BINF5007

Lecture 9: Pandas

Learning Objectives

1. **Pandas Data Structures:** Understand and differentiate between *Series and DataFrame.*
2. **Pandas Integration**: Utilize pandas with libraries like NumPy and Matplotlib.
3. **DataFrame Basics**: Create, modify, and explore DataFrames.
4. **Statistical Analysis**: Compute basic statistics on numerical data.
5. **Data Import/Export**: Read and write data to formats like Excel and CSV.
6. **Data Subsetting**: Extract rows, columns, and subsets using indexing and filtering.

The two primary data structures of pandas

- Series (1-dimensional) and DataFrame (2-dimensional)

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

- These handle the majority of use cases in finance, statistics, social science, and many areas of engineering
- For R users, the DataFrame provides everything that R's data.frame provides and more
- pandas is built on top of NumPy (see on your own section) and is intended to integrate well within a scientific computing environment with many other 3rd party libraries

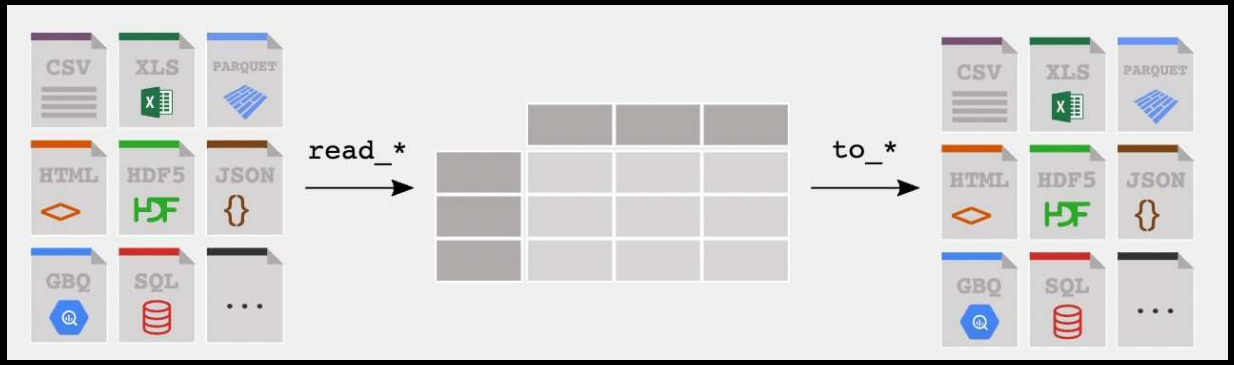
Wes McKinney and the pandas Development Team

Why more than one data structure?

- The best way to think about the pandas data structures is as flexible containers for lower dimensional data
- For example, DataFrame is a container for Series, and Series is a container for scalars
- We can insert and remove objects from these containers in a dictionary-like fashion
- pandas has sensible default behaviors for the common API functions which take into account the typical orientation of time series and cross-sectional data sets

pandas Integration

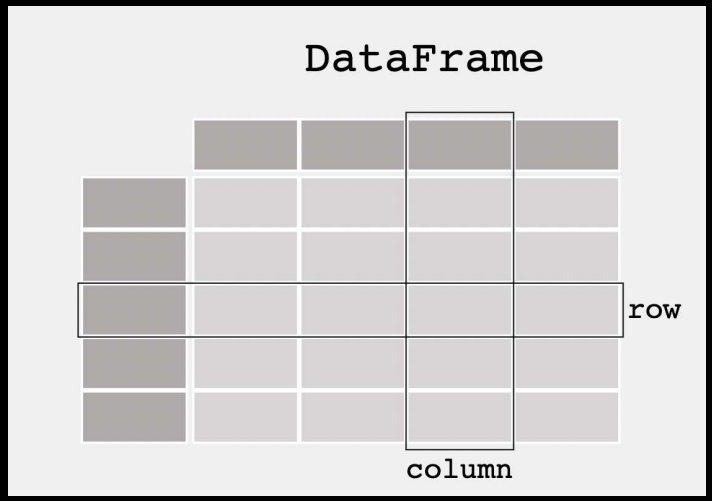
- pandas supports the integration with many file formats or data sources out of the box (csv, excel, sql, json, parquet, etc)
- Importing data from each of these data sources is provided by function with the prefix `read_*`
- Similarly, the `to_*` methods are used to store data



Wes McKinney and the pandas Development Team

pandas DataFrame

- When working with tabular data like the data stored in spreadsheets or databases, pandas is a good tool
- pandas will help you to explore, clean, and process your data
- **In pandas, a data table is called a DataFrame**

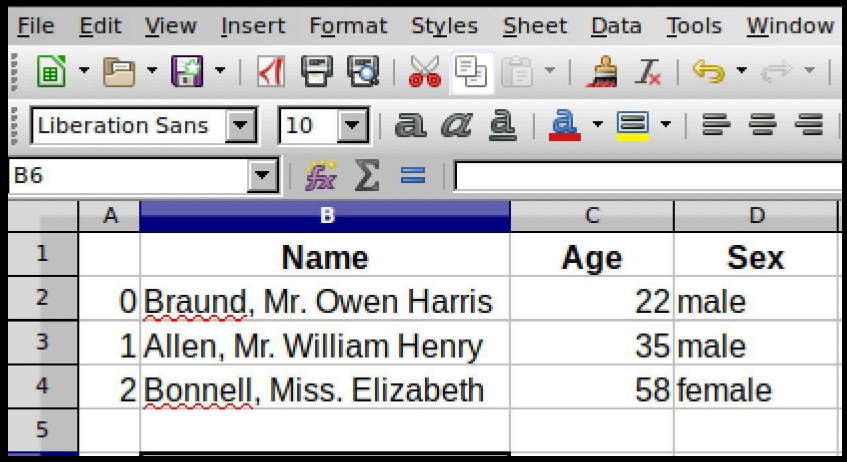


- DataFrames are just another data structure (ds), and we can generate DataFrames using the data structures we learned about last week

Adopted from Wes McKinney and the pandas Development Team

DataFrame: 2-dimensional data structures

- It can store data of different types (including characters, integers, floating point values, categorical data and more) in columns
- It is like a spreadsheet, a SQL table or the data frame in R
- This table has 3 columns, each of them with a column label. The column labels are respectively `Name`, `Age`, and `Sex`
- The column `Name` consists of textual data with each value a string, the column `Age` are numbers, and the column `Sex` is textual data



	A	B	C	D
1		Name	Age	Sex
2	0	Braund, Mr. Owen Harris	22	male
3	1	Allen, Mr. William Henry	35	male
4	2	Bonnell, Miss. Elizabeth	58	female
5				

pandas Data Table Representation

- To load the pandas package and start working with it, import the package
- The community agreed alias for pandas is `pd`, so loading pandas as `pd` is assumed standard practice for all of the pandas documentation
- To manually store data in a table, create a **DataFrame**
- When using a Python **dictionary of lists**, the dictionary keys will be used as column headers and the values in each list as columns of the DataFrame

```
<class 'pandas.core.frame.DataFrame'>
```

```python
import pandas as pd

df = pd.DataFrame(
    {
        "Name": [
            "Braund, Mr. Owen Harris",
            "Allen, Mr. William Henry",
            "Bonnell, Miss. Elizabeth",
        ],
        "Age": [22, 35, 58],
        "Sex": ["male", "male", "female"],
    }
)
print(df)
```

What's this?

Yes, we generate DataFrames using the data structures we learned about last week

```
                       Name  Age     Sex
0   Braund, Mr. Owen Harris   22    male
1  Allen, Mr. William Henry   35    male
2  Bonnell, Miss. Elizabeth   58  female
```

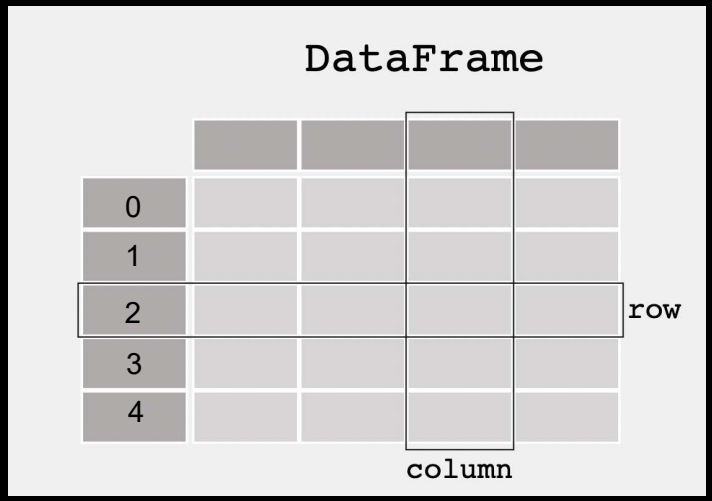
Wes McKinney and the pandas Development Team

Thinking About the DataFrame

- With a DataFrame it is more semantically helpful to think of the index (the rows) and the columns rather than axis 0 and axis 1. Iterating through the columns of the DataFrame thus results in more readable code:

```
for col in df.columns:
    series = df[col]
    print(series)
```

```
0     Braund, Mr. Owen Harris
1     Allen, Mr. William Henry
2     Bonnell, Miss. Elizabeth
Name: Name, dtype: object
0     22
1     35
2     58
Name: Age, dtype: int64
1     male
2     male
2.    female
Name: Sex, dtype: object
```



Wes McKinney and the pandas Development Team

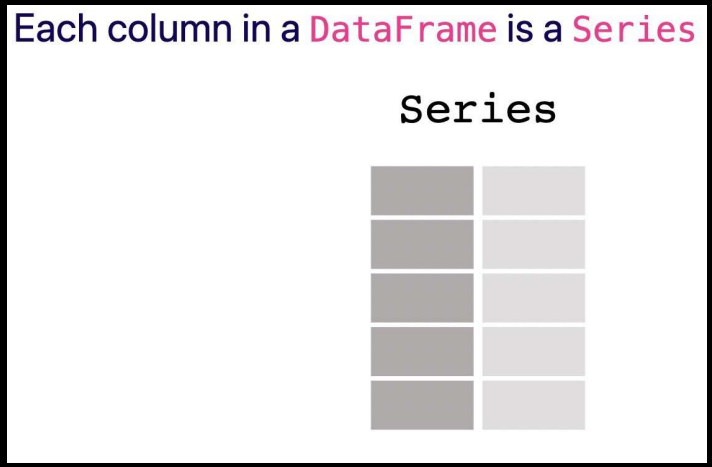
A Series (Column in a DataFrame)

- Interested in working with the data in the column `Age` from our previous example?
- When selecting a single column of a pandas DataFrame, the result is a pandas Series
- To select the column, use the column label in between square brackets `[]`

```
print(df["Age"])
```

```
0    22
1    35
2    58
Name: Age, dtype: int64
```

The selection of a single column is very similar to selection of dictionary values based on the key



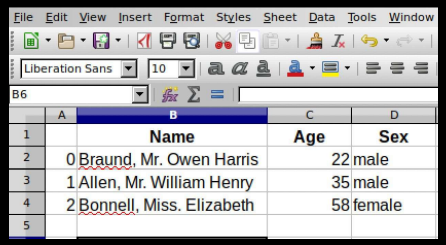

```
<class 'pandas.core.series.Series'>
```

Working with a DataFrame or Series

- Want to know the maximum `Age` of the passenger?
- We can do this on the DataFrame by selecting the `Age` column and applying `max()`:

```
max_age = df["Age"].max()
print(max_age)
max_age = ages.max()
print(max_age)
```

```
58
58
```

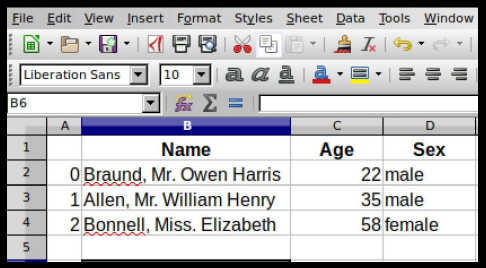


	A	B	C	D
1		Name	Age	Sex
2	0	Braund, Mr. Owen Harris	22	male
3	1	Allen, Mr. William Henry	35	male
4	2	Bonnell, Miss. Elizabeth	58	female
5				

- As illustrated by the `max()` method, you can do things with a DataFrame or Series
- pandas provides a lot of functionalities, through methods that you can apply to a DataFrame or Series. As methods are functions, **you need parentheses ()**

Statistics of the Numerical Data

- Getting basic statistics of the numerical data of my data table
- The `describe()` method provides a quick overview of the numerical data in a DataFrame
- As the `Name` and `Sex` columns are textual data, these are by default not considered by the `describe()` method
- Many pandas' operations return a DataFrame or a Series
- The `describe()` method is an example of a pandas operation returning a pandas Series or a pandas DataFrame



	A	Name	Age	Sex
1				
2	0	Braund, Mr. Owen Harris	22	male
3	1	Allen, Mr. William Henry	35	male
4	2	Bonnell, Miss. Elizabeth	58	female
5				

```
print(df.describe())
```

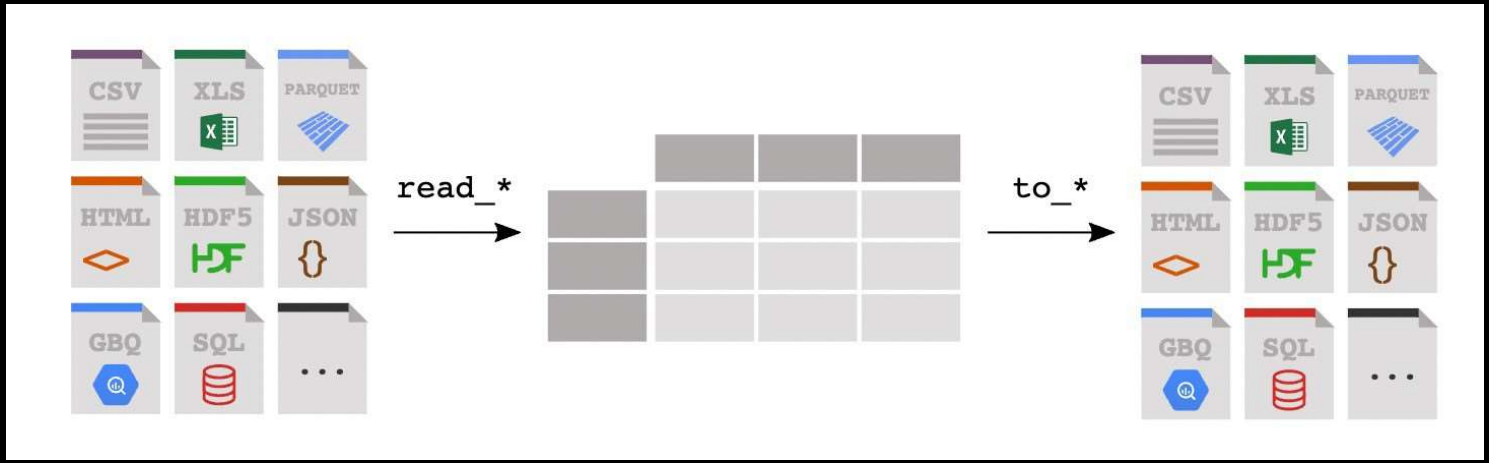
```
              Age
count    3.000000
mean    38.333333
std     18.230012
min     22.000000
25%     28.500000
50%     35.000000
75%     46.500000
max     58.000000
```

Remember

- Import the package, aka `import pandas as pd`
- A table of data is stored as a pandas DataFrame
- Each column in a DataFrame is a Series
- You can do things by applying a method to a DataFrame or Series

Reading in Tabular Data

How do I read and write tabular data?



- Importing data from each of these data sources is provided by function with the prefix `read_*`. Similarly, the `to_*` methods are used to store data
- pandas provides the `read_csv()` function to read data stored as a csv file into a pandas DataFrame

Wes McKinney and the pandas Development Team

Getting the Titanic tabular data for this section

- PassengerId: Id of every passenger.
- Survived: This feature have value 0 and 1. 0 for not survived and 1 for survived.
- Pclass: There are 3 classes: Class 1, Class 2 and Class 3.
- Name: Name of passenger.
- Sex: Gender of passenger.
- Age: Age of passenger.
- SibSp: Indication that passenger have siblings and spouse.
- Parch: Whether a passenger is alone or have family.
- Ticket: Ticket number of passenger.
- Fare: Indicating the fare.
- Cabin: The cabin of passenger.
- Embarked: The embarked category.

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,1,0,237736,30.0708,,C
11,1,3,"Sandstrom, Miss. Marguerite Rut",female,4,1,1,PP 9549,16.7,G6,S
12,1,1,"Bonnell, Miss. Elizabeth",female,58,0,0,113783,26.55,C103,S
```

https://raw.githubusercontent.com/pandas-dev/pandas/main/doc/data/titanic.csv

```
wget https://raw.githubusercontent.com/pandas-dev/pandas/main/doc/data/titanic.csv
```

Checking The Data Import

- Good idea to check on the data after reading in the data
- When displaying a DataFrame, the first and last 5 rows will be shown by default:

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
print(titanic)
```

```
     PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0              1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1              2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C
2              3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3              4         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1      0            113803  53.1000  C123        S
4              5         0       3                           Allen, Mr. William Henry    male  35.0      0      0            373450   8.0500   NaN        S
..           ...       ...     ...                                                ...     ...   ...    ...    ...               ...      ...   ...      ...
886          887         0       2                              Montvila, Rev. Juozas    male  27.0      0      0            211536  13.0000   NaN        S
887          888         1       1                       Graham, Miss. Margaret Edith  female  19.0      0      0            112053  30.0000   B42        S
888          889         0       3           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1      2        W./C. 6607  23.4500   NaN        S
889          890         1       1                              Behr, Mr. Karl Howell    male  26.0      0      0            111369  30.0000  C148        C
890          891         0       3                                Dooley, Mr. Patrick    male  32.0      0      0            370376   7.7500   NaN        Q
```

- To see the first N rows of a DataFrame, use the `head()` method with the required number of rows (in this case 8) as argument: `print(titanic.head(8))`
- To see the last N rows: `titanic.tail(10)`

https://raw.githubusercontent.com/pandas-dev/pandas/main/doc/data/titanic.csv

Check How pandas Interpreted each of the column data types

- Good idea to check the types (later on we'll see how import with typing)

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
print(titanic.dtypes)
```

- For each of the columns, the used data type is enlisted. The data types in this DataFrame are integers (`int64`), floats (`float64`) and strings (`object`)
- When asking for the `dtypes`, no parenthesis are used, b/c it's an attribute of a DataFrame or Series
- Attributes represent a characteristic of a DataFrame/Series, whereas a method (which requires parenthesis) does something with the DataFrame/Series

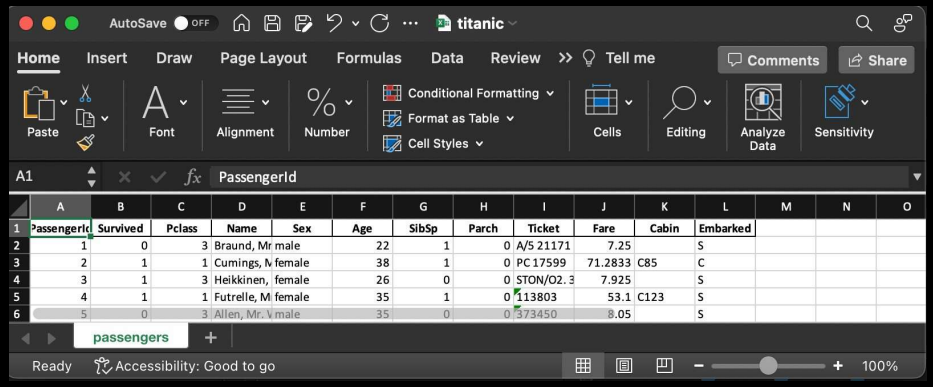
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Print Data to Excel in One Line

- Print the Titanic data to an excel spreadsheet

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
titanic.to_excel("titanic.xlsx", sheet_name="passengers", index=False)
```



If you get a: `ModuleNotFoundError: No module named 'openpyxl'`
Perform a: `pipenv install openpyxl # or pip install openpyxl`

`read_*` functions are used to read data to pandas, and the `to_*` methods are used to store data. The `to_excel()` method stores the data as an excel file. In the example here, the `sheet_name` is named passengers instead of the default `Sheet1`

By setting `index=False` the row index labels are not saved in the spreadsheet

The equivalent `read` function `read_excel()` will reload the data to a DataFrame

Wes McKinney and the pandas Development Team

Technical Information About a DataFrame

- The **method** `info()` provides more technical information about a DataFrame than `dytpes`

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
print(titanic.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

Let's explain the output in more detail: It's a DataFrame

- There are 891 entries, i.e., 891 rows
- Each row has a row label (aka the index) with values ranging from 0 to 890
- The table has 12 columns. **Most** columns have a value for each of the rows (all 891 values are non-null). Some columns do have missing values and less than 891 non-null values
- The columns `Name`, `Sex`, `Cabin` and `Embarked` consists of textual data (strings, aka `object`). The other columns are numerical data with some of them whole numbers (aka `integer`) and others are real numbers (aka `float`)
- The kind of data (characters, integers,…) in the different columns are summarized by listing the `dtypes`
- The approximate amount of RAM used to hold the DataFrame is provided as well

Wes McKinney and the pandas Development Team

Remember

- Import the package, aka `import pandas as pd`
- A table of data is stored as a pandas DataFrame
- Each column in a DataFrame is a Series
- You can do things by applying a method to a DataFrame or Series
- Getting data into pandas from many different file formats or data sources is supported by `read_*` functions
- Exporting data out of pandas is provided by different `to_*` methods
- The `head`/`tail`/`info` methods and the `dtypes` attribute are convenient for a first check

Getting a Subset from a DataFrame

How do I select specific columns from a DataFrame?

- Selecting specific columns from a DataFrame:

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
ages = titanic["Age"]
print(ages)
print(ages.shape))
```

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
(891,)
```

To select a single column, use square brackets `[]` with the column name of the column of interest

Each column in a DataFrame is a Series. As a single column is selected, the returned object is a pandas Series**. How could you verify?**

`DataFrame.shape` is an attribute of a pandas Series and DataFrame containing the number of rows and columns: (nrows, ncolumns). A pandas Series is 1-dimensional and only the number of rows is returned

How do I select specific columns from a DataFrame?

- How do I select **multiple** columns from a DataFrame?

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
age_sex = titanic[["Age", "Sex"]]
print(age_sex)
print(age_sex.shape)
```

```
      Age     Sex
0    22.0    male
1    38.0  female
2    26.0  female
3    35.0  female
4    35.0    male
..    ...     ...
886  27.0    male
887  19.0  female
888   NaN  female
889  26.0    male
890  32.0    male

[891 rows x 2 columns]
(891, 2)
```

To select a multiple columns, use square brackets [] with the column names of interest

When multiple column are selected, the returned object is a pandas DataFrame**. How could you verify?**

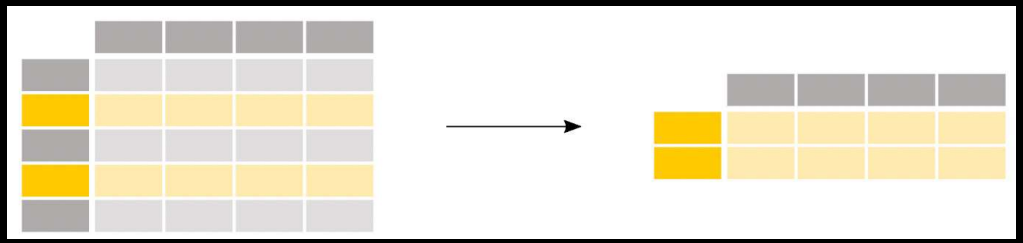
`DataFrame.shape` is an attribute of a pandas Series and DataFrame containing the number of rows and columns: (nrows, ncolumns). A pandas DataFrame is 2-dimensional, so the the number of rows and columns is returned

How do I filter specific rows from a DataFrame?

- Interested in the passengers older than 35 years:

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
above_35 = titanic[titanic["Age"] > 35]
print(above_35.head())
```



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S

To select rows based on a conditional expression, use a condition inside the selection brackets `[]`

How do I filter specific rows from a DataFrame?

- Technically speaking, how did the previous expression work?
- The output of the conditional expression (>, but also ==, !=, <, <=,… would work) is a pandas Series of boolean values (either `True` or `False`) with the same number of rows as the original DataFrame
- A Series of boolean values can filter the DataFrame by putting it b/t the selection brackets `[]`
- Only rows for which the value is `True` will be selected:
  `above_35 = titanic[titanic["Age"] > 35]`
- The original Titanic DataFrame consists of 891 rows. How many rows are in **`above_35`**?
- What type is **`above_35`**?

```
>>> titanic["Age"] > 35
Out[14]:
0       False
1        True
2       False
3       False
4       False
        ...
886     False
887     False
888     False
889     False
890     False
Name: Age, Length: 891, dtype: bool
```

```
>>> above_35.shape[0]
217
>>> len(above_35)
217

>>> type(above_35)
<class 'pandas.core.frame.DataFrame'>
```

DataFrames Can Have `NaN` Values

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
..	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

How can we work with passenger data for which the age is **known, i.e. not `NaN` ?**

How do I filter specific rows from a DataFrame?

- Working with passenger data for which the age is **known, i.e. not `NaN`**
- The **`notna()`** conditional function returns a `True` for each row the values are not a `Null` value, and this can be combined with the selection brackets `[]` to filter the data table

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
age_no_na = titanic[titanic["Age"].notna()]
print(age_no_na.head())
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

- How can you check how many rows were found? `len(age_no_na)`

How do I select specific rows and columns from a DataFrame

- Interested in the names of the passengers older than 35 years

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
adult_names = titanic.loc[titanic["Age"] > 35, "Name"]
print(adult_names.head())
print(type(adult_names.head()))
```

```
1     Cumings, Mrs. John Bradley (Florence Briggs Th...
6                               McCarthy, Mr. Timothy J
11                             Bonnell, Miss. Elizabeth
13                           Andersson, Mr. Anders Johan
15                       Hewlett, Mrs. (Mary D Kingcome)
Name: Name, dtype: object
<class 'pandas.core.series.Series'>
```

Using selection brackets `[]` is **NOT** sufficient anymore for a subset of both rows and columns

Instead, the `loc/iloc` operators are required in front of the selection brackets `[]`

When using `loc/iloc`, the part **before** the comma is the **rows** you want, and the part **after** the comma is the **columns** you want to selectl

You can use a list to return multiple columns, but what would it return?

How do I select specific rows and columns from a DataFrame

- For both the part **before** and **after** the comma, you can use a single label, a list of labels, a slice of labels, a conditional expression or a colon. Using a colon specifies you want to select all rows or columns

```
import pandas as pd

titanic = pd.read_csv("data/titanic.csv")
print(titanic.head())
sub_titanic = titanic.iloc[9:25, 2:5] # rows, columns
print(sub_titanic.head())
```

```
   PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0            1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C
2            3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3            4         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1      0            113803  53.1000  C123        S
4            5         0       3                           Allen, Mr. William Henry    male  35.0      0      0            373450   8.0500   NaN        S
    Pclass                                Name     Sex
9        2  Nasser, Mrs. Nicholas (Adele Achem)  female
10       3      Sandstrom, Miss. Marguerite Rut  female
11       1             Bonnell, Miss. Elizabeth  female
12       3       Saundercock, Mr. William Henry    male
13       3          Andersson, Mr. Anders Johan    male
```

Interested in rows indexes 9 - 24 and columns 2 – 4

9:25

Chesley Leslin

Remember

- When selecting subsets of data, square brackets `[]` are used
- Inside these brackets, you can use a single column/row label, a list of column/row labels, a slice of labels, a conditional expression or a colon
- Select specific rows and/or columns using `loc` when using the row and column **names**
- Select specific rows and/or columns using `iloc` when using the **positions** in the table
- You can assign new values to a selection based on `loc/iloc`

Your Turn

Find all anonymous passengers in the dataset