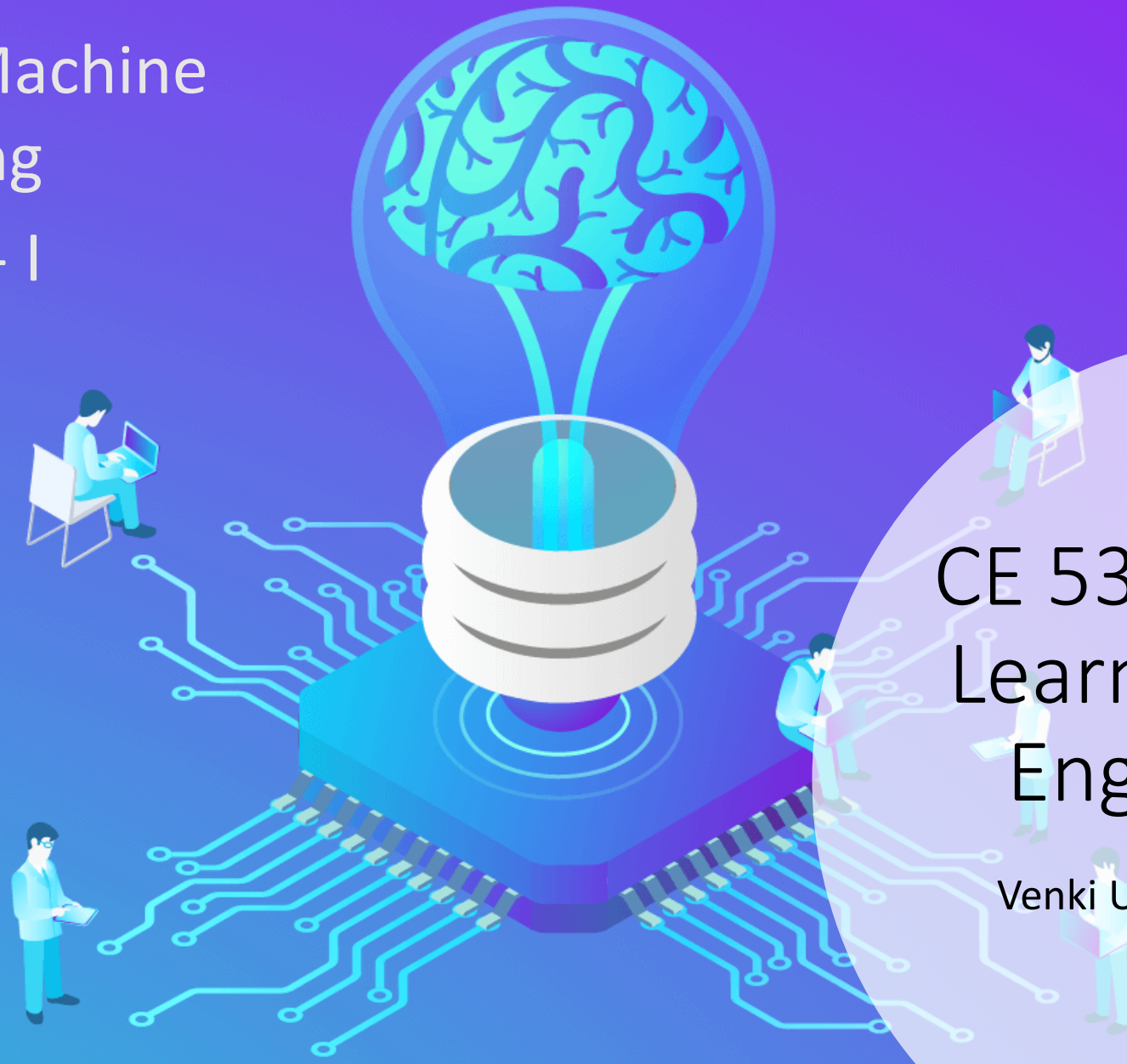
Python for Machine Learning

Basics - I



CE 5331 Machine Learning for Civil Engineers

Venki Uddameri, Ph.D. , P.E.

Recap and Goals

- Installed Python and Anaconda Environments

- Understand basic commands in Python
  - Not exhaustive but aimed at getting you started
    - You need to practice, practice
    - The more you practice the more proficient you will become

  - We shall focus on those functions useful for this class
    - You should see some things you may have encountered in probabilistic methods and numerical methods classes

Python

- Is a powerful, fast, easy to learn high-level programming language
- Open-source, freely usable and distributable
- Developed by Guido von Rossum
  - Was its Benevolent Dictator for Life till 2018
- Python is a full-fledged programming language that can be used for a variety of applications
  - Web and internet development
  - Database applications
  - Scientific and Numeric Computing
  - Network Programming
  - Software and Game Development

Python is becoming an important part of Data Science toolkit

Python versus R for Machine Learning

Both R and Python have their strengths and weaknesses
R offers greater flexibility when it comes to analysis methods
Python offers many programmatic advantages

It is good to be proficient in both

Difference between R and Python (source: www.guru99.com)

Parameter	R	Python
Objective	Data analysis and statistics	Deployment and production
Primary Users	Scholar and R&D	Programmers and developers
Flexibility	Easy to use available library	Easy to construct new models from scratch. I.e., matrix computation and optimization
Learning curve	Difficult at the beginning	Linear and smooth
Popularity of Programming Language. Percentage change	4.23% in 2018	21.69% in 2018
Average Salary	$99.000	$100.000
Integration	Run locally	Well-integrated with app
Task	Easy to get primary results	Good to deploy algorithm
Database size	Handle huge size	Handle huge size
IDE	Rstudio	Spyder, Ipthon Notebook
Important Packages and library	tydiverse, ggplot2, caret, zoo	pandas, scipy, scikit-learn, TensorFlow, caret
Disadvantages	Slow High Learning curve Dependencies between library	Not as many libraries as R
Advantages	• Graphs are made to talk. R makes it beautiful • Large catalog for data analysis • GitHub interface • RMarkdown • Shiny	• Jupyter notebook: Notebooks help to share data with colleagues • Mathematical computation • Deployment • Code Readability • Speed • Function in Python

Python Basics

- Python is a high-level language
  - Interaction using English like commands
- Python is an interpreted language
  - Code is executed sequentially line by line
  - Compiled code can be embedded as functions
    - Many matrix calculations are carried out this way
- Python is a very lean language
  - Has very little in-terms of built in functionality
  - But is augmented by use of libraries (modules)
- Python is object-oriented language
  - Class-Object-Attribute-Value Hierarchy
- Python offers structural programming constructs
  - If-elif-else statement
  - For loops
  - While loops

- Python can be used to develop user-defined functions
  - Can be called in other programs
- Native Python is limited in terms of data structures
  - Strings, lists, numbers, *tuples and sets*
  - Dictionaries
- Packages are developed to enhance data types in R
  - Numpy, pandas
- Python does not come with extensive scientific computation functions
  - Numpy and Scipy enhances this functionality

In this lecture, I will provide a basic introduction to Python using a data-science perspective

Focus on the logic
Identify what steps are necessary
Then worry about the syntax

Python Basics

Mastery in Python means how comfortable are you searching and finding necessary syntax

The lecture here is not intended to give you formal training in Python

Python is too vast to be taught extensively along with machine learning concepts

I will expose you to syntax necessary for performing ML tasks of the course (Good starting point for your explorations)

I urge you to practice offline and become familiar with the syntax

Focus on the algorithm and see what steps are necessary

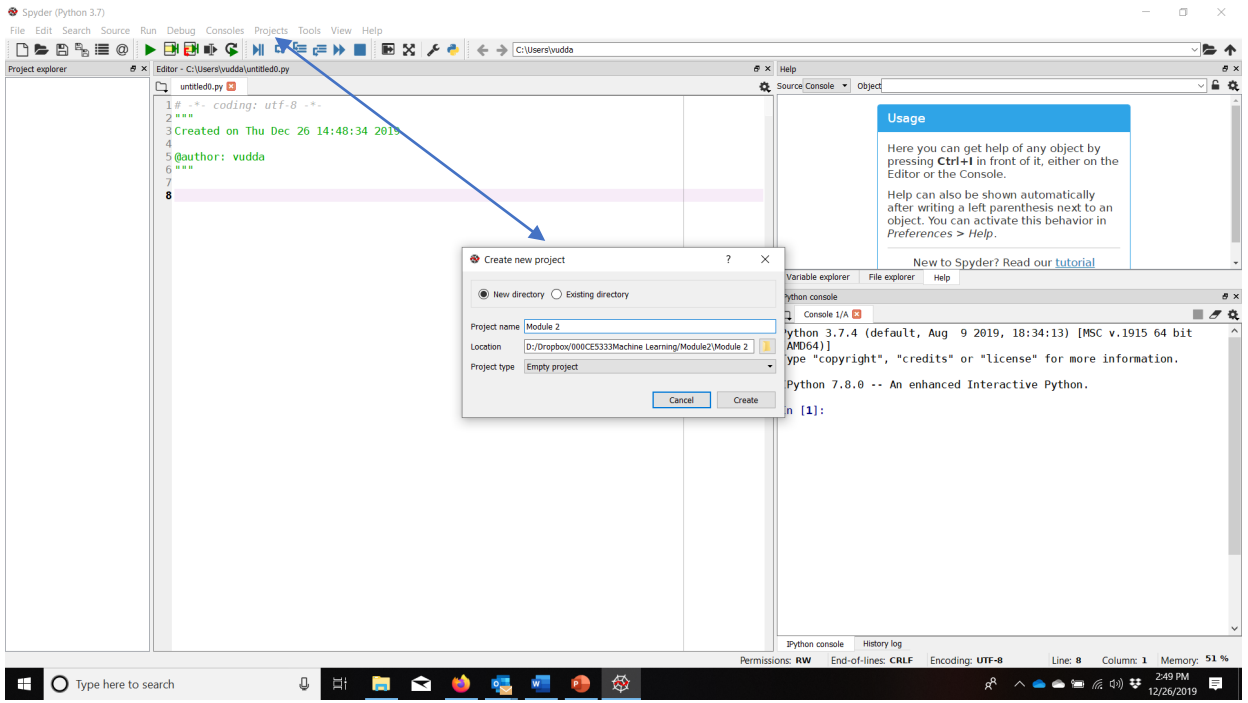
Then figure out what syntax is necessary

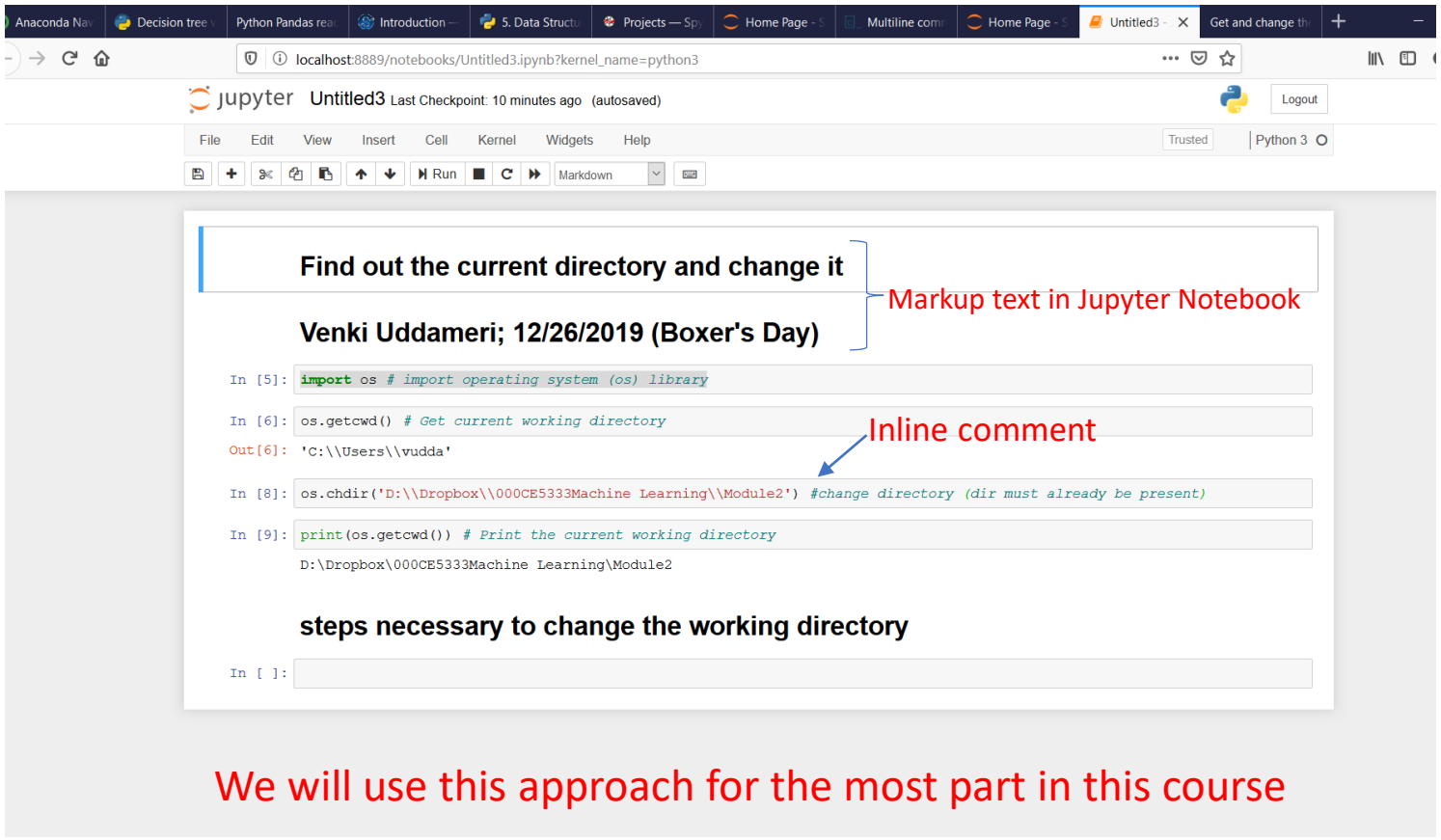
Google is your friend

Setting a Working Directory

Set working Directory – SPYDER Project

- In machine learning projects it is best to set a working directory which has all your code and data files for a given task
- There are a couple of ways to set a working directory
  - Start a new SPYDER project
  - Links the current working directory
    - Puts the working directory in PYPATH (path where all python modules can access)

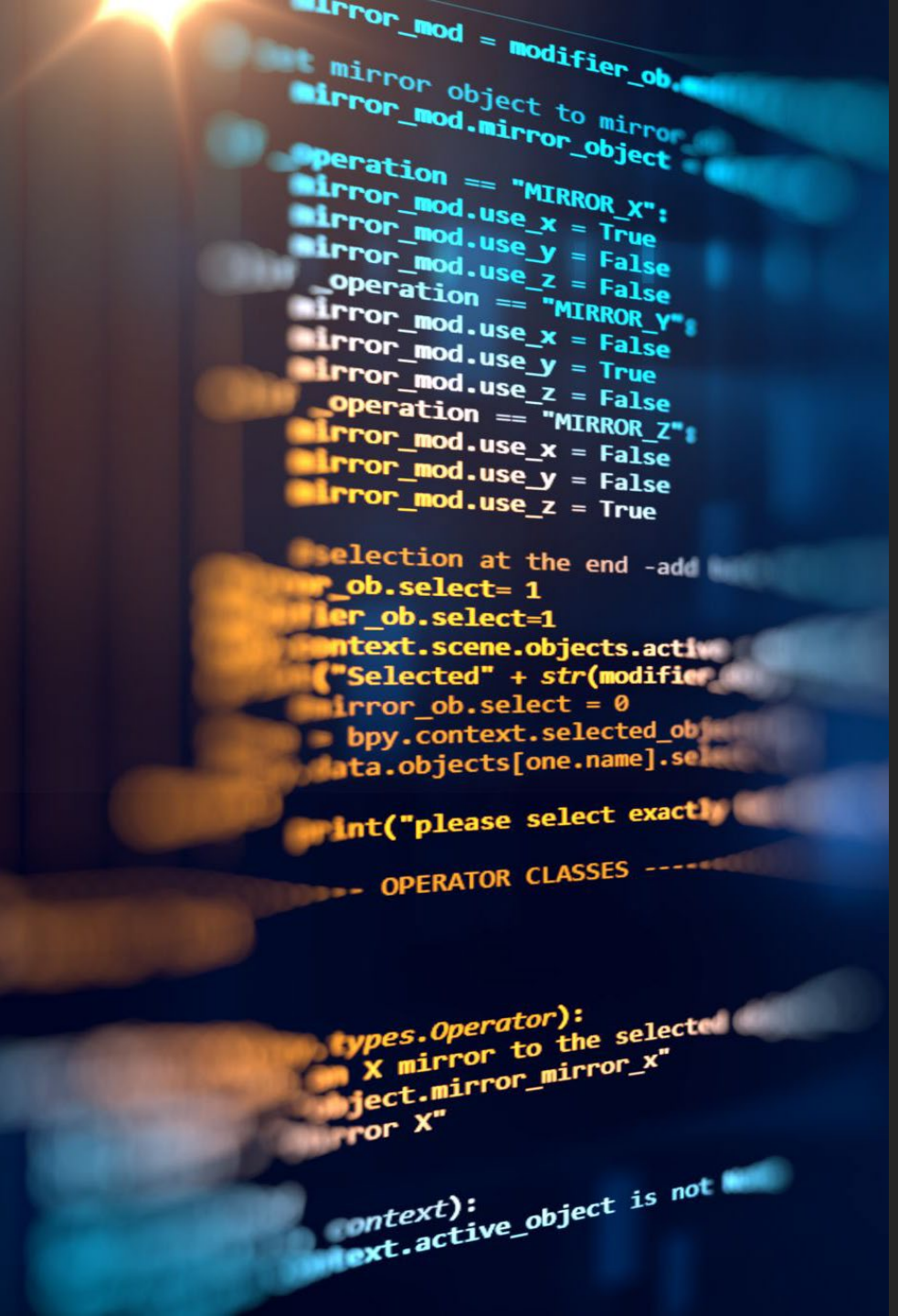




We will use this approach for the most part in this course

Setting aWorking directory Programmatically

- You can use functions in 'os' library to change working directory
  - Use import os
    - To load the os library
  - os.getcwd() to find out the current working directory
  - os.chdir() to change the current working directory
    - This directory must already exist
  - os.mkdir() can be used to make a new directory



Commenting Your Code

Extremely Important but often Ignored

Comment lines in Python

- Comment lines are the heart of any programming/scripting endeavor
  - Makes sense of your code
  - Demonstrates your logic
  - Helps you remember what you did two months ago!!
  - Helps you share code with other developers (when legal to do so)
- The **#** operator is used in python to add comments
  - Python interpreter ignores everything on the line after #
  - Comments can be placed in-line (same line as the code)
  - A # is required for each comment line

Get into the habit of extensively documenting your code your comment lines
A good rule of thumb is to have one comment line per each line of executable code

‘docstrings’ – Special Type of Comments

- You can put a set of lines (multiple lines) within triple quotes “”” after:
  - The definition of a function or a class
  - At the beginning of a module (.py file)
- The text between the triple quotes is called a ‘docstring’
- ‘docstrings’ are special type of comments
  - They are assigned to a special type of object myobj._doc_

Triple quotes are also used to define multiline strings
They are only ‘docstrings’ if placed on the top of a module or after a function definition

It is recommended to place docstrings for each function and method

‘docstrings’ are used to explain **what** the function does not how the function does it. Comment lines are used to describe various (how) steps

‘docstrings’ are used to build documentation of the functions (can include necessary inputs to the functions)

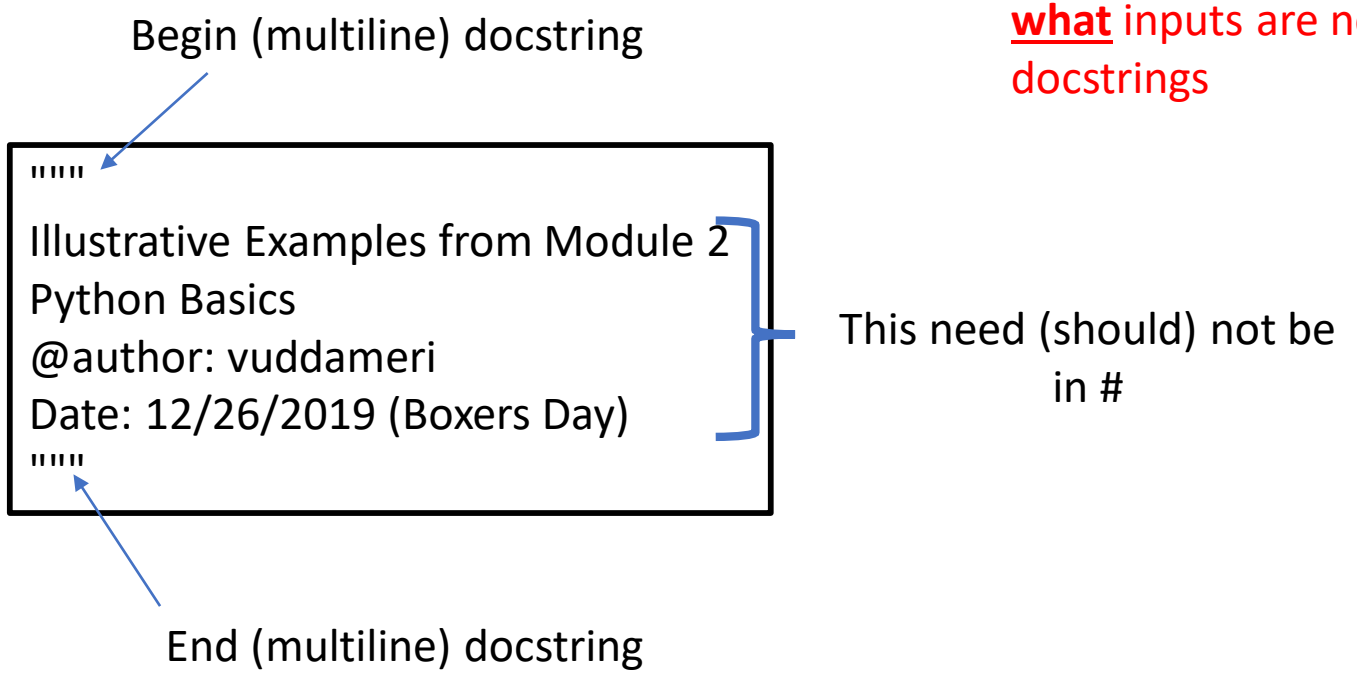
‘docstrings’ are interpreted by the python interpreter while comment lines are not

‘docstrings’ occur only once within a function or a module (after the function or class is defined). Comment lines can occur anywhere

‘docstrings’ Example

Always a good practice to use docstrings to describe the function

Focus on **what** the function does and **what** inputs are necessary when writing docstrings

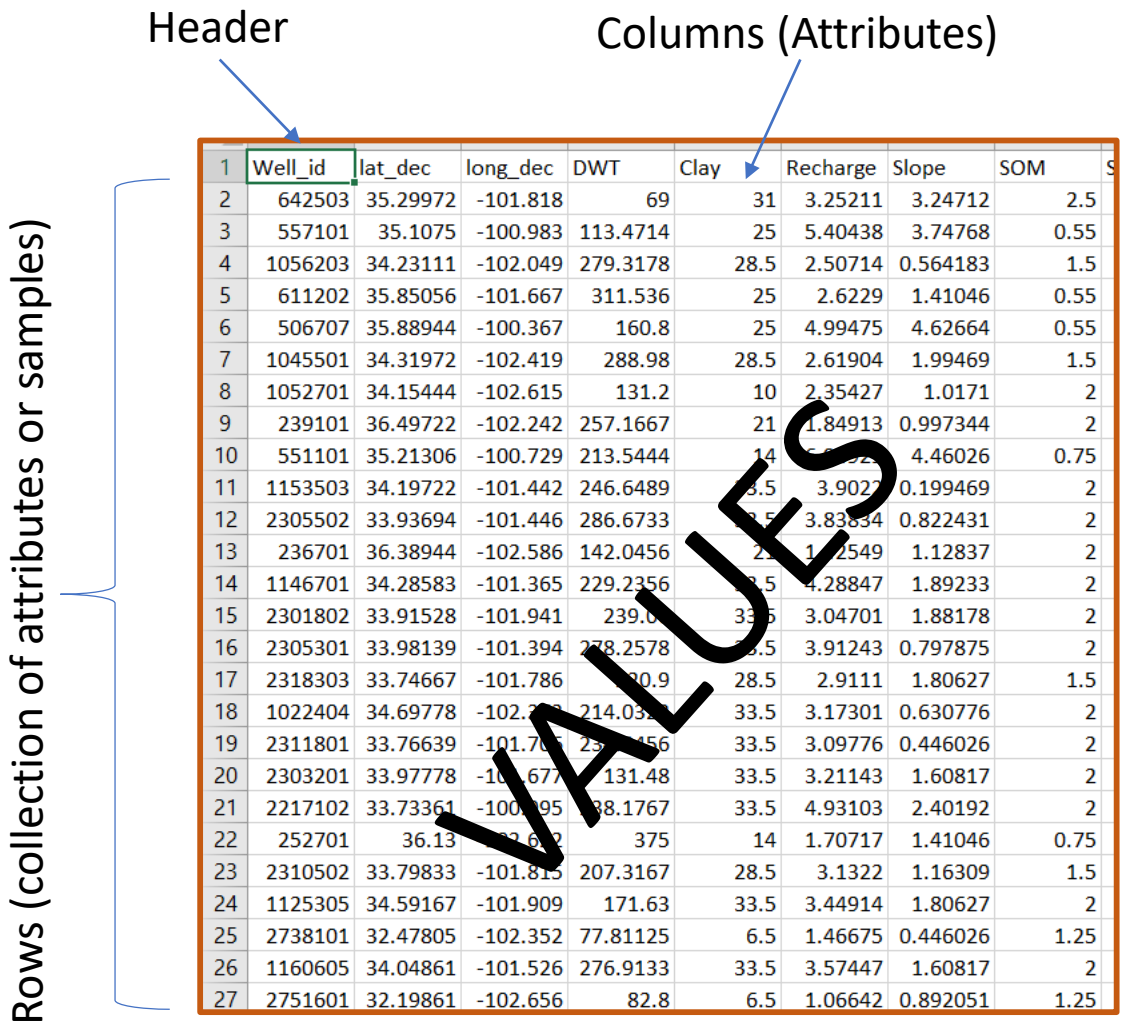


Remember a ‘docstring’ is a multiline string but is placed at the top of the module or after a function definition

Reading Data into Python

Read a ‘csv’ file

- In Machine Learning we typically deal with tabular data
  - Columns → Attributes or features
  - Rows → Different set of measurements (exemplars)
  - Cells have values
- A comma separated value or csv is the best way to store such tabular data
  - Data is stored as strings (ASCII format)
  - Can be read using a text-editor or a spreadsheet
- Reading and writing ‘.csv’ files is an important task in ML
- Native Python does not provide .csv support
  - However libraries are available to fill in this gap!!

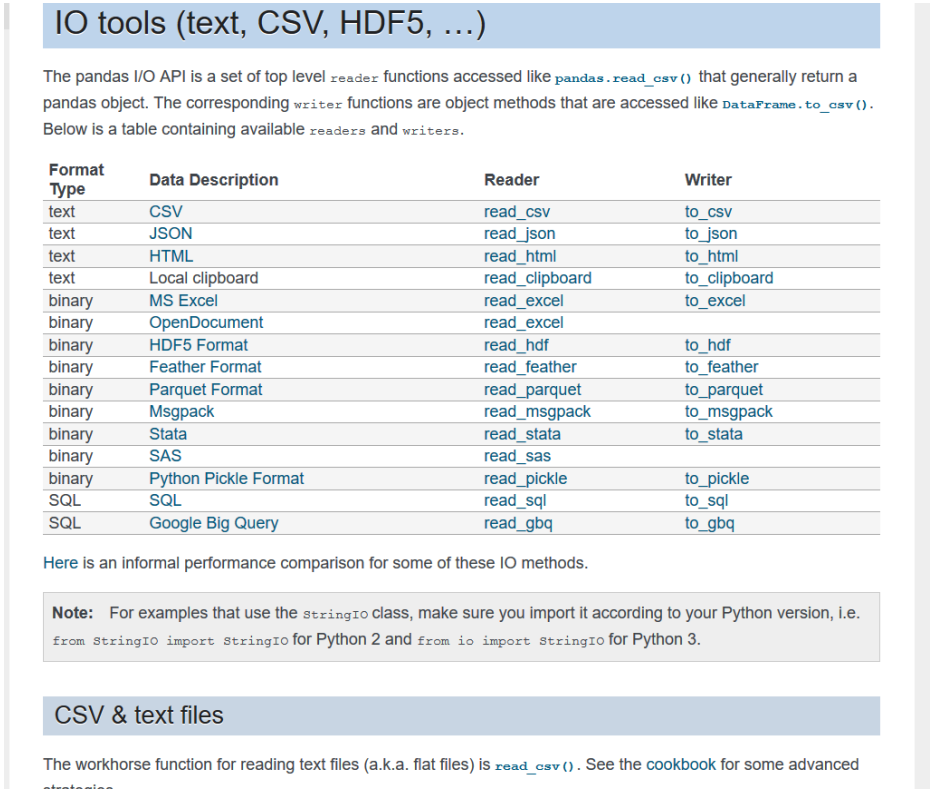


	Well_id	lat_dec	long_dec	DWT	Clay	Recharge	Slope	SOM
2	642503	35.29972	-101.818	69	31	3.25211	3.24712	2.5
3	557101	35.1075	-100.983	113.4714	25	5.40438	3.74768	0.55
4	1056203	34.23111	-102.049	279.3178	28.5	2.50714	0.564183	1.5
5	611202	35.85056	-101.667	311.536	25	2.6229	1.41046	0.55
6	506707	35.88944	-100.367	160.8	25	4.99475	4.62664	0.55
7	1045501	34.31972	-102.419	288.98	28.5	2.61904	1.99469	1.5
8	1052701	34.15444	-102.615	131.2	10	2.35427	1.0171	2
9	239101	36.49722	-102.242	257.1667	21	1.84913	0.997344	2
10	551101	35.21306	-100.729	213.5444	14	[illegible]	4.46026	0.75
11	1153503	34.19722	-101.442	246.6489	[illegible]	3.902[illegible]	0.199469	2
12	2305502	33.93694	-101.446	286.6733	[illegible]	3.83[illegible]	0.822431	2
13	236701	36.38944	-102.586	142.0456	[illegible]	[illegible]2549	1.12837	2
14	1146701	34.28583	-101.365	229.2356	[illegible]	4.28847	1.89233	2
15	2301802	33.91528	-101.941	239.[illegible]	33[illegible]	3.04701	1.88178	2
16	2305301	33.98139	-101.394	278.2578	[illegible]	3.91243	0.797875	2
17	2318303	33.74667	-101.786	[illegible]0.9	28.5	2.9111	1.80627	1.5
18	1022404	34.69778	-102.[illegible]	214.03[illegible]	33.5	3.17301	0.630776	2
19	2311801	33.76639	-101.7[illegible]	23[illegible]56	33.5	3.09776	0.446026	2
20	2303201	33.97778	-1[illegible].677	131.48	33.5	3.21143	1.60817	2
21	2217102	33.73361	-100.[illegible]95	[illegible]8.1767	33.5	4.93103	2.40192	2
22	252701	36.13	[illegible]2.6[illegible]	375	14	1.70717	1.41046	0.75
23	2310502	33.79833	-101.8[illegible]	207.3167	28.5	3.1322	1.16309	1.5
24	1125305	34.59167	-101.909	171.63	33.5	3.44914	1.80627	2
25	2738101	32.47805	-102.352	77.81125	6.5	1.46675	0.446026	1.25
26	1160605	34.04861	-101.526	276.9133	33.5	3.57447	1.60817	2
27	2751601	32.19861	-102.656	82.8	6.5	1.06642	0.892051	1.25

We shall use ‘pandas’ library to read and write ‘csv’ files

Read a .csv file – Use Pandas

- Pandas is a library of python that is open-source and aimed at creating high performance data structures and data analysis tools for python programming language
  - https://pandas.pydata.org/
- Pandas comes pre-installed with anaconda
  - Can also be installed from the shell as *conda install pandas (if necessary)*
- Pandas – stands for ‘panel data’
  - A term in econometrics to mean multidimensional data



IO tools (text, CSV, HDF5, …)

The pandas I/O API is a set of top level `reader` functions accessed like `pandas.read_csv()` that generally return a pandas object. The corresponding `writer` functions are object methods that are accessed like `DataFrame.to_csv()`. Below is a table containing available `readers` and `writers`.

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	OpenDocument	read_excel	
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

Here is an informal performance comparison for some of these IO methods.

Note: For examples that use the `StringIO` class, make sure you import it according to your Python version, i.e. `from StringIO import StringIO` for Python 2 and `from io import StringIO` for Python 3.

CSV & text files

The workhorse function for reading text files (a.k.a. flat files) is `read_csv()`. See the cookbook for some advanced

Source: www.pandas.org

Pandas I/O functions

- Pandas provides a variety of functions to perform Input-Output operations
  - While CSV files are the workhorse, pandas can also read a variety of other file formats

Pandas – Reading CSV files

- The ‘read_csv’ function stores the contents of a file into a pandas dataframe
- Pandas has two main data structures of interest
  - Pandas series
    - One dimensional data
  - Pandas dataframe
    - Multidimensional data
- Once the dataframe is read into python we want to make certain manipulations to the data
  - Extract certain columns
    - Subset of attributes for further analysis
  - Extract certain rows
    - Separate training and testing datasets

In addition to reading csv files, read_csv can also read other delimited files

Data types for each attributes can be specified within read_csv to read in data correctly

You can skip a certain number of lines when reading data

Code Snippet

```
import pandas as pd  # Pandas as a pd object
a = pd.read_csv('Ogallaladata.csv')  #read the csv file
print(a.head(5)) # look at the first 5 lines of data
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

Pandas - Series and Dataframe

Pandas has two basic data structures

Data structures

Dimensions	Name	Description
1	Series	1D labeled homogeneously-typed array
2	DataFrame	General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

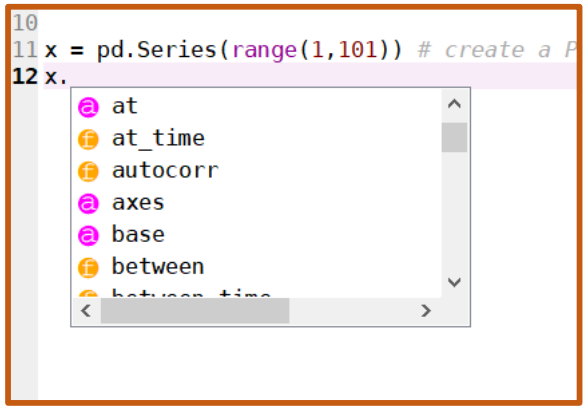
Pandas – Series

- Pandas series is the basic data structure to store 1-D data
  - Similar to a vector in R
- Python is more object-oriented than R
  - Each object in python usually has associated functions (methods) and attributes
  - The ‘dot’ notation is used to access these functions and attributes

```
import pandas as pd   # Import pandas as a pd object
# create a Pandas Series
x = pd.Series(range(1,101))
x.size
```

Series is a method associated with object pd

Size is a value associated with object x



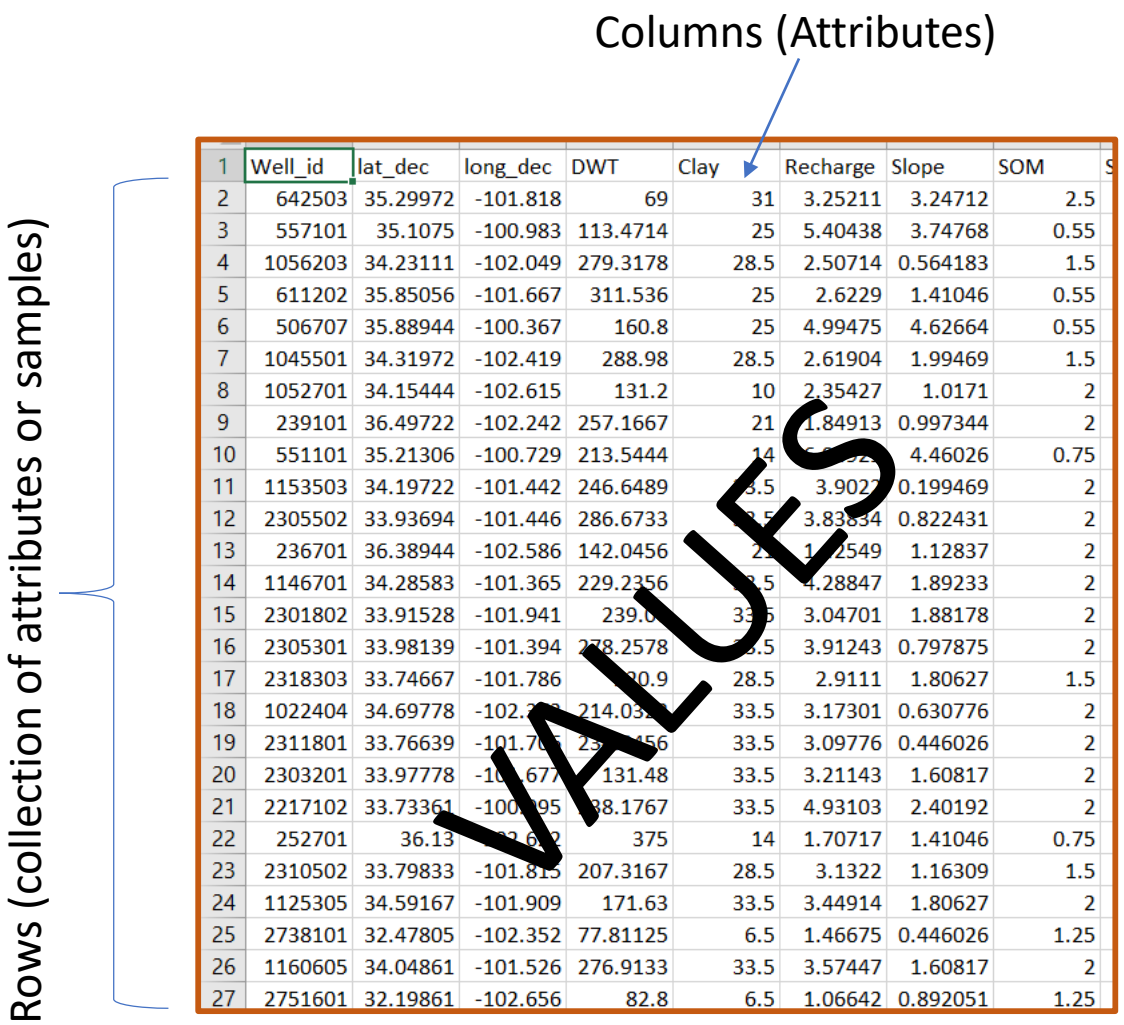
Attributes & Functions associated with object x

Python is case-sensitive so keep track of capitializaton

SPYDER provides a list of methods and values associated with each object and can be useful for initial explorations

Pandas - Dataframe

- A dataframe consists of 3 components
  - Rows, Columns and values
- Rows and Columns are indexed
  - Indices start at zero in python by default
  - Can be changed in Pandas
- Each column holds a specific attribute (feature)
- A dataframe can comprise of columns having different datatypes
  - Strings, integers, real



	Well_id	lat_dec	long_dec	DWT	Clay	Recharge	Slope	SOM
2	642503	35.29972	-101.818	69	31	3.25211	3.24712	2.5
3	557101	35.1075	-100.983	113.4714	25	5.40438	3.74768	0.55
4	1056203	34.23111	-102.049	279.3178	28.5	2.50714	0.564183	1.5
5	611202	35.85056	-101.667	311.536	25	2.6229	1.41046	0.55
6	506707	35.88944	-100.367	160.8	25	4.99475	4.62664	0.55
7	1045501	34.31972	-102.419	288.98	28.5	2.61904	1.99469	1.5
8	1052701	34.15444	-102.615	131.2	10	2.35427	1.0171	2
9	239101	36.49722	-102.242	257.1667	21	1.84913	0.997344	2
10	551101	35.21306	-100.729	213.5444	14	[illegible]	4.46026	0.75
11	1153503	34.19722	-101.442	246.6489	[illegible]	3.9022	0.199469	2
12	2305502	33.93694	-101.446	286.6733	[illegible]	3.83634	0.822431	2
13	236701	36.38944	-102.586	142.0456	2[illegible]	[illegible]2549	1.12837	2
14	1146701	34.28583	-101.365	229.2356	[illegible]	4.28847	1.89233	2
15	2301802	33.91528	-101.941	239.[illegible]	33[illegible]	3.04701	1.88178	2
16	2305301	33.98139	-101.394	[illegible]78.2578	[illegible].5	3.91243	0.797875	2
17	2318303	33.74667	-101.786	[illegible]0.9	28.5	2.9111	1.80627	1.5
18	1022404	34.69778	-102.[illegible]	214.03[illegible]	33.5	3.17301	0.630776	2
19	2311801	33.76639	-101.7[illegible]	23[illegible]56	33.5	3.09776	0.446026	2
20	2303201	33.97778	-10[illegible].677	131.48	33.5	3.21143	1.60817	2
21	2217102	33.73361	-100.[illegible]95	[illegible]8.1767	33.5	4.93103	2.40192	2
22	252701	36.13	-[illegible].672	375	14	1.70717	1.41046	0.75
23	2310502	33.79833	-101.815	207.3167	28.5	3.1322	1.16309	1.5
24	1125305	34.59167	-101.909	171.63	33.5	3.44914	1.80627	2
25	2738101	32.47805	-102.352	77.81125	6.5	1.46675	0.446026	1.25
26	1160605	34.04861	-101.526	276.9133	33.5	3.57447	1.60817	2
27	2751601	32.19861	-102.656	82.8	6.5	1.06642	0.892051	1.25

Dataframes are typically read in from a spreadsheet or as a comma separated value (csv) file

Subsetting and Extracting Columns (Attributes)

- Pandas also provides many functions and methods to subset and extract data from a dataframe

Python (Pandas) uses zero-based indexing

Row and column indices in a dataframe start at zero (0) and move forward

1-D array in Pandas is called as ‘series’
2+D array in Pandas is called a ‘dataframe’

Function	Utility
a = read_csv(‘filename.csv’)	Reads the csv file and creates a pandas dataframe
list(a.columns)	Lists the names of attributes in dataframe ‘a’
a.xxx	Extracts column with name ‘xxx’ from dataframe as a pandas series
a[[‘z’,’y’]]	Extracts 2 columns z and y within ‘a’ into a separate dataframe (Note double brackets)
a.columns = [‘x’,’y’,’z’]	Rename column names in a dataframe
a.rename(columns={‘x': ‘xold'}, inplace=True)	Rename only a selected column in a dataframe

Extracting values from a list or series

- Consider a list z = [1,2,3,4]
  - Python uses zero based indexing and a (n-1) stop rule
- z[0] = 1, z[1] = 2, etc.
- z[0:2] = [1,2] (pulls out 0, 1 indexed elements)
  - (n-1) stop rule means for n = 2 the last element pulled out has an index of 1; keep in mind the indexing starts at 0
- z[0:5] will not throw an error but z[5] will – Why?

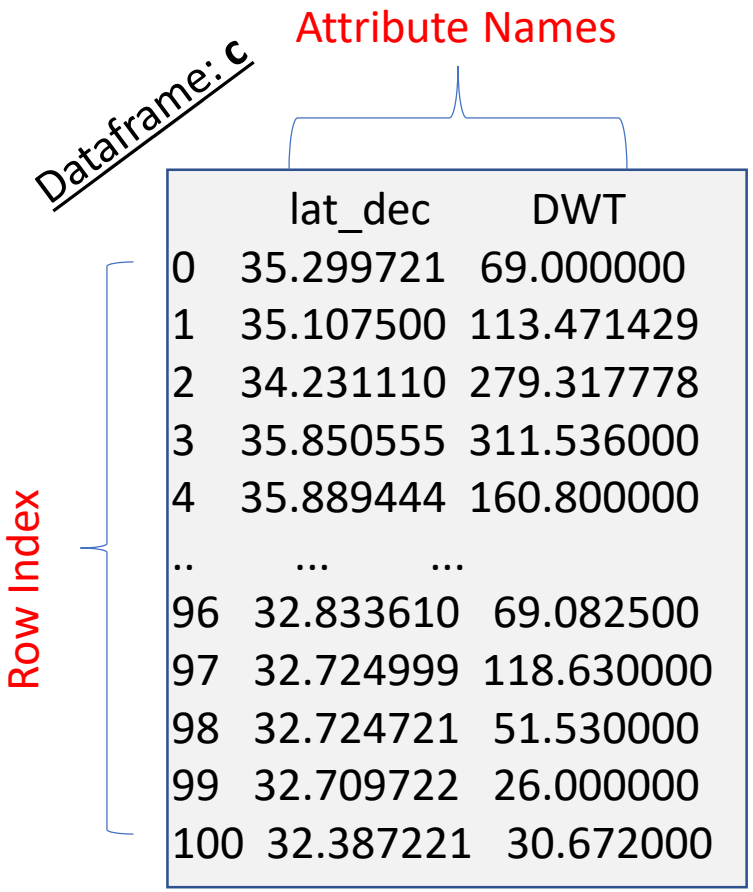
Extracting Rows from a Dataframe

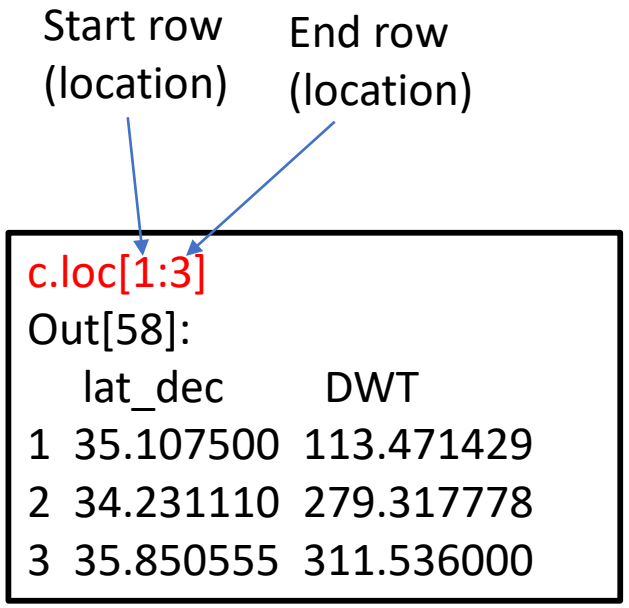
- A dataframe can have both row names and column names
  - But it is usually more common to only have column names
  - Rows in a dataframe are given an index by pandas
    - Indexing by default begins at zero
- Pandas provides several ways to extract rows
  - The two most important ways are ‘loc’ and ‘iloc’ methods

As their name suggests ‘loc’ works with the location

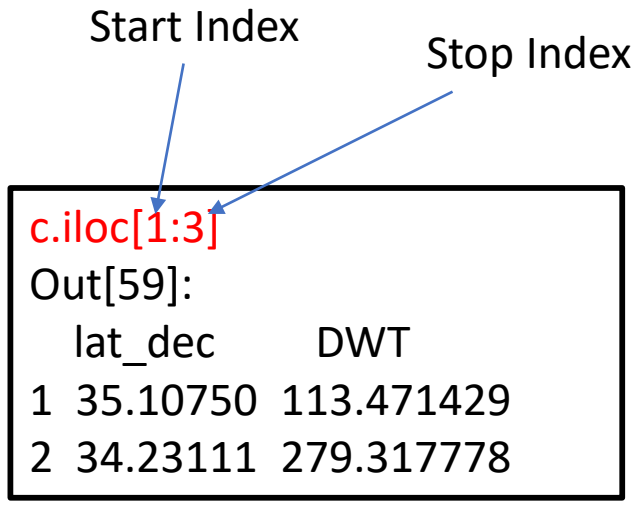
‘iloc’ works with the index of the location

Extracting Rows from a Dataframe





Locations 1 – 3 are extracted by loc



Locations 1 and 2 are extracted by iloc

Remember indexing in python starts at zero by default; Row corresponding to stop index in iloc is not retrieved

Extracting Rows and Columns

- You want to extract specific rows and certain columns from a dataset
  - For example split your training dataset
- Both ‘loc’ and ‘iloc’ can be used for this purpose but it is easier to do it with ‘loc’ so I will only show this here

Dataframe ‘d’

	DWT	Clay	NO3Av	Train
0	69.000000	31.0	6.390	Training
1	113.471429	25.0	5.600	Training
2	279.317778	28.5	8.920	Training
3	311.536000	25.0	7.155	Training
4	160.800000	25.0	4.740	Training
..	...	...	...	...
96	69.082500	7.5	42.110	Testing
97	118.630000	7.5	21.520	Testing
98	51.530000	7.5	21.300	Testing
99	26.000000	7.5	58.210	Testing
100	30.672000	14.0	35.200	Testing

```
d.loc[d['Train']=='Training',['DWT','NO3Av']]
```

Out[69]:

	DWT	NO3Av
0	69.000000	6.390
1	113.471429	5.600
2	279.317778	8.920
3	311.536000	7.155
4	160.800000	4.740
..	...	...
70	54.592727	41.500
71	48.500000	62.200
72	19.890000	182.765
73	53.556000	116.205
74	169.484444	26.680

Extracted DWT and NO3AV
for Training (75 rows of data)

Accessing Scalar Values

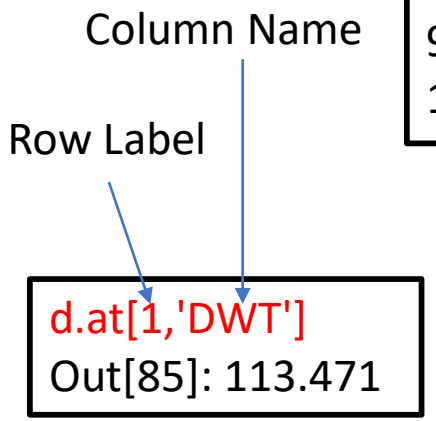
- For fast access of scalar values (individual cell) in a dataframe or a series – You can use pandas ‘at’ or ‘iat’ functions

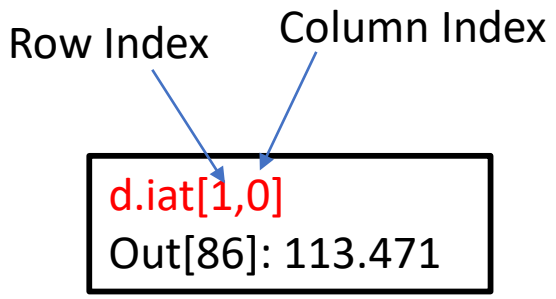
Dataframe ‘d’

	DWT	Clay	NO3Av	Train (3)
0	69.000000	31.0	6.390	Training
1	113.471429	25.0	5.600	Training
2	279.317778	28.5	8.920	Training
3	311.536000	25.0	7.155	Training
4	160.800000	25.0	4.740	Training
..	...	...	...	...
96	69.082500	7.5	42.110	Testing
97	118.630000	7.5	21.520	Testing
98	51.530000	7.5	21.300	Testing
99	26.000000	7.5	58.210	Testing
100	30.672000	14.0	35.200	Testing

df.at[row index, column name]

df.iat[row index, column index]





Remember both ‘at’ and ‘iat’ can only be used to obtain a single value (scalar) from a series or a dataframe

Also remember that python indexing starts from zero and not unity

Other Functions

- Pandas offers many other functions that are useful to understand data
  - Data can be reshaped in many formats
  - Summary statistics of the dataframe can be obtained using df.describe() method

Illustrative Example

- Read the ‘ogallaladata.csv’ python using the pandas library
- Subset the training dataset
- Extract well depth and average nitrogen concentration for training data

You Should Know

- How to set up a working directory in Python
- How to add comment lines
- What is a ‘docstrings’ why is it used
- How to read a csv file into Python using ‘pandas’ library
- How to extract specific columns
- How to extract specific rows
- How to extract rows and columns
- Obtain basic summary measures of the dataframe