Pandas1-Objects

Course content

Part 1: Basics of programming (W34-40)

- Programming languages
- Programming environment
- Python syntax
- Variables
- Strings and numbers
- Lists
- Conditional statements
- Loop statements
- Functions
- Dictionary, tuple, and set

Part 2: Data extraction and visualization (W41-47)

- Pandas Series and DataFrame
- Read csv files
- Basic statistics
- Data manipulation
- Missing data handling
- Data aggregation
- Data visualization – Pandas
- Data visualization - Matplotlib
- Data visualization - Seaborn
- Time series data

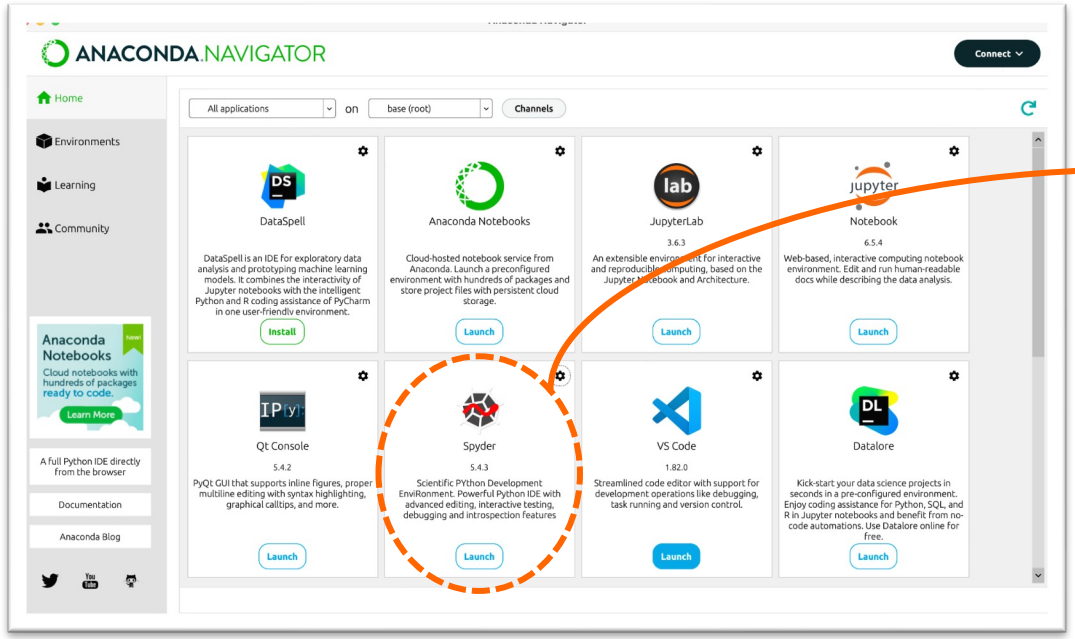
Outline

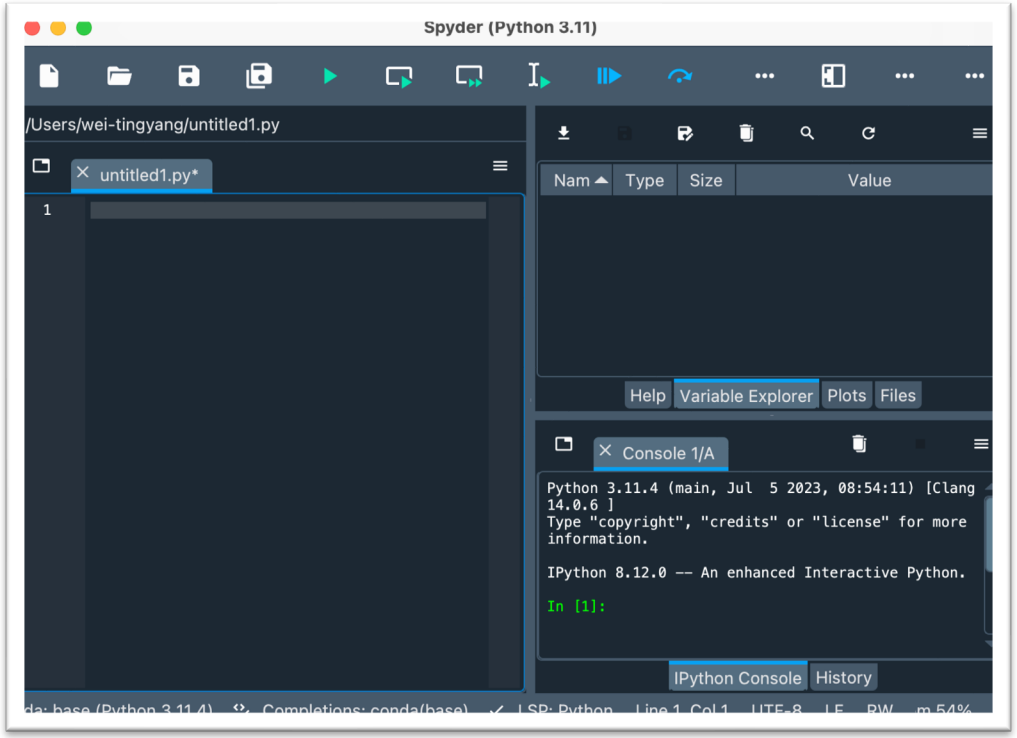
- **Python file**
- **Modules and packages**
- **Pandas**
  - Install package
  - Pandas data structure: Series, DataFrame
  - Read data from file to DataFrame
  - Sorting

Different ways to write and run Python code

Coding environments	Supported file types
Jupyter notebook	• ipynb file
Integrated Development Environments (IDEs) (e.g., Spyder, PyCharm)	• py file
Code Editors (e.g., VScode, Atom)	• py file • ipynb file (if Jupyter Notebook extension has been installed and enabled.
Text Editors (e.g., Notepad++, Sublime Text)	• py file

Write code using an IDE

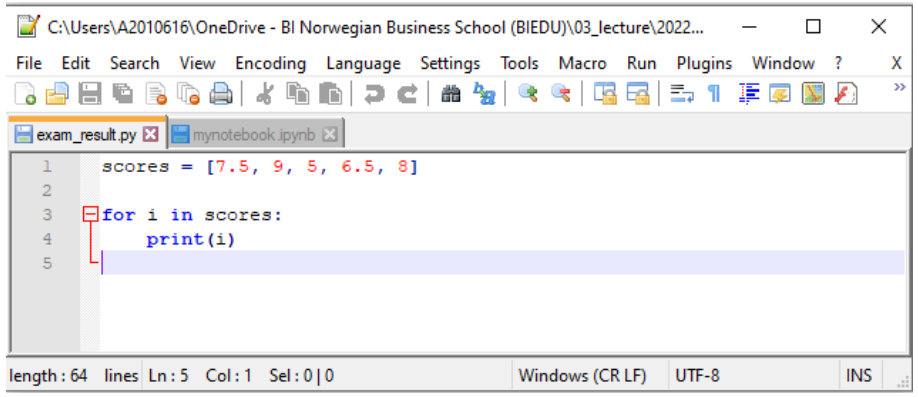




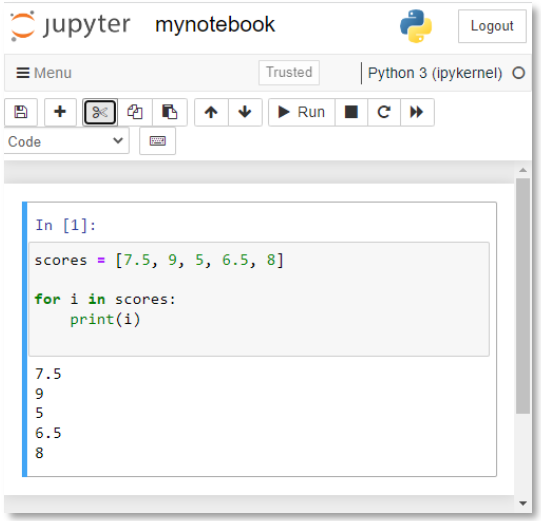
Python file

- .py is a regular python file. It's plain text and contains just your code.
- .ipynb is an interactive python notebook and it can contain code, text, execution results and visualization.

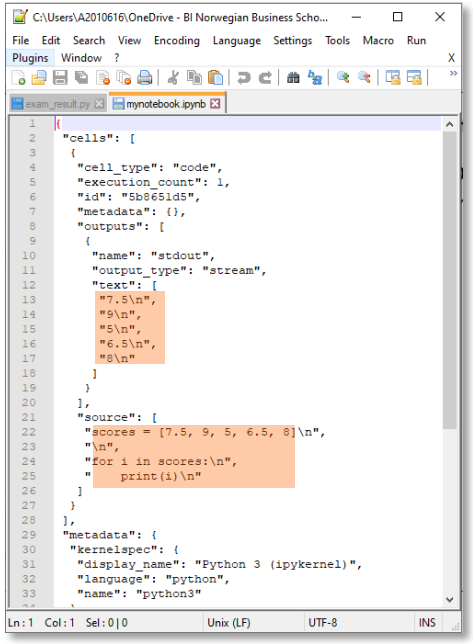
Open .py file in Notepad



.ipynb file



Open .ipynb file in Notepad

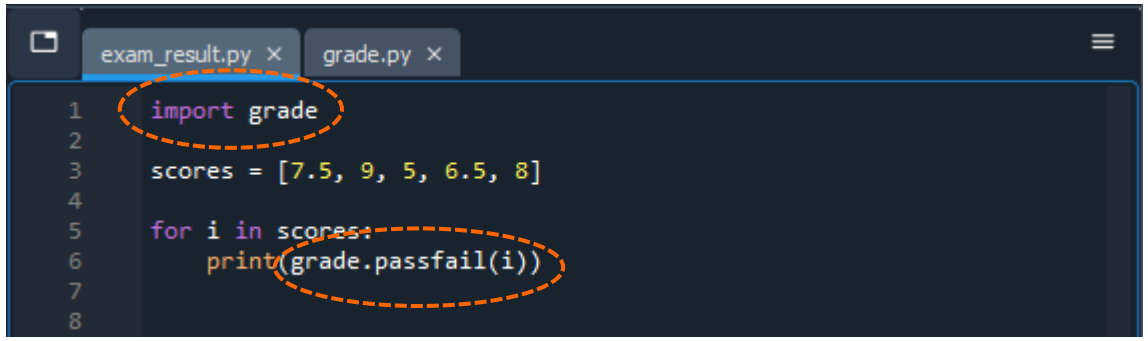


- For mac users, try to use the app TextEdit to open the file.
- Jupyter Notebooks use JSON schema as its underlying structure, enabling an interactive computing environment and allowing display in a web browser. https://nbformat.readthedocs.io/en/latest/format_description.html

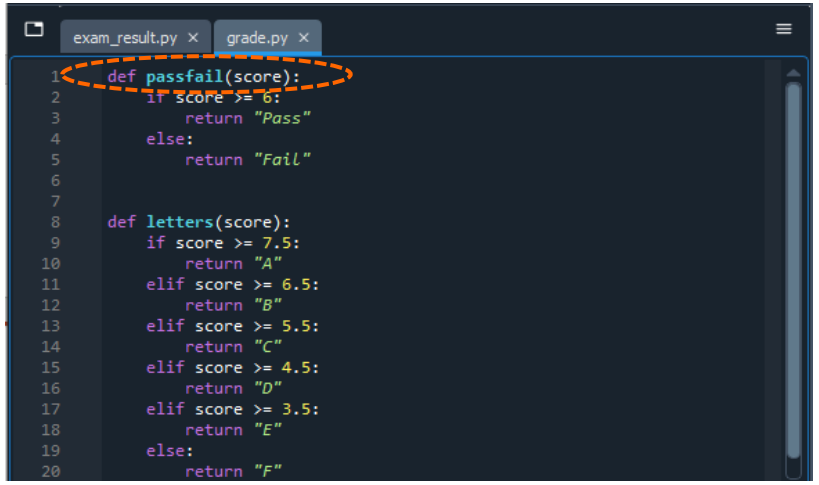
Modules

- Script:
  - A script is a python file (.py) designed to be run directly.
- Module:
  - A module is a python file (.py) that contains reusable code, such as functions, classes, and variables.
  - A module is designed to be imported and used in other modules or scripts.
  - The filename is the module name.

Script: exam_result.py



Module: grade.py



- https://docs.python.org/3/tutorial/modules.html#

Optional exercise: Create your own module

- Step1: Create a module named **student**. (Use **student.py** as file name).
- Step2: Write some functions and save the file.
- Step3: Under the same folder, create a jupyter notebook.
- Step4: Import the module **“student”.**
- Step5: Call the function **greeting()** from the module.

student.py

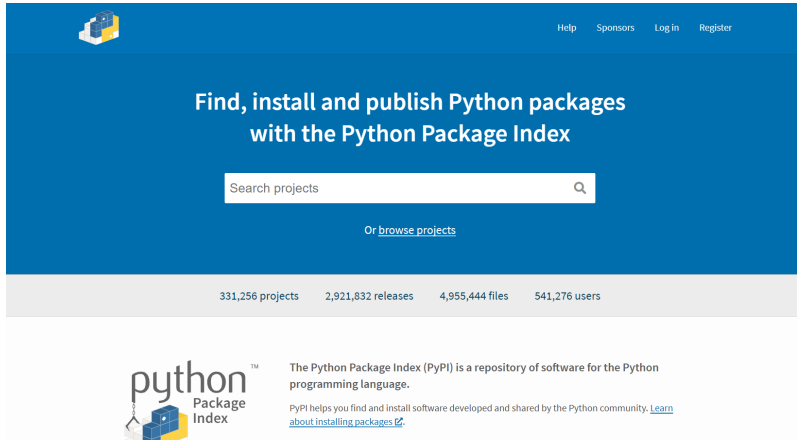
```python
def greeting():
    print("Welcome to BI!")

def profile(student_id, name):
    print("New Student Profile:")
    print(f"ID: {student_id}")
    print(f"Name: {name}")

```

```
In [1]: import student

In [2]: student.greeting()
        Welcome to BI!

In [3]: student.profile("S01234", "Anna")
        New Student Profile:
        ID: S01234
        Name: Anna
```

Packages

- Packages are a collection of modules.
- The **Py**thon **P**ackage **I**ndex (PyPI) is a repository of pacakges for the Python programming language.
- **pip** is the package installer for Python. You can use it to install packages from PyPI.

https://pypi.org/



https://docs.python.org/3/tutorial/modules.html#packages

Packages – packages for data analysis

- **NumPy** (Numerical Python)
  - Large multidimensional array operations
- **SciPy** (Scientific Python)
  - Many efficient numerical routines such as routines for numerical integration and optimization
- **Pandas**
  - Data manipulation and data visualization
- **Matplotlib**
  - Data exploration and data visualization
- **Seaborn**
  - High-level data visualization library based on Matplotlib
- **Scikit-learn**
  - Machine learning and statistical modeling

Pandas

Pandas

- Pandas
  - An open-source package providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
- Installation
  - Installing pandas from a jupyter notebook

```
!pip install pandas
```

  - Installing pandas from Anaconda Navigator
    - https://docs.anaconda.com/free/navigator/tutorials/manage-packages/#installing-a-package
- Import package
  - The "`as pd`" part means that we can write the shorthand "pd" instead of "pandas" when we use it later. In principle, you could use other names than "pd", but this is the common alias.

```
import pandas as pd
```


https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html#cookbook

Data structures

- **Python list**

```
mylist = [1,2,3,4,5]
print(mylist)
```

```
[1, 2, 3, 4, 5]
```

- **NumPy array**:
  - Numpy array can directly handle a mathematical operations.

```
import numpy as np
myarray = np.array([1,2,3,4,5])
print(myarray)
```

```
[1 2 3 4 5]
```

```
np.square(myarray)
```

```
array([ 1,  4,  9, 16, 25])
```

```
np.log(myarray)
```

```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791])
```

Pandas data structures

- **Pandas data structures (data type)**
  - **Series class:** A one-dimensional array-like structure.
    - A series contains an array of data and an associated array of index.

0	216
1	143
2	98
3	455
4	108

  - **DataFrame class:** A tabular, spreadsheet-like structure.
    - A DataFrame contains an ordered collection of columns, each of which can be a different data type (numeric, string, boolean, etc.).
    - A DataFrame has both a row and column index.

	High	Width	weight	group
0	20	20	0.1	A
1	45	30	0.8	C
2	54	43	1.5	C
3	25	15	2.3	B
4	18	34	0.2	A

DataFrame

	High
0	20
1	45
2	54
3	25
4	18

Series

	Width
0	20
1	30
2	43
3	15
4	34

Series

	weight
0	0.1
1	0.8
2	1.5
3	2.3
4	0.2

Series

	group
0	A
1	C
2	C
3	B
4	A

Series

Series - creation

- Create a series from a list

```
Series1 = pd.Series([4, 7, -5, 3])
Series1
```

```
0    4
1    7
2   -5
3    3
dtype: int64
```

index

- Get values and index

```
Series1.values
```

```
array([ 4,  7, -5,  3], dtype=int64)
```

The attribute “values” are simply a NumPy array.

```
Series1.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

RangeIndex is the default index type used by Series and DataFrame when no explicit index is provided by the user.

BI

- A class is a blueprint to create objects, which can have attributes and methods.
  - **Attributes** are variables that store data or information about an object.
  - **Methods** are functions defined for a specific object (data type).
- https://pandas.pydata.org/docs/reference/api/pandas.Series.html

Series - index

- Assign index

```
Series2 = pd.Series([4, 7, -5, 3], index = ["a","b","c","d"])
Series2
```

```
a    4
b    7
c   -5
d    3
dtype: int64
```

- Get index

```
Series2.index
```

```
Index(['a', 'b', 'c', 'd'], dtype='object')
```


https://pandas.pydata.org/docs/reference/api/pandas.Index.html

DataFrame - creation

- DataFrame: A tabular, spreadsheet-like structure.
- Create a DataFrame from a dictionary of equal-length lists.

```
data = {"state": ["Ohio","Ohio","Ohio","Nevada","Nevada","Nevada"],
        "year":[2000,2001,2002,2001,2002,2003],
        "pop":[1.5,1.7,3.6,2.4,2.9,3.2]}
```

```
df1 = pd.DataFrame(data)
df1
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Nevada	2003	3.2

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

DataFrame – index and dimension

- Assign index

```
df2 = pd.DataFrame(data, index = ["a","b","c","d","e","f"])
df2
```

	state	year	pop
a	Ohio	2000	1.5
b	Ohio	2001	1.7
c	Ohio	2002	3.6
d	Nevada	2001	2.4
e	Nevada	2002	2.9
f	Nevada	2003	3.2

- The "shape" attribute returns a tuple representing the dimensions of the DataFrame.

```
df2.shape
```

number of rows `(6, 3)` number of columns

DataFrame – a collection of series

- A single column in a DataFrame is a Pandas Series.

	state	year	pop
a	Ohio	2000	1.5
b	Ohio	2001	1.7
c	Ohio	2002	3.6
d	Nevada	2001	2.4
e	Nevada	2002	2.9
f	Nevada	2003	3.2

```
year_series = df2["year"]

print(year_series)
```

```
a    2000
b    2001
c    2002
d    2001
e    2002
f    2003
Name: year, dtype: int64
```

```
print(type(year_series))
```

```
<class 'pandas.core.series.Series'>
```

Exercise

Exercise.A

(A.1) Create a dictionary containing three key-value pairs. Use the keys "exam1", "exam2" and "exam3" and specify the following list as the corresponding values.

```
score_list1 = [70, 85, 90, 50, 75]
score_list2 = [80, 65, 85, 60, 80]
score_list3 = [85, 70, 70, 75, 80]
```

(A.2) Create a dataframe from the dictionary you obtained in (A.1) with the following list as the index. Display the dataframe.

```
ID_list = ['S01', 'S02', 'S03', 'S04', 'S05']
```

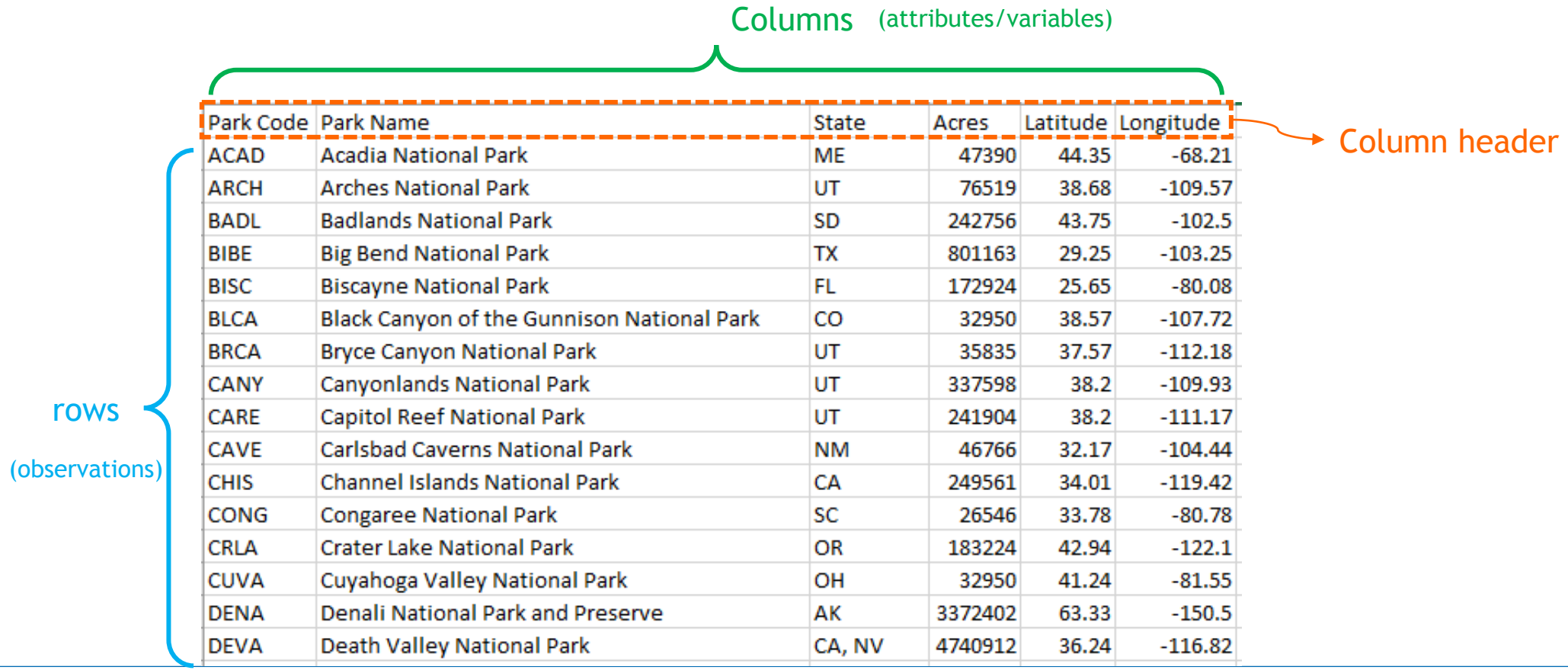
	exam1	exam2	exam3
S01	70	80	85
S02	85	65	70
S03	90	85	70
S04	50	60	75
S05	75	80	80

(A.3) Using the dataframe obtained in (A.2), print the number of rows and columns.

Read csv file into Pandas DataFrame

Dataset

- A dataset is a collection of data with a defined structure.



Park Code	Park Name	State	Acres	Latitude	Longitude
ACAD	Acadia National Park	ME	47390	44.35	-68.21
ARCH	Arches National Park	UT	76519	38.68	-109.57
BADL	Badlands National Park	SD	242756	43.75	-102.5
BIBE	Big Bend National Park	TX	801163	29.25	-103.25
BISC	Biscayne National Park	FL	172924	25.65	-80.08
BLCA	Black Canyon of the Gunnison National Park	CO	32950	38.57	-107.72
BRCA	Bryce Canyon National Park	UT	35835	37.57	-112.18
CANY	Canyonlands National Park	UT	337598	38.2	-109.93
CARE	Capitol Reef National Park	UT	241904	38.2	-111.17
CAVE	Carlsbad Caverns National Park	NM	46766	32.17	-104.44
CHIS	Channel Islands National Park	CA	249561	34.01	-119.42
CONG	Congaree National Park	SC	26546	33.78	-80.78
CRLA	Crater Lake National Park	OR	183224	42.94	-122.1
CUVA	Cuyahoga Valley National Park	OH	32950	41.24	-81.55
DENA	Denali National Park and Preserve	AK	3372402	63.33	-150.5
DEVA	Death Valley National Park	CA, NV	4740912	36.24	-116.82

Download csv file

- Download the dataset from itslearning
  - Resources/Part2: Data Extraction and Visualization/Dataset/**parks.csv** (US national park data)

Open in Excel

	A	B	C	D	E	F
1	Park Code	Park Name	State	Acres	Latitude	Longitude
2	ACAD	Acadia National Pa	ME	47390	44.35	-68.21
3	ARCH	Arches National Pa	UT	76519	38.68	-109.57
4	BADL	Badlands National	SD	242756	43.75	-102.5
5	BIBE	Big Bend National F	TX	801163	29.25	-103.25
6	BISC	Biscayne National F	FL	172924	25.65	-80.08
7	BLCA	Black Canyon of the	CO	32950	38.57	-107.72
8	BRCA	Bryce Canyon Natic	UT	35835	37.57	-112.18
9	CANY	Canyonlands Natio	UT	337598	38.2	-109.93
10	CARE	Capitol Reef Natior	UT	241904	38.2	-111.17
11	CAVE	Carlsbad Caverns N	NM	46766	32.17	-104.44
12	CHIS	Channel Islands Na	CA	249561	34.01	-119.42
13	CONG	Congaree National	SC	26546	33.78	-80.78
14	CRLA	Crater Lake Nationa	OR	183224	42.94	-122.1
15	CUVA	Cuyahoga Valley Na	OH	32950	41.24	-81.55
16	DENA	Denali National Par	AK	3372402	63.33	-150.5
17	DEVA	Death Valley Natio	CA, NV	4740912	36.24	-116.82
18	DRTO	Dry Tortugas Natior	FL	64701	24.63	-82.87
19	EVER	Everglades Nationa	FL	1508538	25.32	-80.93
20	GAAR	Gates Of The Arctic	AK	7523898	67.78	-153.3
21	GLAC	Glacier National Pa	MT	1013572	48.8	-114
22	GLBA	Glacier Bay Nationa	AK	3224840	58.5	-137
23	GRBA	Great Basin Nationa	NV	77180	38.98	-114.3
24	GRCA	Grand Canyon Nati	AZ	1217403	36.06	-112.14

Open in NotePad or TextEdit(Mac)

parks.csv

```
Park Code,Park Name,State,Acres,Latitude,Longitude
ACAD,Acadia National Park,ME,47390,44.35,-68.21
ARCH,Arches National Park,UT,76519,38.68,-109.57
BADL,Badlands National Park,SD,242756,43.75,-102.5
BIBE,Big Bend National Park,TX,801163,29.25,-103.25
BISC,Biscayne National Park,FL,172924,25.65,-80.08
BLCA,Black Canyon of the Gunnison National Park,CO,32950,38.57,-107.72
BRCA,Bryce Canyon National Park,UT,35835,37.57,-112.18
CANY,Canyonlands National Park,UT,337598,38.2,-109.93
CARE,Capitol Reef National Park,UT,241904,38.2,-111.17
CAVE,Carlsbad Caverns National Park,NM,46766,32.17,-104.44
CHIS,Channel Islands National Park,CA,249561,34.01,-119.42
CONG,Congaree National Park,SC,26546,33.78,-80.78
CRLA,Crater Lake National Park,OR,183224,42.94,-122.1
CUVA,Cuyahoga Valley National Park,OH,32950,41.24,-81.55
DENA,Denali National Park and Preserve,AK,3372402,63.33,-150.5
DEVA,Death Valley National Park,"CA, NV",4740912,36.24,-116.82
DRTO,Dry Tortugas National Park,FL,64701,24.63,-82.87
EVER,Everglades National Park,FL,1508538,25.32,-80.93
GAAR,Gates Of The Arctic National Park and Preserve,AK,7523898,67.78,-153
GLAC,Glacier National Park,MT,1013572,48.8,-114
GLBA,Glacier Bay National Park and Preserve,AK,3224840,58.5,-137
GRBA,Great Basin National Park,NV,77180,38.98,-114.3
GRCA,Grand Canyon National Park,AZ,1217403,36.06,-112.14
GRSA,Great Sand Dunes National Park and Preserve,CO,42984,37.73,-105.51
GRSM,Great Smoky Mountains National Park,"TN, NC",521490,35.68,-83.53
GRTE,Grand Teton National Park,WY,309995,43.73,-110.8
GUMO,Guadalupe Mountains National Park,TX,86416,31.92,-104.87
HALE,Haleakala National Park,HI,29094,20.72,-156.17
HAVO,Hawaii Volcanoes National Park,HI,323431,19.38,-155.2
HOSP,Hot Springs National Park,AR,5550,34.51,-93.05
ISRO,Isle Royale National Park,MI,571790,48.1,-88.55
```

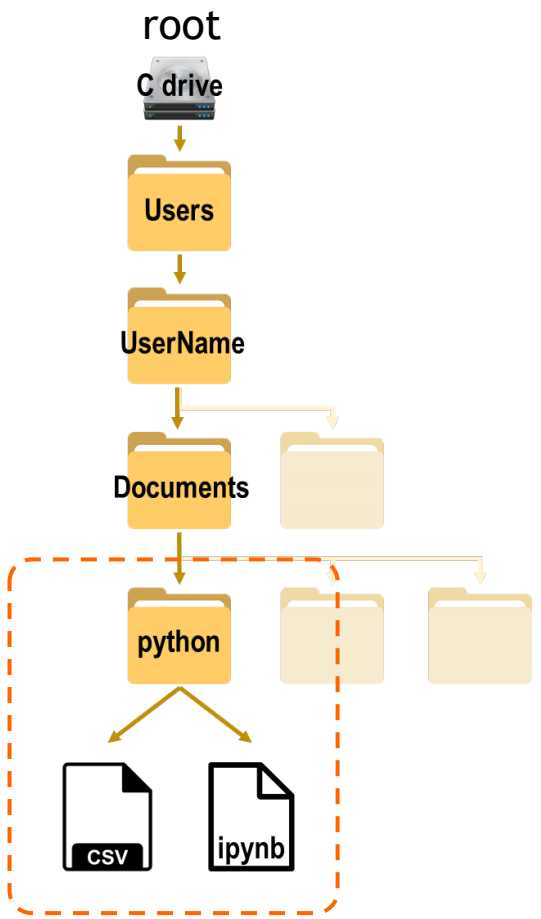
- A .csv file ("comma separated value") is a file where the data is stored in a text format, separated by commas (or other separation marks such as space or semi-colon).

Absolute path and relative path

- An absolute path is defined as specifying the location of a file from the root directory. An absolute path is a complete path.

windows
```
df = pd.read_csv("C:/Users/UserName/Documents/python/park.csv")
```

Mac
```
df = pd.read_csv("/Users/UserName/Documents/python/park.csv")
```



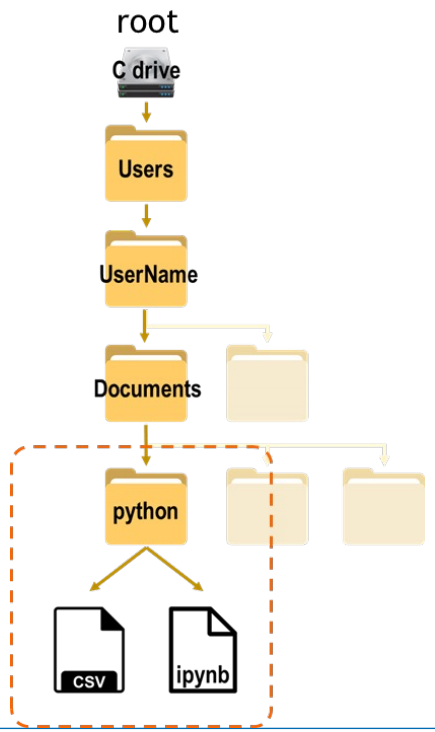
- A relative path is defined as the location related to the current working file. Pandas can find the file from where your notebook is running.

```
df = pd.read_csv("park.csv")
```

Read csv file into a jupyter notebook

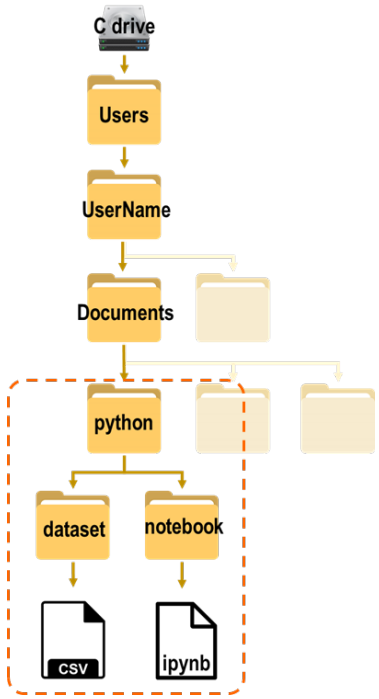
Case1: If you put the csv file in the same folder as jupyter notebook.

```
park_df = pd.read_csv('parks.csv')
```



Case2: If you put csv file in a different folder.

```
park_df = pd.read_csv('../dataset/parks.csv')
```



- In this example, the file "parks.csv" is stored one folder back (../), then inside the folder “dataset".

View data

- View the first N rows

```
park_df.head()
```

default is 5

	Park Code	Park Name	State	Acres	Latitude	Longitude
0	ACAD	Acadia National Park	ME	47390	44.35	-68.21
1	ARCH	Arches National Park	UT	76519	38.68	-109.57
2	BADL	Badlands National Park	SD	242756	43.75	-102.50
3	BIBE	Big Bend National Park	TX	801163	29.25	-103.25
4	BISC	Biscayne National Park	FL	172924	25.65	-80.08

```
park_df.head(10)
```

	Park Code	Park Name	State	Acres	Latitude	Longitude
0	ACAD	Acadia National Park	ME	47390	44.35	-68.21
1	ARCH	Arches National Park	UT	76519	38.68	-109.57
2	BADL	Badlands National Park	SD	242756	43.75	-102.50
3	BIBE	Big Bend National Park	TX	801163	29.25	-103.25
4	BISC	Biscayne National Park	FL	172924	25.65	-80.08
5	BLCA	Black Canyon of the Gunnison National Park	CO	32950	38.57	-107.72
6	BRCA	Bryce Canyon National Park	UT	35835	37.57	-112.18
7	CANY	Canyonlands National Park	UT	337598	38.20	-109.93
8	CARE	Capitol Reef National Park	UT	241904	38.20	-111.17
9	CAVE	Carlsbad Caverns National Park	NM	46766	32.17	-104.44

Descriptive statistics - info

- A summary of a DataFrame
  - Index, data type of each columns, number of non-null values, and memory usage.

```
park_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Park Code  56 non-null     object 
 1   Park Name  56 non-null     object 
 2   State      56 non-null     object 
 3   Acres      56 non-null     int64  
 4   Latitude   56 non-null     float64
 5   Longitude  56 non-null     float64
dtypes: float64(2), int64(1), object(3)
memory usage: 2.8+ KB
```

In Pandas, string data is always stored with an object dtype.

Exercise

Exercise.B

(B.1) Read the csv file `diabetes.csv` as a pandas dataframe. Display the first 10 rows.

- **Pregnancies**: Number of times pregnant
- **Glucose**: Plasma glucose concentration over 2 hours in an oral glucose tolerance test
- **BloodPressure**: Diastolic blood pressure (mm Hg)
- **SkinThickness**: Triceps skin fold thickness (mm)
- **Insulin**: 2-Hour serum insulin (mu U/ml)
- **BMI**: Body mass index (weight in kg/(height in m)2)
- **DiabetesPedigreeFunction**: Diabetes pedigree function (a function which scores likelihood of diabetes based on family history)
- **Age**: Age (years)
- **Outcome**: Class variable (0 if non-diabetic, 1 if diabetic)

(B.2) What are the number of rows and columns of the dataframe you obtained in (A.1)?

(B.3) Show a summary of the dataframe, including column names and their data types.

Descriptive statistics - pandas data types

- Pandas `dtype` mapping

Pandas dtype	Python type	Usage
int64	int	Integer numbers
float64	float	Floating point numbers
object	str or mixed	Text or mixed numeric and non-numeric values
bool	bool	True/False values
datetime64	datetime	Date and time values
timedelta[ns]	--	Differences between two datetimes
category	--	Finite list of text values

numerical data (quantitative)

categorical data (qualitative data)

- Numpy data type (dtype) https://numpy.org/doc/stable/reference/arrays.dtypes.html
- Example: int64 (64-bit integers): -9223372036854775808 to 9223372036854775807

Descriptive statistics – categorical data and numerical data

- **Categorical data** describes characteristics or groups.
  - Gender (Male, Female)
  - Product types (Wood, Plastic, Metal)
  - Risk level (Low, Medium, High)
  - Days of the week (Monday, Tuesday, Wednesday)
  - Months of the year (January, February, March)

- **Numerical data** expresses information in the form of numbers.
  - Number of customers (25, 19, 35,...)
  - Prices of products (200, 150, 80,...)
  - Interest rate (0.5, 1.4, 2.3,...)

Descriptive statistics – change data type

- Use `astype()` to change the data type (dtype).

```
park_df["Acres"] = park_df["Acres"].astype(float)
park_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Park Code   56 non-null     object
 1   Park Name   56 non-null     object
 2   State       56 non-null     object
 3   Acres       56 non-null     float64
 4   Latitude    56 non-null     float64
 5   Longitude   56 non-null     float64
dtypes: float64(3), object(3)
memory usage: 2.8+ KB
```

or

Another way to select a column.

```
park_df.Acres = park_df.Acres.astype(float)
```

Descriptive statistics – numerical column

- Use the `describe()` to obtain descriptive statistics for all numeric columns.
- Use `round()` to round a DataFrame to a specified number of decimal places.

```
park_df.describe()
```

	Acres	Latitude	Longitude
count	5.600000e+01	56.000000	56.000000
mean	9.279291e+05	41.233929	-113.234821
std	1.709258e+06	10.908831	22.440287
min	5.550000e+03	19.380000	-159.280000
25%	6.901050e+04	35.527500	-121.570000
50%	2.387645e+05	38.550000	-110.985000
75%	8.173602e+05	46.880000	-103.400000
max	8.323148e+06	67.780000	-68.210000

```
park_df.describe().round(2)
```

	Acres	Latitude	Longitude
count	56.00	56.00	56.00
mean	927929.14	41.23	-113.23
std	1709258.31	10.91	22.44
min	5550.00	19.38	-159.28
25%	69010.50	35.53	-121.57
50%	238764.50	38.55	-110.98
75%	817360.25	46.88	-103.40
max	8323148.00	67.78	-68.21

- https://en.wikipedia.org/wiki/Standard_deviation
- https://en.wikipedia.org/wiki/Quartile
- https://en.wikipedia.org/wiki/Scientific_notation

Descriptive statistics – numerical column

- Descriptive statistics for a single numerical column.

```
park_df["Acres"].describe()
```

```
count    5.600000e+01
mean     9.279291e+05
std      1.709258e+06
min      5.550000e+03
25%      6.901050e+04
50%      2.387645e+05
75%      8.173602e+05
max      8.323148e+06
Name: Acres, dtype: float64
```

or

```
park_df["Acres"].count()
```

```
56
```

```
park_df["Acres"].mean()
```

```
927929.1428571428
```

```
park_df["Acres"].std()
```

```
1709258.309313917
```

```
park_df["Acres"].min()
```

```
5550.0
```

```
park_df["Acres"].quantile(0.25)
```

```
69010.5
```

Descriptive statistics – categorical column

- Use `value_counts()` to get the count of each unique value of a categorical column.

```
park_df.State.value_counts()
```

```
AK            8
CA            7
UT            5
CO            4
FL            3
AZ            3
WA            3
SD            2
TX            2
HI            2
ME            1
MI            1
ND            1
KY            1
VA            1
MN            1
AR            1
MT            1
WY            1
TN, NC        1
NV            1
CA, NV        1
OH            1
OR            1
SC            1
NM            1
WY, MT, ID    1
Name: State, dtype: int64
```

or

```
park_df["State"].value_counts()
```

```
AK            8
CA            7
UT            5
CO            4
FL            3
AZ            3
WA            3
SD            2
TX            2
HI            2
ME            1
MI            1
ND            1
KY            1
VA            1
MN            1
AR            1
MT            1
WY            1
TN, NC        1
NV            1
CA, NV        1
OH            1
OR            1
SC            1
NM            1
WY, MT, ID    1
Name: State, dtype: int64
```

Exercise

Exercise.C

(C.1) Use the dataframe obtained in (B.1). Change the data type of `Outcome` to `object`.

(C.2) Show the descriptive statistics for the `Age` column of the dataframe.

(C.3) The `Outcome` column indicates whether an individual has diabetes, where 0 represents non-diabetic and 1 represents diabetic. How many individuals in this dataset have diabetes?

Sort

Sort a DataFrame

- Use the method `sort_values()` to sort a DataFrame.
- The data is sorted in ascending order by default but can be sorted in descending order by specifying `ascending = False`.

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Nevada	2003	3.2

```
df.sort_values(by = "year")
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
3	Nevada	2001	2.4
2	Ohio	2002	3.6
4	Nevada	2002	2.9
5	Nevada	2003	3.2


```
df.sort_values(by = "year", ascending = False)
```

	state	year	pop
5	Nevada	2003	3.2
2	Ohio	2002	3.6
4	Nevada	2002	2.9
1	Ohio	2001	1.7
3	Nevada	2001	2.4
0	Ohio	2000	1.5

Sort a DataFrame – by multiple columns

- Pass a list of column names.

```
df.sort_values(by = ["state", "year"])
```

	state	year	pop
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Nevada	2003	3.2
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6

```
df.sort_values(by = ["state", "year"], ascending = [True, False])
```

	state	year	pop
5	Nevada	2003	3.2
4	Nevada	2002	2.9
3	Nevada	2001	2.4
2	Ohio	2002	3.6
1	Ohio	2001	1.7
0	Ohio	2000	1.5

Sort a DataFrame - inplace

- Recall: In-place methods are methods that directly modify the variable they are applied to.
- Most DataFrame methods are NOT in-place methods by default.
- To keep the resulting dataframe:

Option-1: Store the result in a new variable

```
df_sorted = df.sort_values(by = "year")
df_sorted
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
3	Nevada	2001	2.4
2	Ohio	2002	3.6
4	Nevada	2002	2.9
5	Nevada	2003	3.2

Option-2: Use `inplace = True`

```
df.sort_values(by = "year", inplace = True)
df
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
3	Nevada	2001	2.4
2	Ohio	2002	3.6
4	Nevada	2002	2.9
5	Nevada	2003	3.2

Reset index

- Reset the index of the DataFrame after sorting.
  - `drop = True`: Drop the old index.
  - `drop = False (default)`: Add the old index as an additional column to your DataFrame.

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
3	Nevada	2001	2.4
2	Ohio	2002	3.6
4	Nevada	2002	2.9
5	Nevada	2003	3.2

```
df.reset_index()
```

	index	state	year	pop
0	0	Ohio	2000	1.5
1	1	Ohio	2001	1.7
2	3	Nevada	2001	2.4
3	2	Ohio	2002	3.6
4	4	Nevada	2002	2.9
5	5	Nevada	2003	3.2

```
df.reset_index(drop = True)
```

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Nevada	2001	2.4
3	Ohio	2002	3.6
4	Nevada	2002	2.9
5	Nevada	2003	3.2

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

Exercise

Exercise.D

(D.1) Use the dataframe obtained in (C.1). Sort the dataframe in ascending order based on the `Age` column and store the result in a new variable. Display the result.

(D.2) Use the dataframe obtained in (D.1). Reset the index and drop the old one.