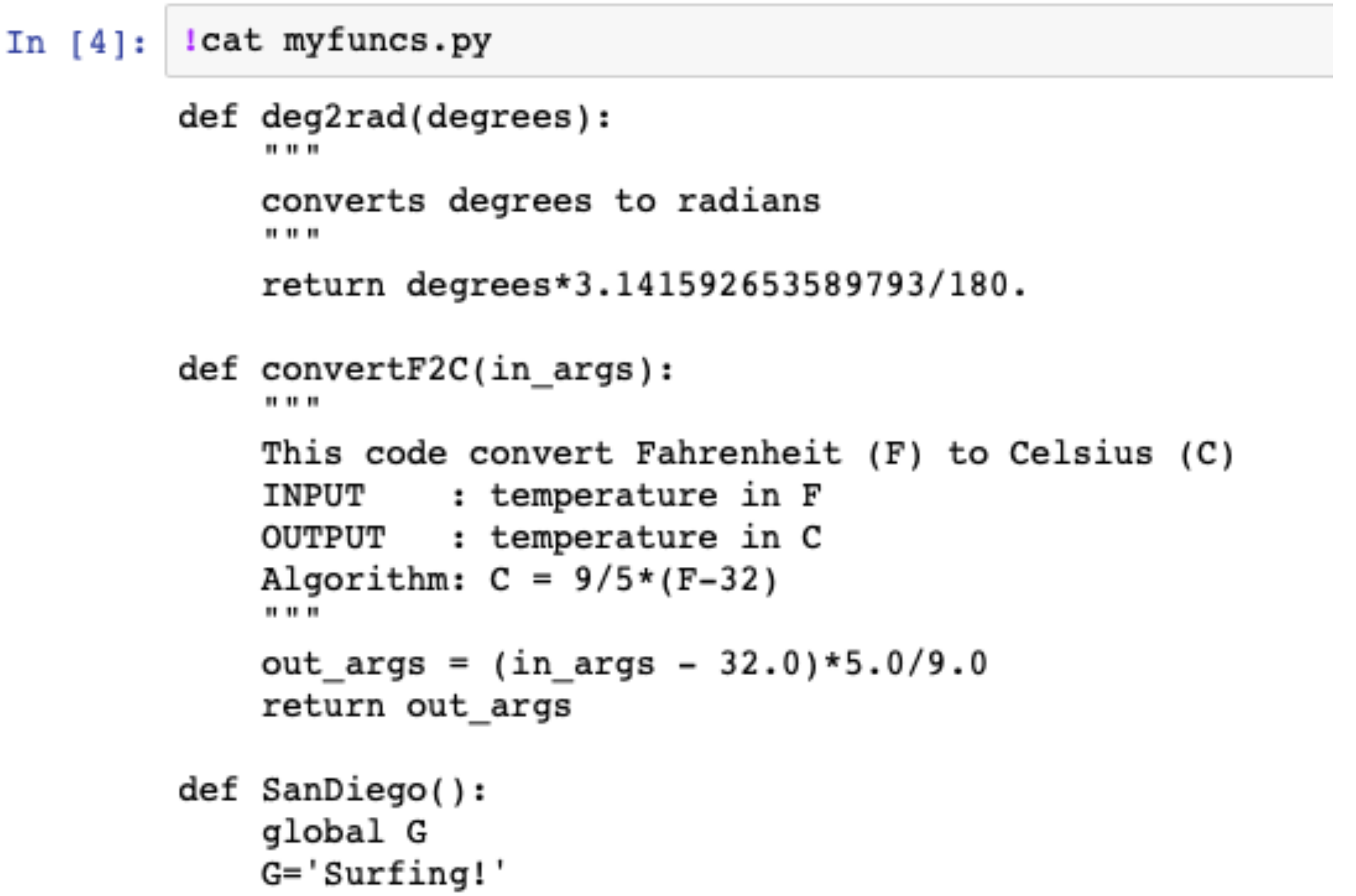
Survival UNIX commands: `cat, rm`

- Another useful command is **`cat`** which lists the contents of a file with the UNIX command, **cat** comes from concatenate.

Syntax: `cat file_name`

Example: `cat myfuncs.py`

Function: show the contents of a file



```
In [4]: !cat myfuncs.py

def deg2rad(degrees):
    """
    converts degrees to radians
    """
    return degrees*3.141592653589793/180.

def convertF2C(in_args):
    """
    This code convert Fahrenheit (F) to Celsius (C)
    INPUT    : temperature in F
    OUTPUT   : temperature in C
    Algorithm: C = 9/5*(F-32)
    """
    out_args = (in_args - 32.0)*5.0/9.0
    return out_args

def SanDiego():
    global G
    G='Surfing!'
```

Syntax: `rm File_NAME`

Example: `rm myfuncs.py`

Function: delete a file named "file_name"

Be careful when using rm!

Try head and tail in the practice problems

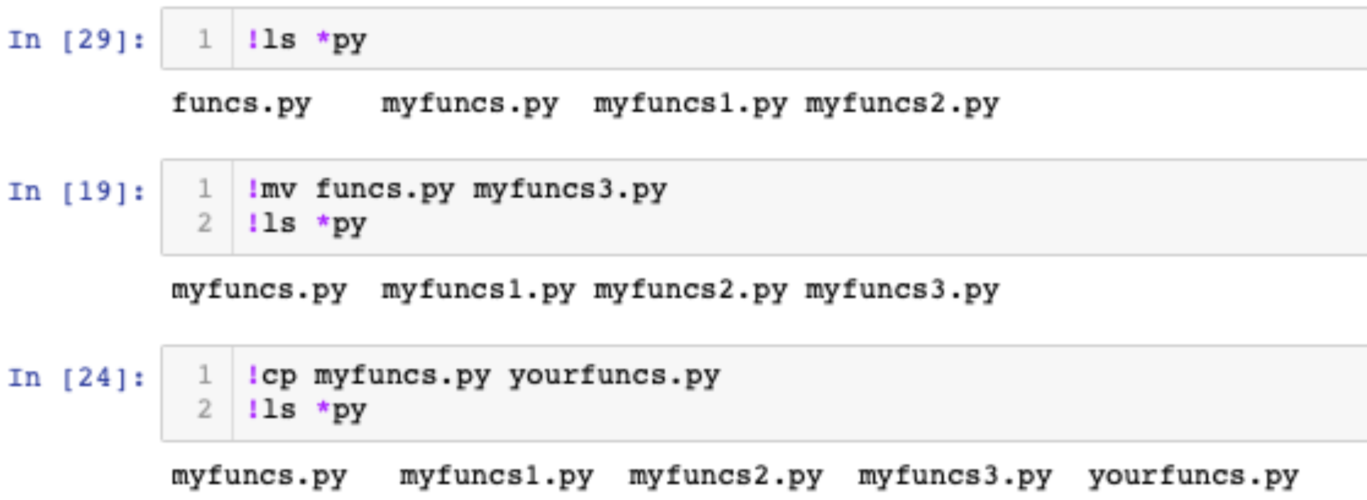
Survival UNIX commands: `cp, mv`

- Another useful command is **`cat`** which lists the contents of a file with the UNIX command, **cat** comes from concatenate.

Syntax: `cp file_name1 file_name2`

Example: `cp myfuncs.py funds.py`

Function: copy the file_name1 to be a new file called file_name2 (creating anew file)



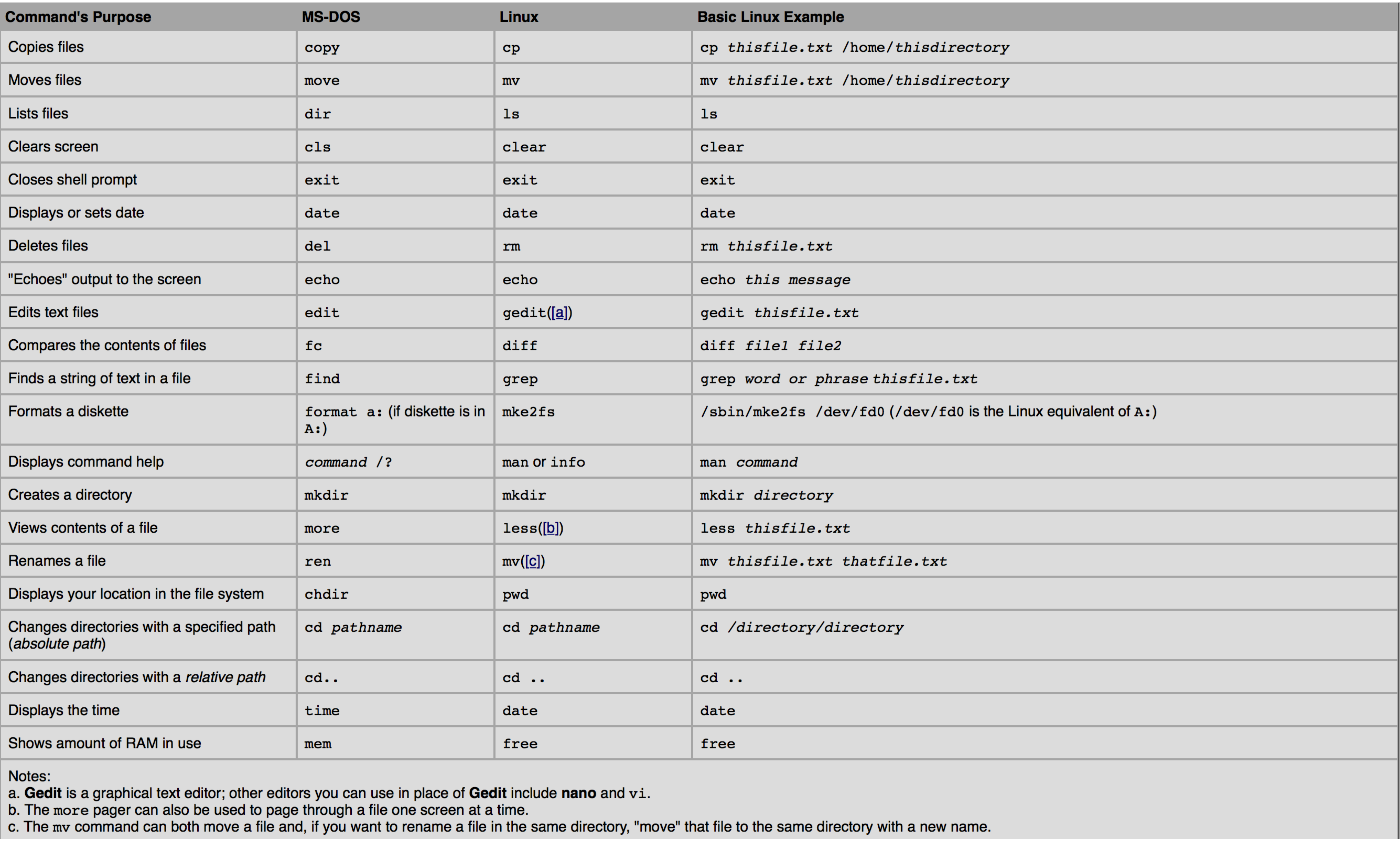
```
In [29]: !ls *py

funcs.py     myfuncs.py   myfuncs1.py myfuncs2.py

In [19]: !mv funcs.py myfuncs3.py
         !ls *py

myfuncs.py  myfuncs1.py myfuncs2.py myfuncs3.py

In [24]: !cp myfuncs.py yourfuncs.py
         !ls *py

myfuncs.py   myfuncs1.py  myfuncs2.py  myfuncs3.py  yourfuncs.py
```

Syntax: `mv file_name1 file_name2`

Example: `mv funcs.py myfuncs1.py`

Function: rename the file_name1 to be a file_name2 (not creating a new file)

cmd commands (Windows) vs Linux/Unix/Mac



Command's Purpose	MS-DOS	Linux	Basic Linux Example
Copies files	copy	cp	cp *thisfile.txt* /home/*thisdirectory*
Moves files	move	mv	mv *thisfile.txt* /home/*thisdirectory*
Lists files	dir	ls	ls
Clears screen	cls	clear	clear
Closes shell prompt	exit	exit	exit
Displays or sets date	date	date	date
Deletes files	del	rm	rm *thisfile.txt*
"Echoes" output to the screen	echo	echo	echo *this message*
Edits text files	edit	gedit([a])	gedit *thisfile.txt*
Compares the contents of files	fc	diff	diff *file1 file2*
Finds a string of text in a file	find	grep	grep *word or phrase thisfile.txt*
Formats a diskette	format a: (if diskette is in A:)	mke2fs	/sbin/mke2fs /dev/fd0 (/dev/fd0 is the Linux equivalent of A:)
Displays command help	*command* /?	man or info	man *command*
Creates a directory	mkdir	mkdir	mkdir *directory*
Views contents of a file	more	less([b])	less *thisfile.txt*
Renames a file	ren	mv([c])	mv *thisfile.txt thatfile.txt*
Displays your location in the file system	chdir	pwd	pwd
Changes directories with a specified path (*absolute path*)	cd *pathname*	cd *pathname*	cd */directory/directory*
Changes directories with a *relative path*	cd..	cd ..	cd ..
Displays the time	time	date	date
Shows amount of RAM in use	mem	free	free

Notes:
a. **Gedit** is a graphical text editor; other editors you can use in place of **Gedit** include **nano** and vi.
b. The more pager can also be used to page through a file one screen at a time.
c. The mv command can both move a file and, if you want to rename a file in the same directory, "move" that file to the same directory with a new name.

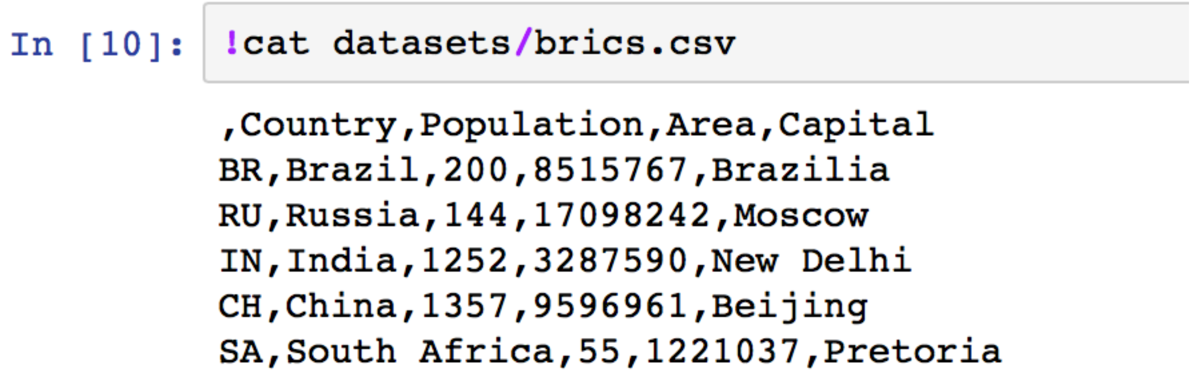
Introduction to Pandas

Overview

- Huge amounts of data are common in data analysis/data science
- Can use 2-D Numpy arrays (recall HW examples)
  - Only one type of data allowed!
- Pandas
  - High-level data manipulation
  - DataFrame

Load .csv files using Pandas

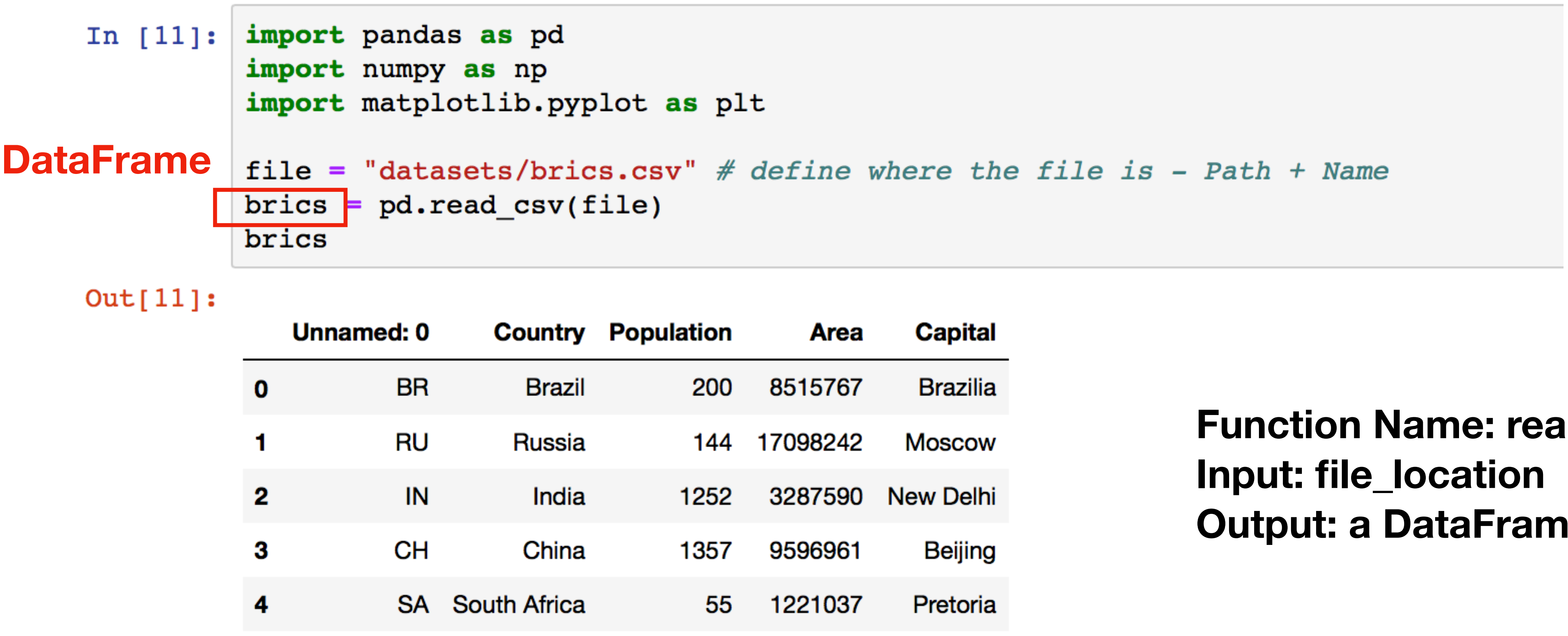
Now we have the following data file named “brics.csv”, let’s cat the data file to see what it looks like:



```
In [10]: !cat datasets/brics.csv

         ,Country,Population,Area,Capital
         BR,Brazil,200,8515767,Brazilia
         RU,Russia,144,17098242,Moscow
         IN,India,1252,3287590,New Delhi
         CH,China,1357,9596961,Beijing
         SA,South Africa,55,1221037,Pretoria
```

The “brics.csv” file is a 2-D data table with informations of different data types. It is not very convenient to use the NumPy functions such as np.loadtxt() to import the data. Why?

Now we use the Pandas library:



```
In [11]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt

         file = "datasets/brics.csv" # define where the file is - Path + Name
         brics = pd.read_csv(file)
         brics
```

DataFrame

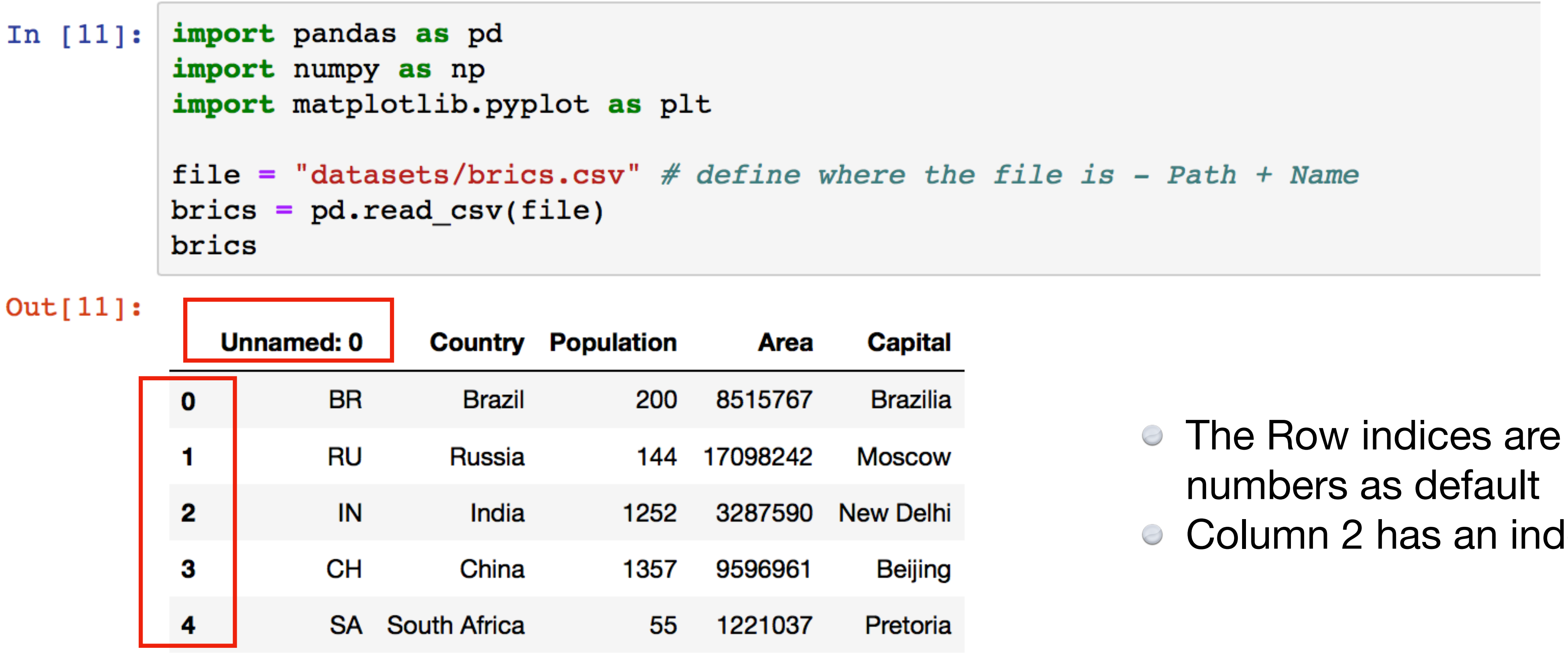
Out[11]:

	Unnamed: 0	Country	Population	Area	Capital
0	BR	Brazil	200	8515767	Brazilia
1	RU	Russia	144	17098242	Moscow
2	IN	India	1252	3287590	New Delhi
3	CH	China	1357	9596961	Beijing
4	SA	South Africa	55	1221037	Pretoria

Function Name: read_csv()
Input: file_location
Output: a DataFrame named brics

Load .csv files using Pandas

But the DataFrame looks a bit weird:



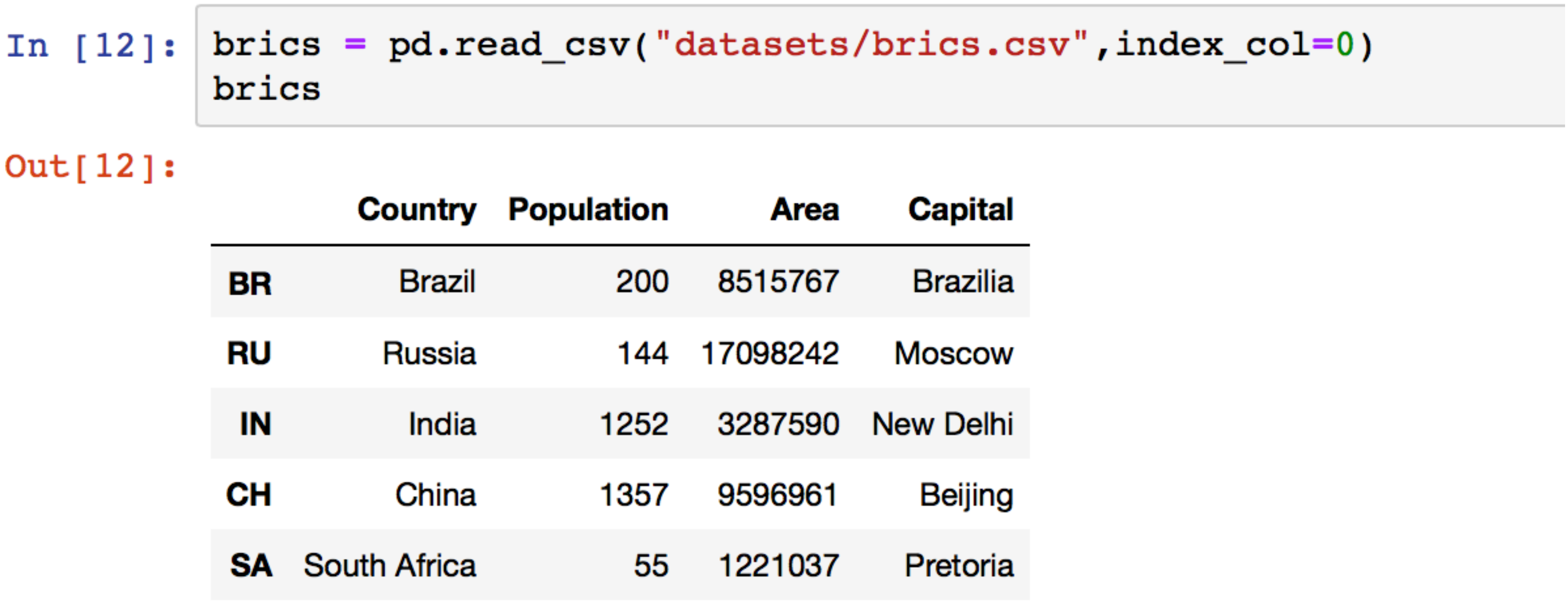
```
In [11]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt

         file = "datasets/brics.csv" # define where the file is - Path + Name
         brics = pd.read_csv(file)
         brics
```

Out[11]:

	Unnamed: 0	Country	Population	Area	Capital
0	BR	Brazil	200	8515767	Brazilia
1	RU	Russia	144	17098242	Moscow
2	IN	India	1252	3287590	New Delhi
3	CH	China	1357	9596961	Beijing
4	SA	South Africa	55	1221037	Pretoria

- The Row indices are set as numeric numbers as default
- Column 2 has an index “Unnamed:0”

To fix it, let’s tell Pandas that we want the first column to be used as the row indices:

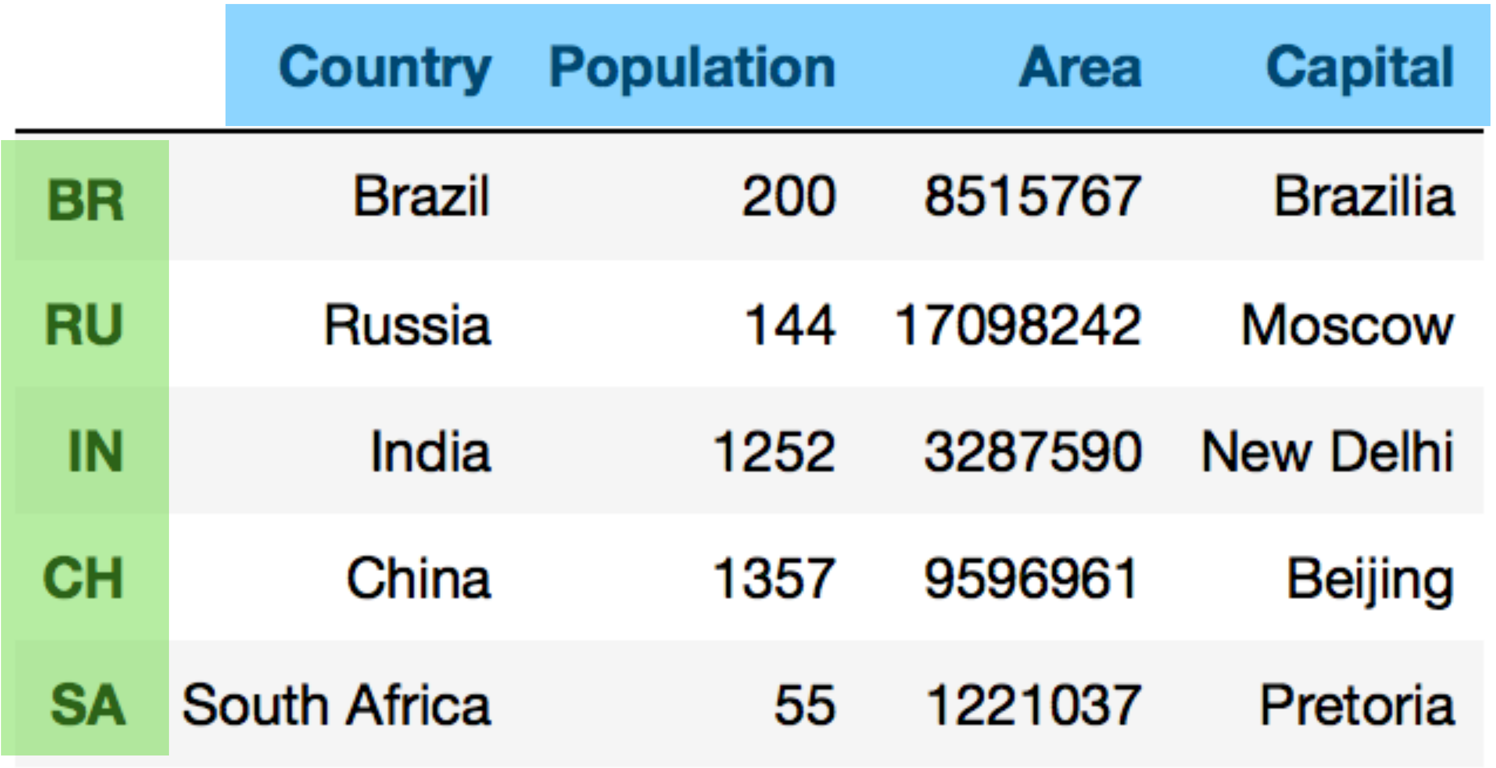


```
In [12]: brics = pd.read_csv("datasets/brics.csv",index_col=0)
         brics
```

Out[12]:

	Country	Population	Area	Capital
BR	Brazil	200	8515767	Brazilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

DataFrame



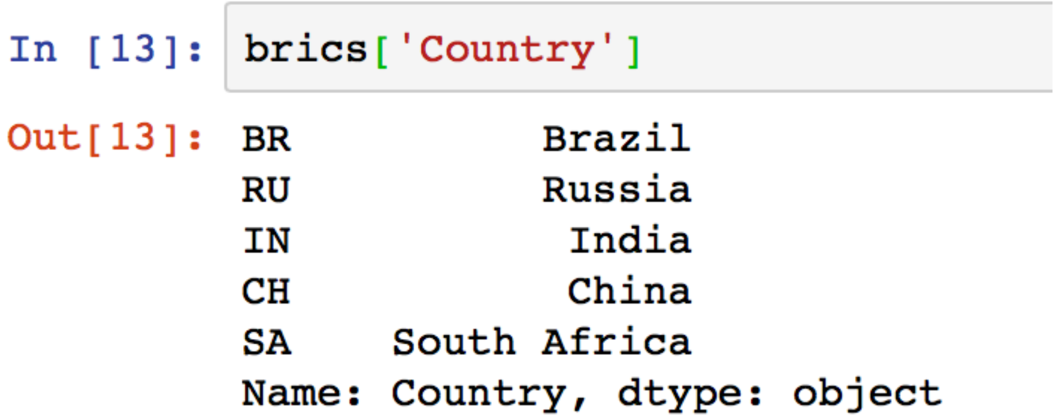
	Country	Population	Area	Capital
BR	Brazil	200	8515767	Brazilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

column index

row index

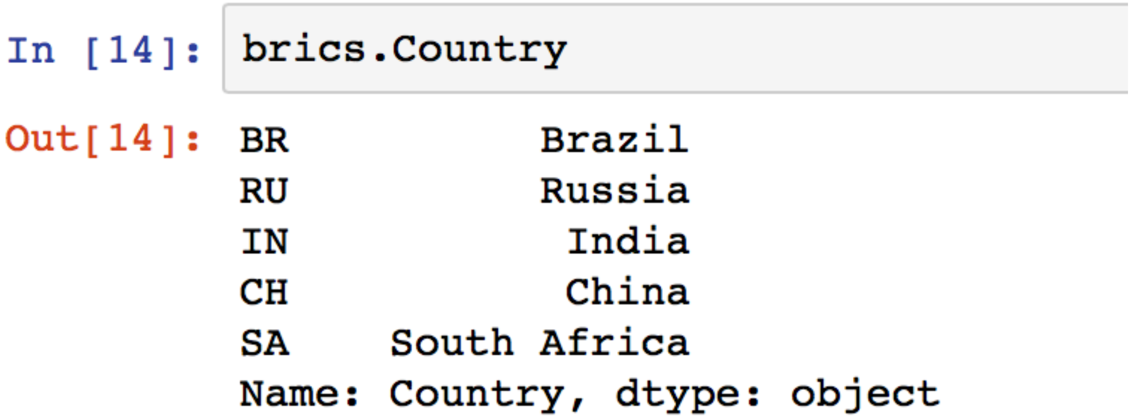
Column Access

method 1:



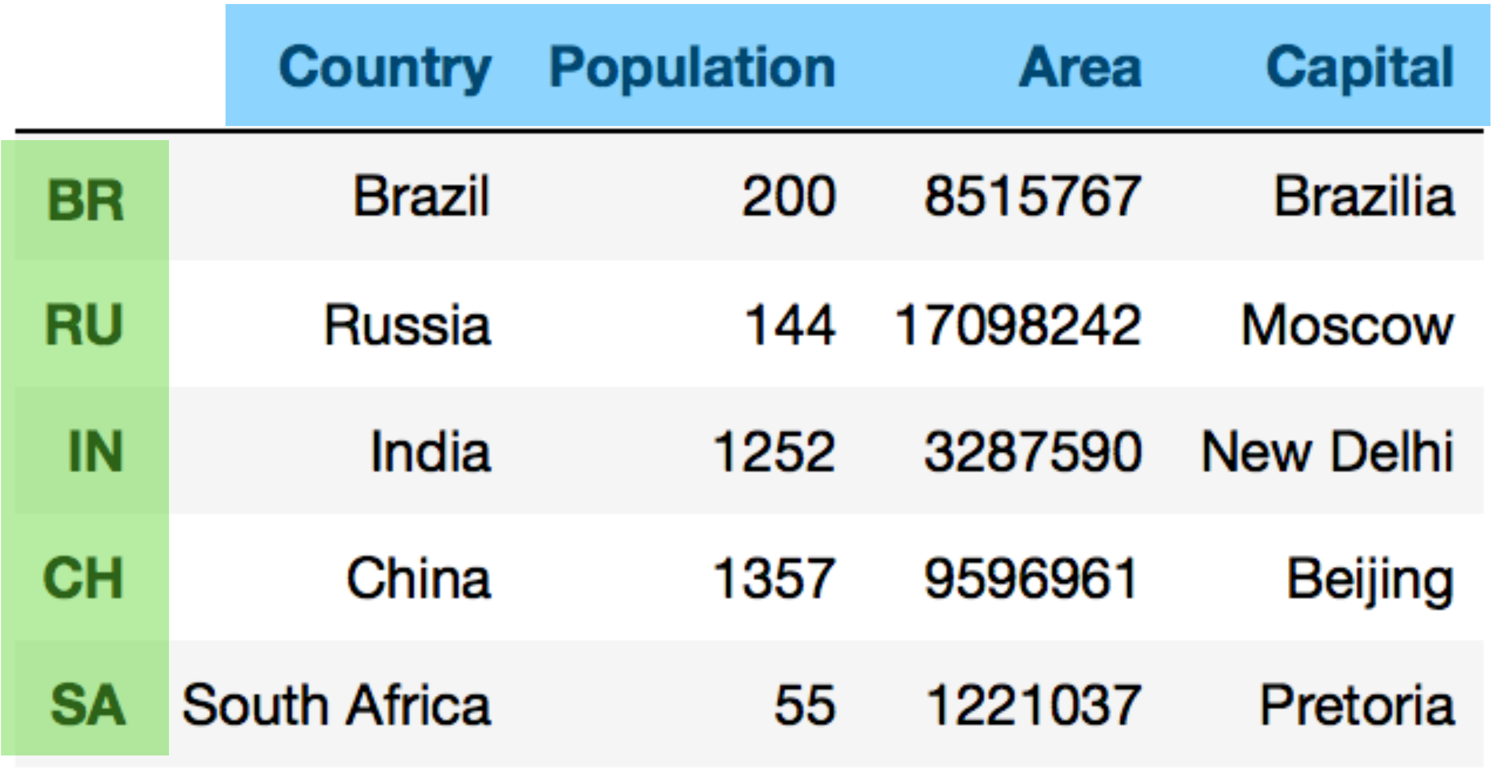
```
In [13]: brics['Country']

Out[13]: BR          Brazil
         RU          Russia
         IN           India
         CH           China
         SA    South Africa
         Name: Country, dtype: object
```

method 2:



```
In [14]: brics.Country

Out[14]: BR          Brazil
         RU          Russia
         IN           India
         CH           China
         SA    South Africa
         Name: Country, dtype: object
```

DataFrame

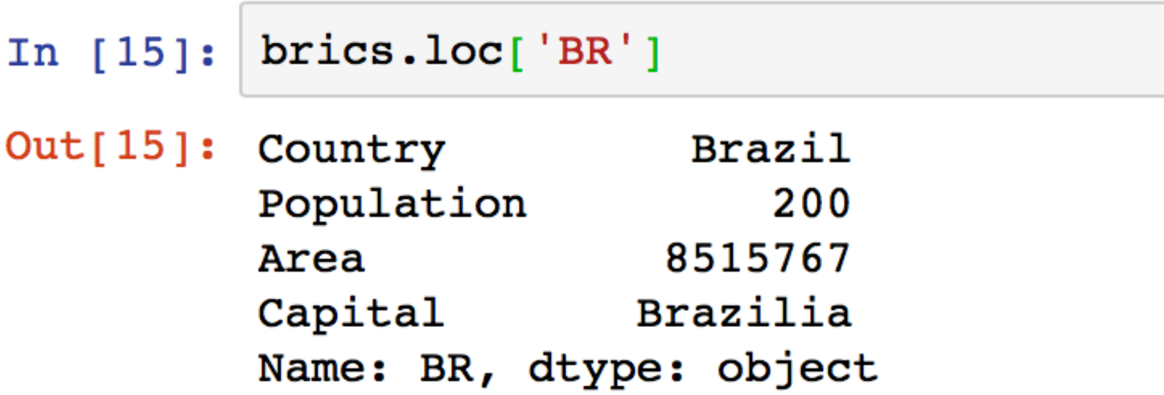


	Country	Population	Area	Capital
BR	Brazil	200	8515767	Brazilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

column index

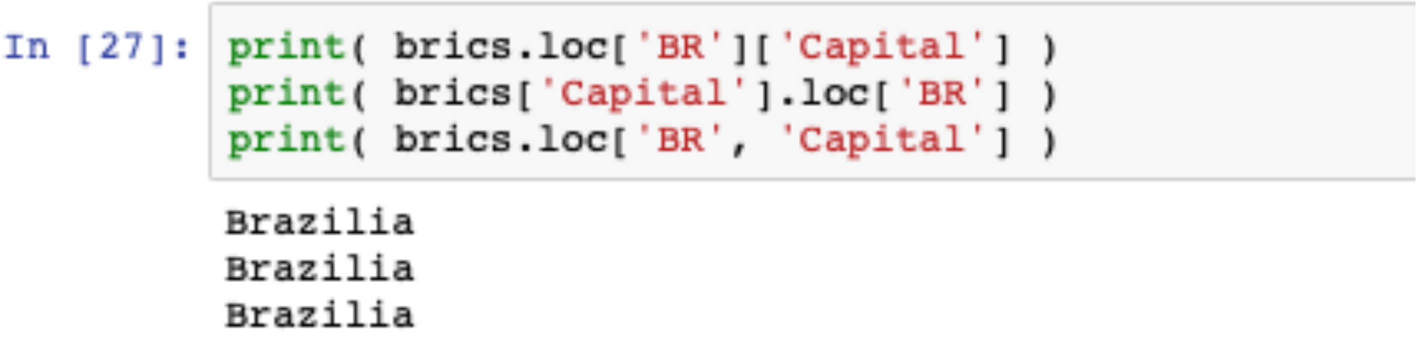
row index

Row Access



```
In [15]: brics.loc['BR']

Out[15]: Country          Brazil
         Population          200
         Area            8515767
         Capital        Brazilia
         Name: BR, dtype: object
```

Element Access

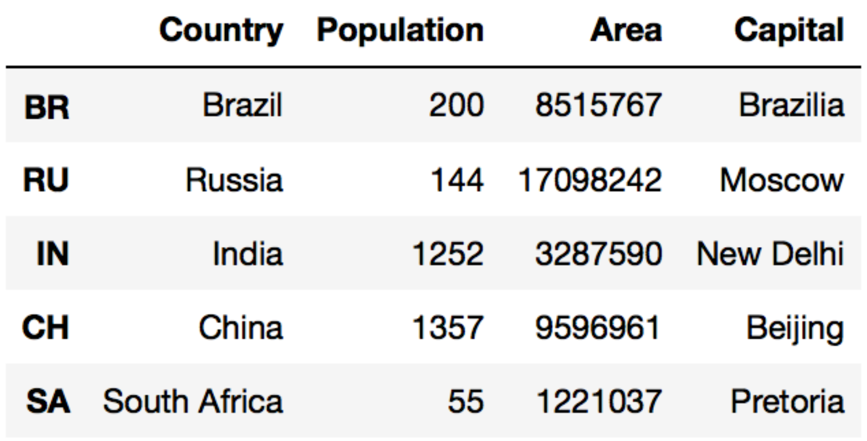


```
In [27]: print( brics.loc['BR']['Capital'] )
         print( brics['Capital'].loc['BR'] )
         print( brics.loc['BR', 'Capital'] )

         Brazilia
         Brazilia
         Brazilia
```

Note: to access a column data in a Pandas DataFrame, you can use the name of the index directly; however, to access a row data in a DataFrame, you need to use functions such as the `.loc()`,

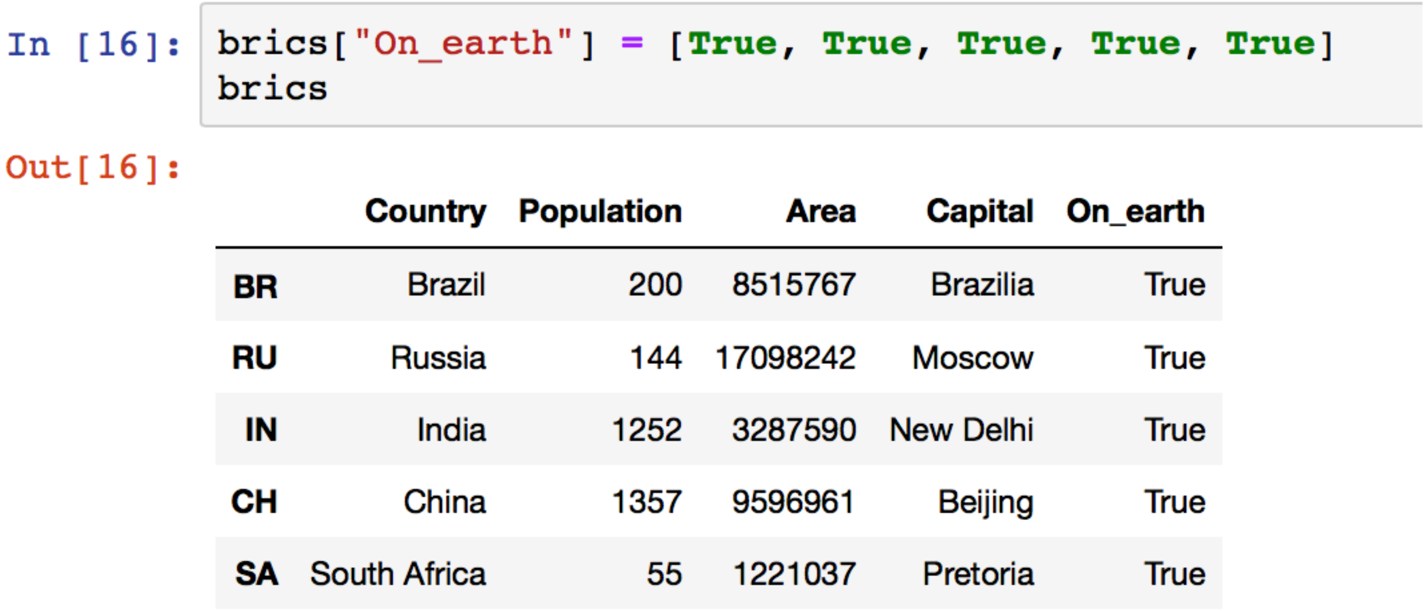
Adding a new column to an existing DataFrame

Given the existing DataFrame:



	Country	Population	Area	Capital
BR	Brazil	200	8515767	Brazilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

You can add column data simply by:

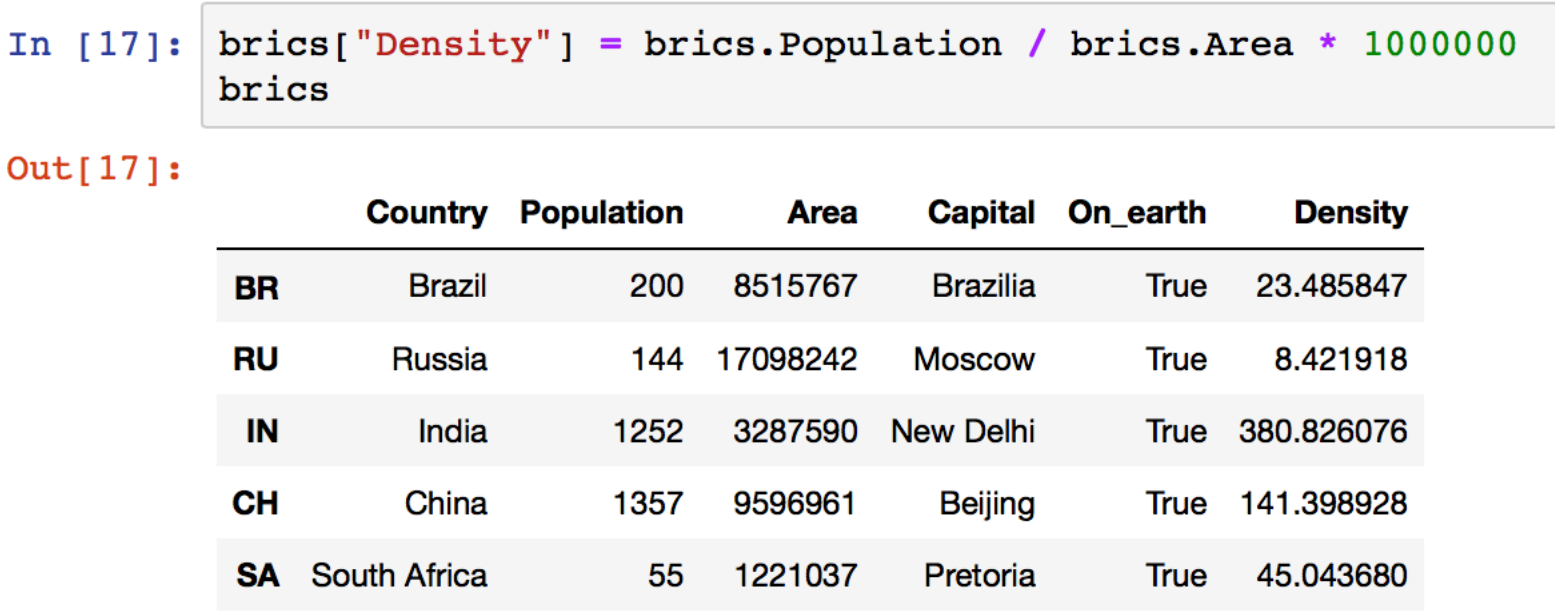


```
In [16]: brics["On_earth"] = [True, True, True, True, True]
         brics
```

Out[16]:

	Country	Population	Area	Capital	On_earth
BR	Brazil	200	8515767	Brazilia	True
RU	Russia	144	17098242	Moscow	True
IN	India	1252	3287590	New Delhi	True
CH	China	1357	9596961	Beijing	True
SA	South Africa	55	1221037	Pretoria	True

Or adding using NumPy array operations



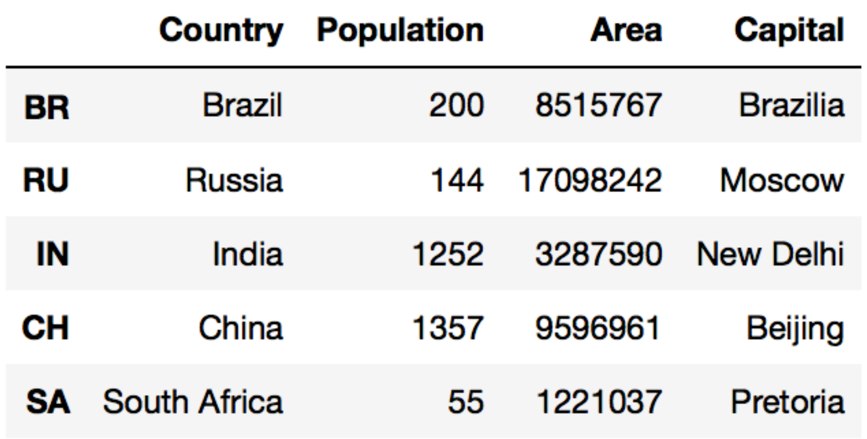
```
In [17]: brics["Density"] = brics.Population / brics.Area * 1000000
         brics
```

Out[17]:

	Country	Population	Area	Capital	On_earth	Density
BR	Brazil	200	8515767	Brazilia	True	23.485847
RU	Russia	144	17098242	Moscow	True	8.421918
IN	India	1252	3287590	New Delhi	True	380.826076
CH	China	1357	9596961	Beijing	True	141.398928
SA	South Africa	55	1221037	Pretoria	True	45.043680

Rename a column in an existing DataFrame

Given the existing DataFrame:

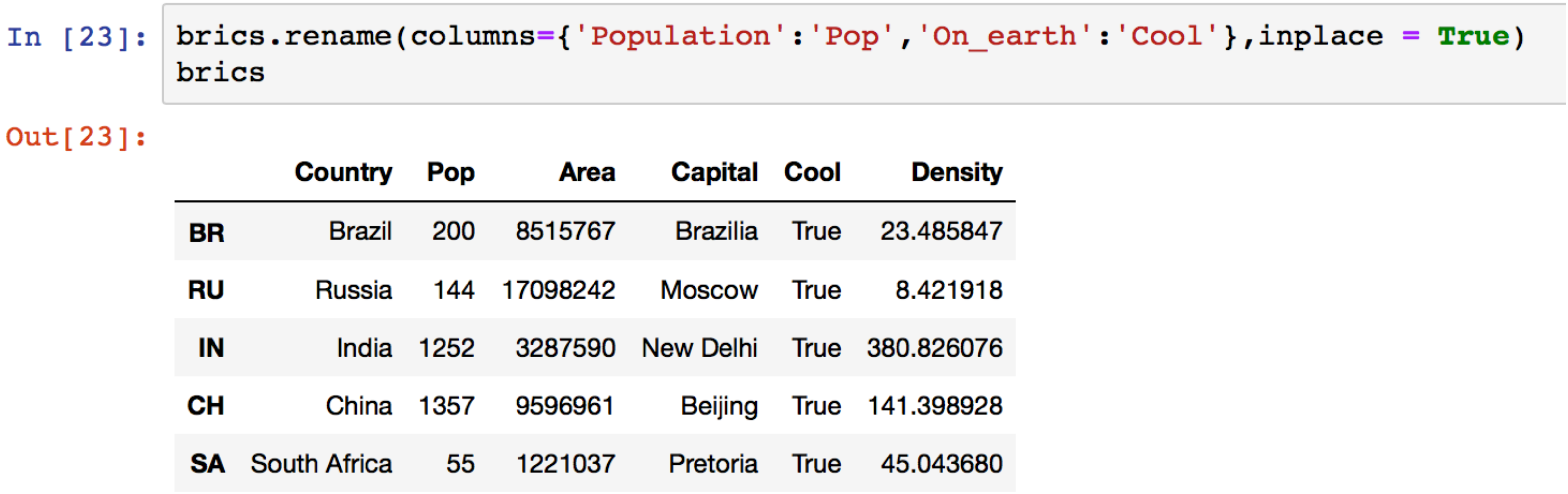


	Country	Population	Area	Capital
BR	Brazil	200	8515767	Brazilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

Syntax:

.rename(columns={‘old_name’:’new_name’}, inplace = True)

You can rename the index of column data using the rename() function:



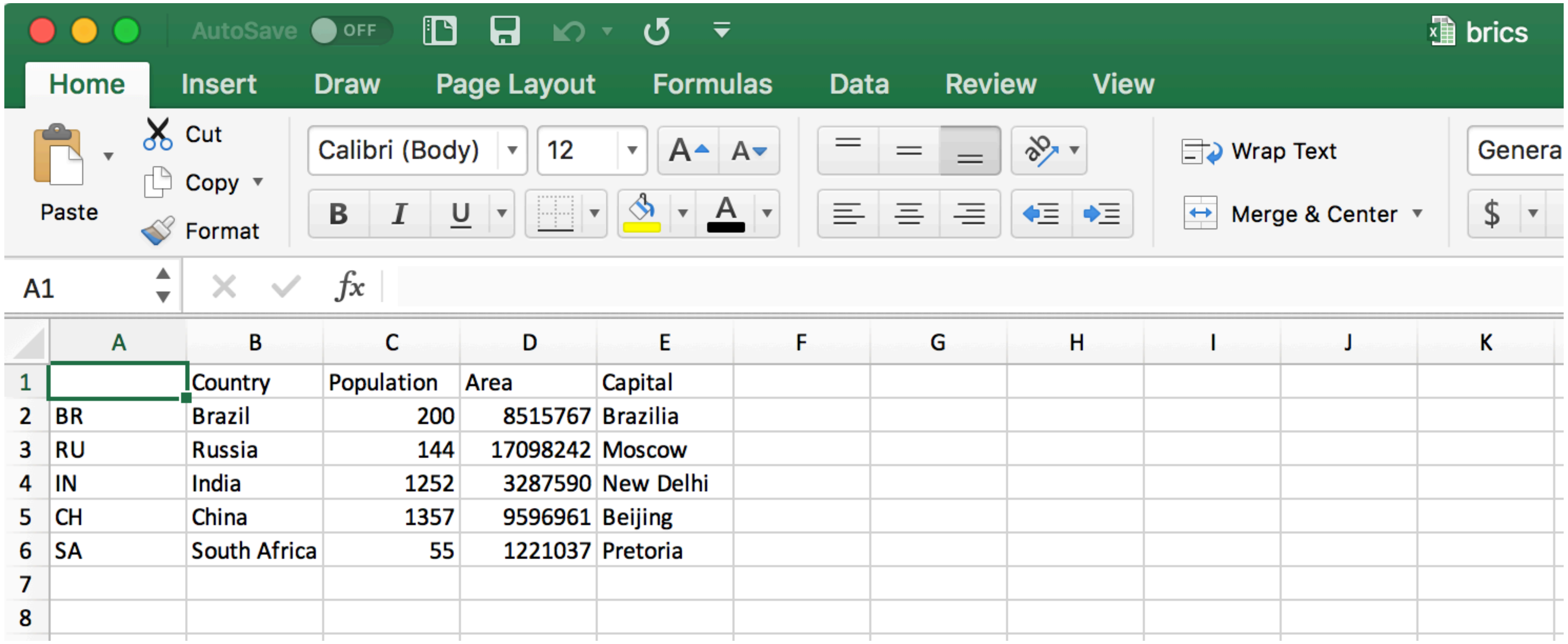
```
In [23]: brics.rename(columns={'Population':'Pop','On_earth':'Cool'},inplace = True)
         brics
```

Out[23]:

	Country	Pop	Area	Capital	Cool	Density
BR	Brazil	200	8515767	Brazilia	True	23.485847
RU	Russia	144	17098242	Moscow	True	8.421918
IN	India	1252	3287590	New Delhi	True	380.826076
CH	China	1357	9596961	Beijing	True	141.398928
SA	South Africa	55	1221037	Pretoria	True	45.043680

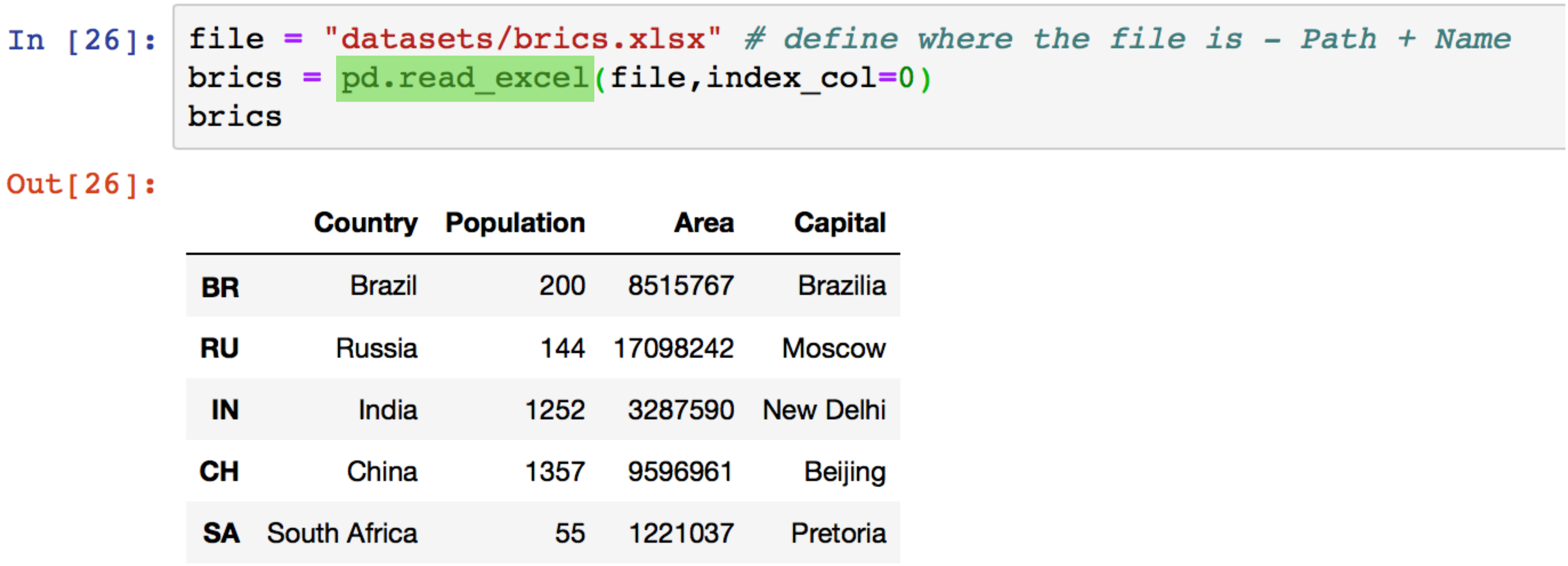
Load Excel files using Pandas

Now we have the brics data collected in an excel file named brics.xlsx



	A	B	C	D	E
1		Country	Population	Area	Capital
2	BR	Brazil	200	8515767	Brazilia
3	RU	Russia	144	17098242	Moscow
4	IN	India	1252	3287590	New Delhi
5	CH	China	1357	9596961	Beijing
6	SA	South Africa	55	1221037	Pretoria
7					
8					

We can load the data in brics.xlsx using Pandas:



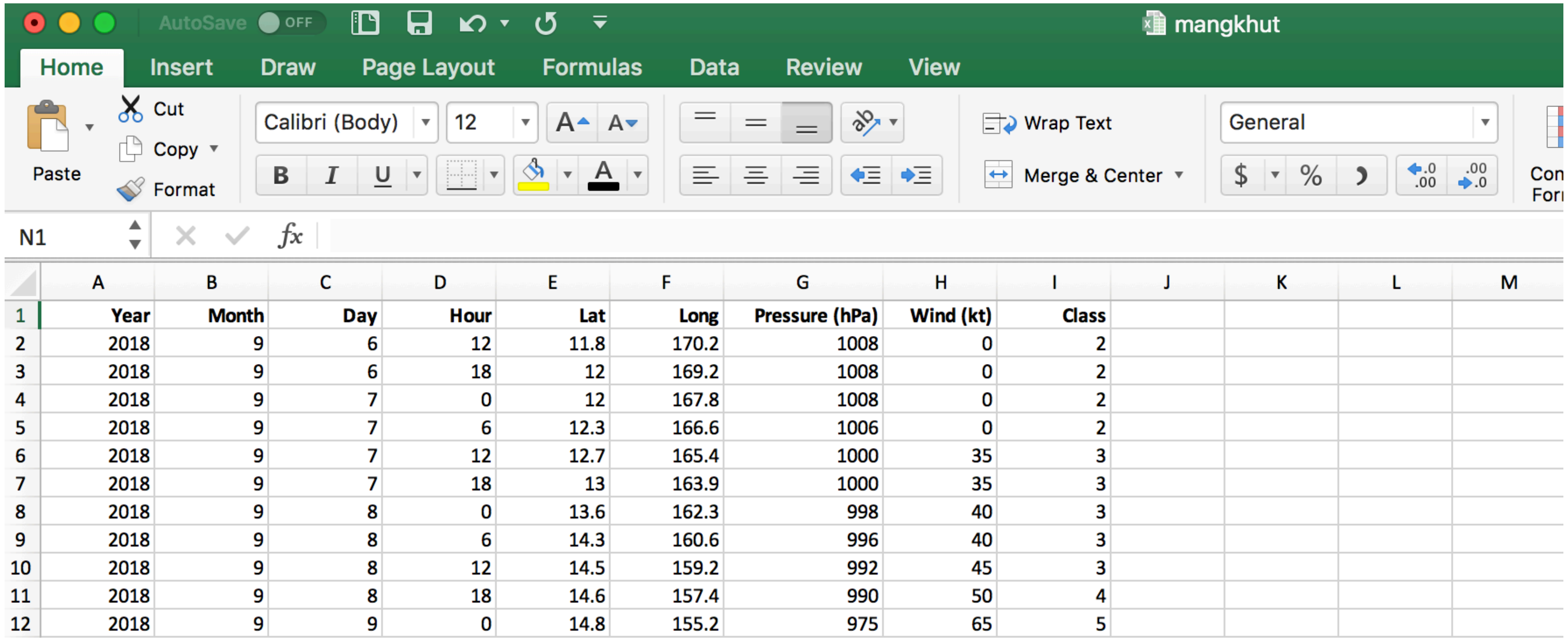
```
In [26]: file = "datasets/brics.xlsx" # define where the file is - Path + Name
         brics = pd.read_excel(file,index_col=0)
         brics
```

Out[26]:

	Country	Population	Area	Capital
BR	Brazil	200	8515767	Brazilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

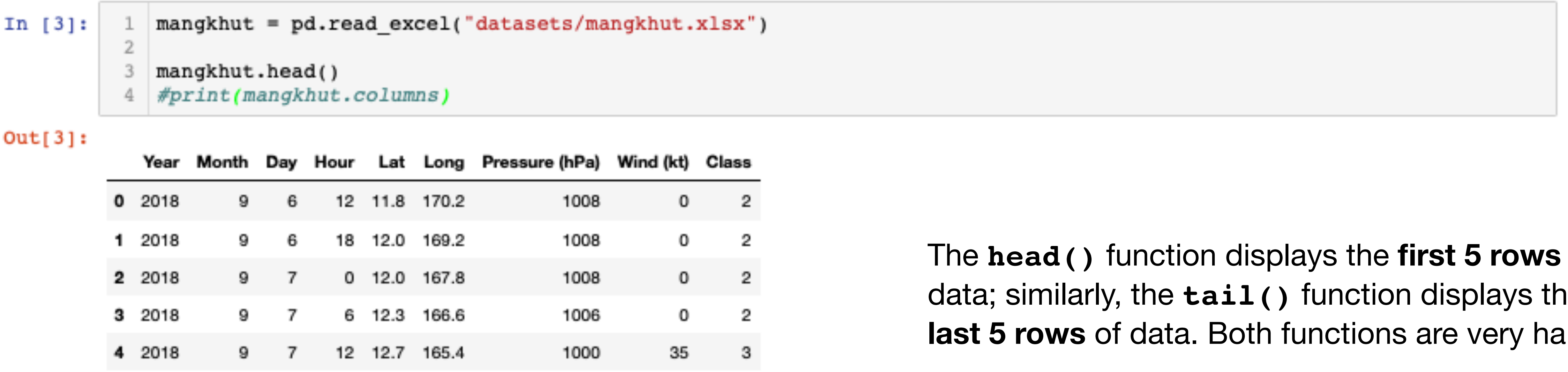
Now let’s revisit the Hurricane Mangkhut data file

Now I have saved the track data of Hurricane Mangkhut into an excel file which looks like:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Year	Month	Day	Hour	Lat	Long	Pressure (hPa)	Wind (kt)	Class				
2	2018	9	6	12	11.8	170.2	1008	0	2				
3	2018	9	6	18	12	169.2	1008	0	2				
4	2018	9	7	0	12	167.8	1008	0	2				
5	2018	9	7	6	12.3	166.6	1006	0	2				
6	2018	9	7	12	12.7	165.4	1000	35	3				
7	2018	9	7	18	13	163.9	1000	35	3				
8	2018	9	8	0	13.6	162.3	998	40	3				
9	2018	9	8	6	14.3	160.6	996	40	3				
10	2018	9	8	12	14.5	159.2	992	45	3				
11	2018	9	8	18	14.6	157.4	990	50	4				
12	2018	9	9	0	14.8	155.2	975	65	5				

Let’s load the data file into a dataFrame called mangkhut:



In [3]:

```
mangkhut = pd.read_excel("datasets/mangkhut.xlsx")

mangkhut.head()
#print(mangkhut.columns)
```

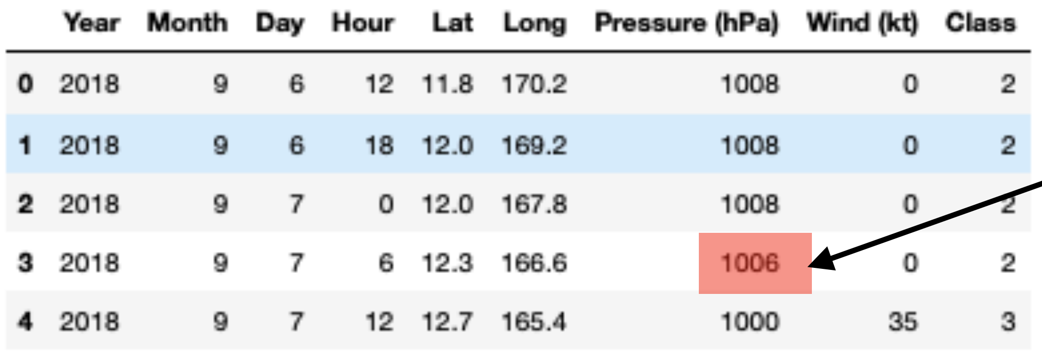
Out[3]:

	Year	Month	Day	Hour	Lat	Long	Pressure (hPa)	Wind (kt)	Class
0	2018	9	6	12	11.8	170.2	1008	0	2
1	2018	9	6	18	12.0	169.2	1008	0	2
2	2018	9	7	0	12.0	167.8	1008	0	2
3	2018	9	7	6	12.3	166.6	1006	0	2
4	2018	9	7	12	12.7	165.4	1000	35	3

The `head()` function displays the **first 5 rows** of data; similarly, the `tail()` function displays the **last 5 rows** of data. Both functions are very handy

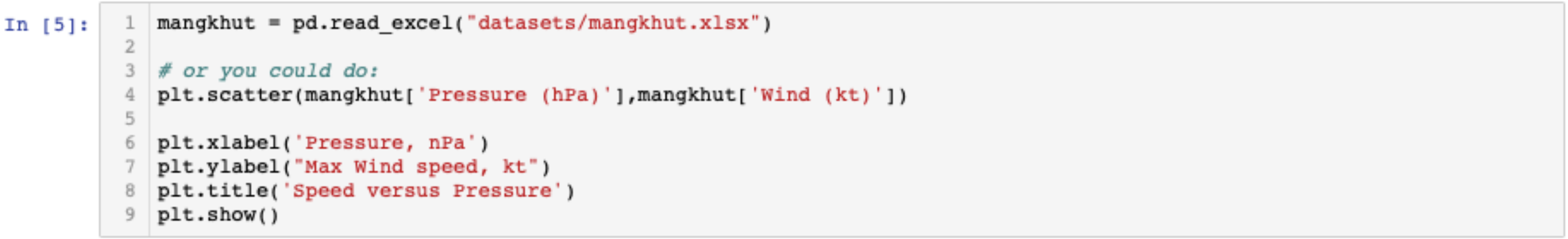
Now let's revisit the Hurricane Mangkhut data file

Let's access the column data of Pressure and Wind to make a plot showing the relationship between the two:

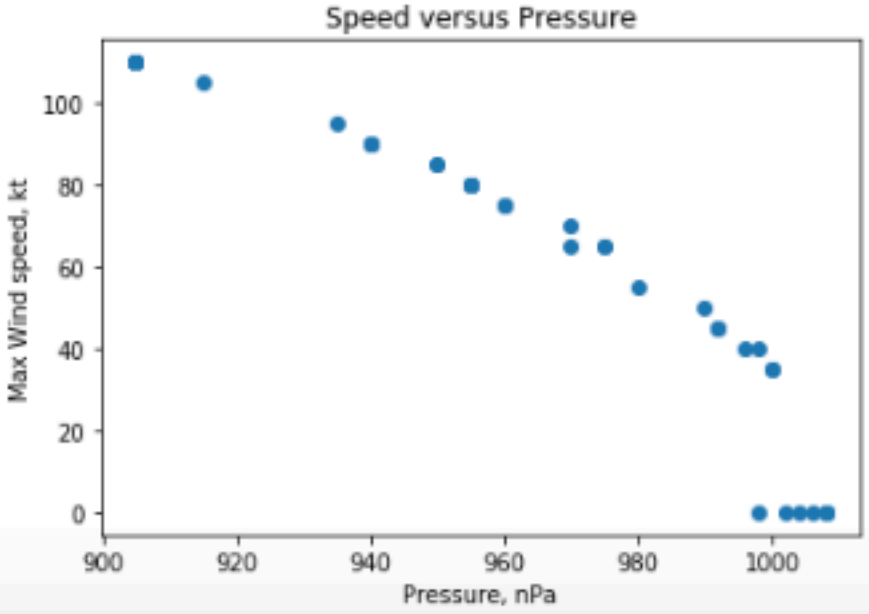


	Year	Month	Day	Hour	Lat	Long	Pressure (hPa)	Wind (kt)	Class
0	2018	9	6	12	11.8	170.2	1008	0	2
1	2018	9	6	18	12.0	169.2	1008	0	2
2	2018	9	7	0	12.0	167.8	1008	0	2
3	2018	9	7	6	12.3	166.6	1006	0	2
4	2018	9	7	12	12.7	165.4	1000	35	3

Question: How to access this number?



```
In [5]:
mangkhut = pd.read_excel("datasets/mangkhut.xlsx")

# or you could do:
plt.scatter(mangkhut['Pressure (hPa)'],mangkhut['Wind (kt)'])

plt.xlabel('Pressure, nPa')
plt.ylabel("Max Wind speed, kt")
plt.title('Speed versus Pressure')
plt.show()
```



recall: when using NumPy functions to load 2-D array, we used index slicing method, e.g., mangkhut[:, 6], to access the Pressure data. In a Pandas dataFrame, we use the column index (usually a string with physical meaning) to do similar things

Now let’s try Pandas!