Python Programming

Lesson 8 – Hands-on with Pandas

Lesson 8 - Outline

- Recap the basic of Pandas
- Data processing/cleansing/analysis with Pandas

Recap the basic of Pandas

Recap the basic of Pandas

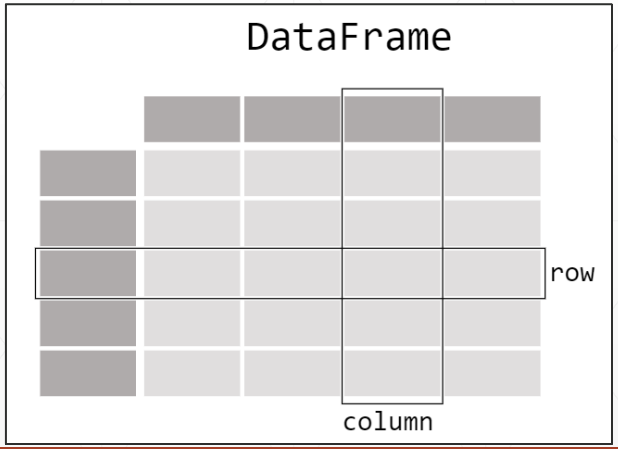
- **Install of Pandas module**

```
pip install pandas
```

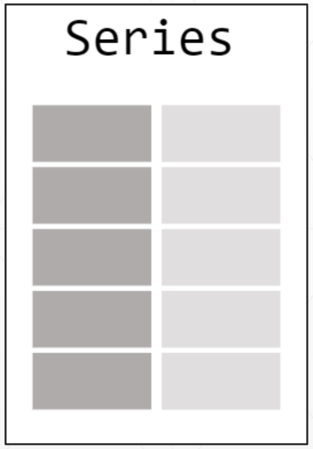
- **Import Pandas module**

```
import pandas as pd
```

Pandas data table representation



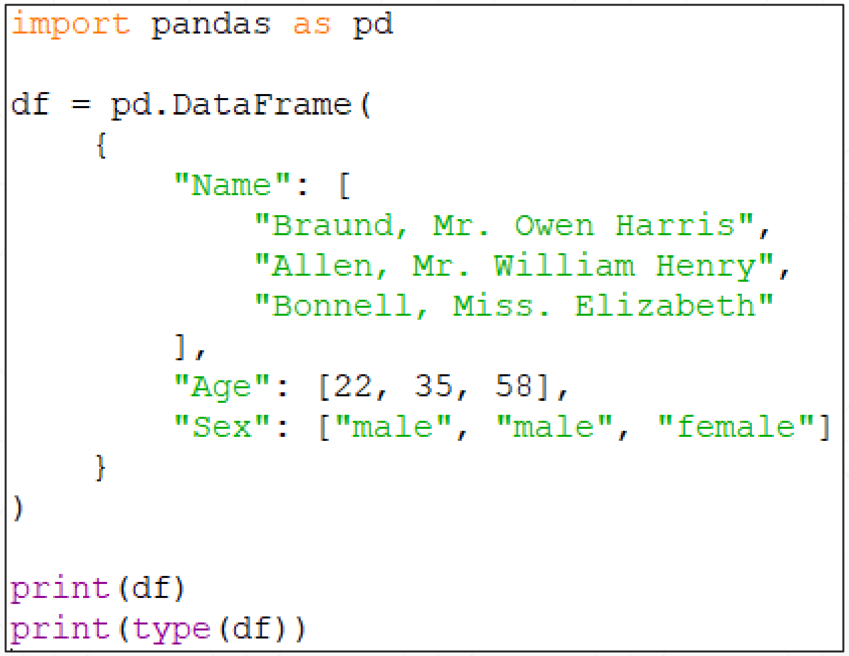
Each column in a DataFrame is a Series



Recap the basic of Pandas

pandas

- **DataFrame Example**



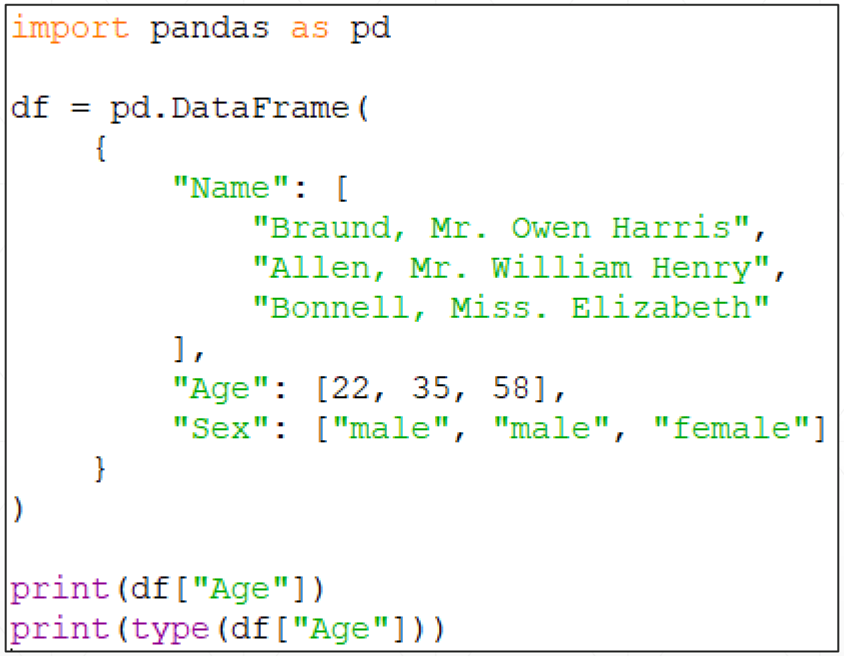
```python
import pandas as pd

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)
print(type(df))
```

- **Explanation**

- Print the DataFrame and show the data type.

Recap the basic of Pandas

- **Series Example**



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

- **Explanation**

- Print the Series (i.e. Age) and show the data type.

Data processing/ cleansing/ analysis with Pandas

Data Processing with Pandas

- **DataFrame Example**

```
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"]
    }
)

#Only non-textual data will be taken account by default
print(df.describe())
#Specify a column to be described
print(df["Sex"].describe())
```

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
```

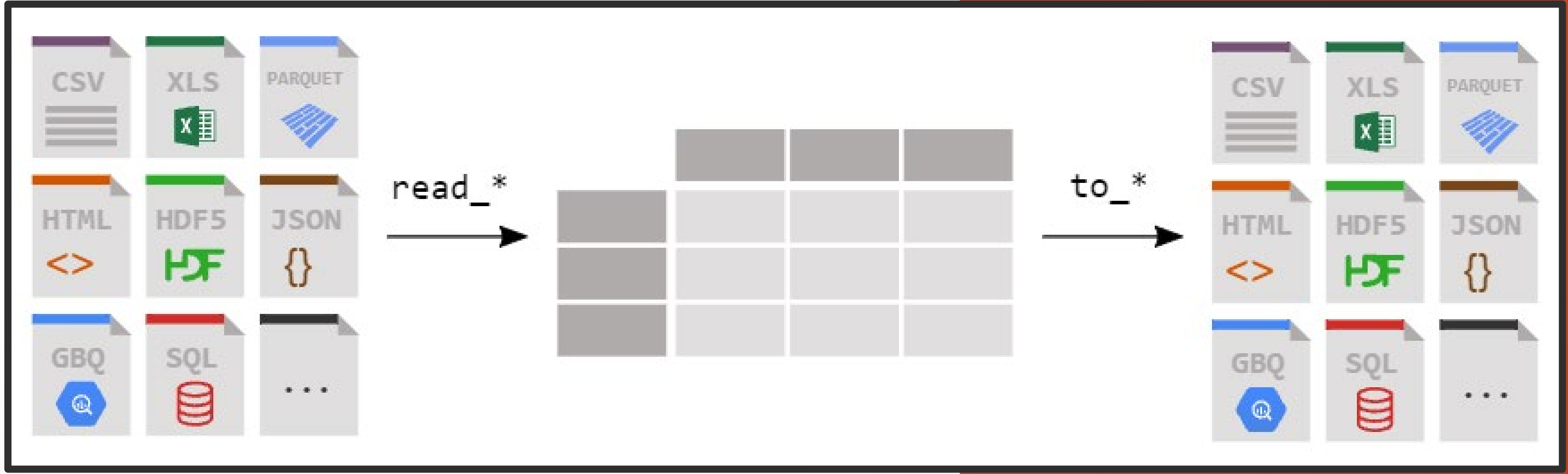
df["Sex"].describe()

```
count         3
unique        2
top        male
freq          2
Name: Sex, dtype: object
```

- **Explanation**
- Show the related statistic of DataFrame

Read/Convert data files for Pandas

CSV, Excel files, etc.



Read/Convert data files

- **Explain the Data Model of the "titanic.csv" example**

- **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.

Read/Convert data files

- **Syntax of reading CSV**

```
pd.read_csv("sample.csv")
```

- **Example**

```
titanic = pd.read_csv("titanic.csv")
```

- **Full Example**

```
import pandas as pd

titanic = pd.read_csv("titanic.csv")

print(titanic)
```

```
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0              1         0       3  ...   7.2500   NaN         S
1              2         1       1  ...  71.2833   C85         C
2              3         1       3  ...   7.9250   NaN         S
3              4         1       1  ...  53.1000  C123         S
4              5         0       3  ...   8.0500   NaN         S
..           ...       ...     ...  ...      ...   ...       ...
886          887         0       2  ...  13.0000   NaN         S
887          888         1       1  ...  30.0000   B42         S
888          889         0       3  ...  23.4500   NaN         S
889          890         1       1  ...  30.0000  C148         C
890          891         0       3  ...   7.7500   NaN         Q

[891 rows x 12 columns]
```

You may use:
`read_excel`
`read_json`
`read_sql`
etc. for different file types

Read/Convert data files

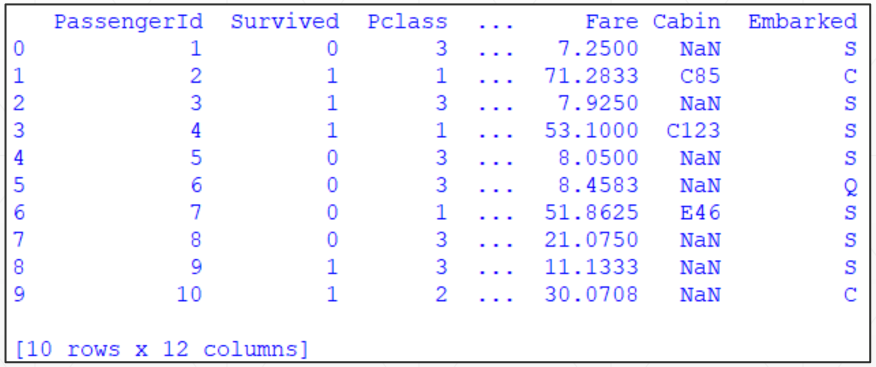
- **Example of using `head()` to show top records**

```
titanic = pd.read_csv("titanic.csv")
print(titanic.head(10))
```

- **Example of using `tail()` to show bottom records**

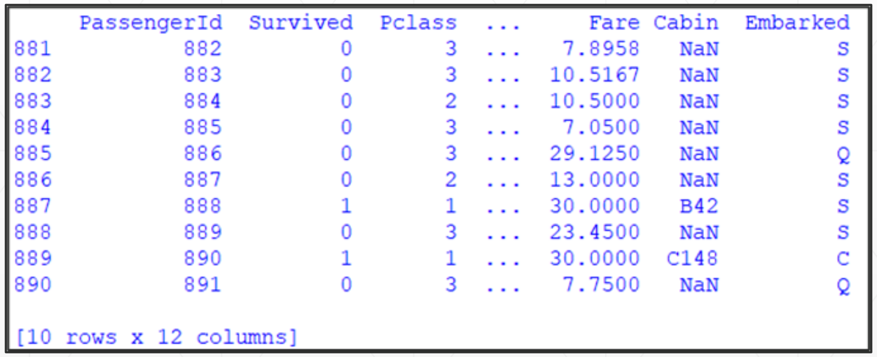
```
titanic = pd.read_csv("titanic.csv")
print(titanic.tail(10))
```

head()



```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S
5            6         0       3  ...   8.4583   NaN         Q
6            7         0       1  ...  51.8625   E46         S
7            8         0       3  ...  21.0750   NaN         S
8            9         1       3  ...  11.1333   NaN         S
9           10         1       2  ...  30.0708   NaN         C

[10 rows x 12 columns]
```

tail()



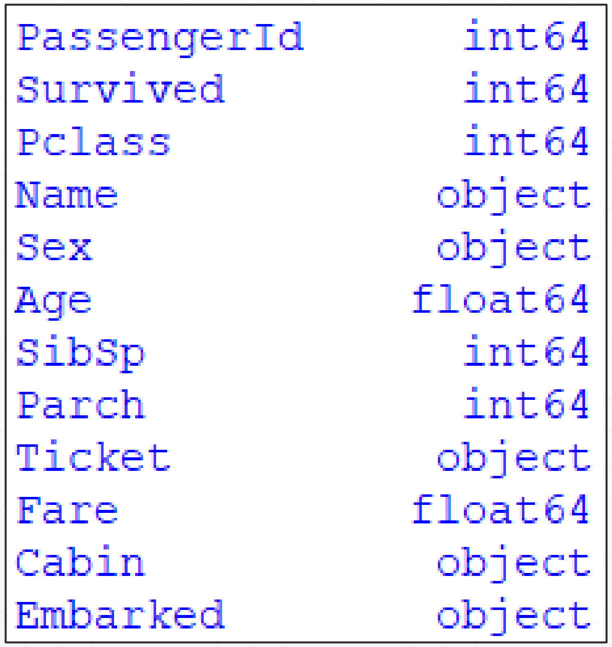
```
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
881          882         0       3  ...   7.8958   NaN         S
882          883         0       3  ...  10.5167   NaN         S
883          884         0       2  ...  10.5000   NaN         S
884          885         0       3  ...   7.0500   NaN         S
885          886         0       3  ...  29.1250   NaN         Q
886          887         0       2  ...  13.0000   NaN         S
887          888         1       1  ...  30.0000   B42         S
888          889         0       3  ...  23.4500   NaN         S
889          890         1       1  ...  30.0000  C148         C
890          891         0       3  ...   7.7500   NaN         Q

[10 rows x 12 columns]
```

Read/Convert data files

- **Example of using `dtypes()` to show the datatypes**

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

dtypes



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

Read/Convert data files

- **Syntax of converting CSV to Excel**

```
pd.to_excel("sample.xlsx", sheet_name="Sample
Sheet", index=False)
```

- **Example**

```
titanic.to_excel("titanic.xlsx",
sheet_name="passengers", index=False)
```

- **Full Example**

If error exists, install **openpyxl** module

```
import pandas as pd

titanic = pd.read_csv("titanic.csv")

titanic.to_excel("titanic.xlsx", sheet_name="passengers", index=False)
```

Go to the folder of the python file and verify if the Excel file exists ☺

Read/Convert data files

- **Syntax of reading Excel**
```
pd.read_excel("sample.xlsx",
sheet_name="Sample Sheet")
```

- **Example (use the Excel created in previous example)**
```
titanic = pd.read_excel("titanic.xlsx",
sheet_name="passengers")
```

If error exists, install **openpyxl** module

- **Full Example (with openpyxl)**

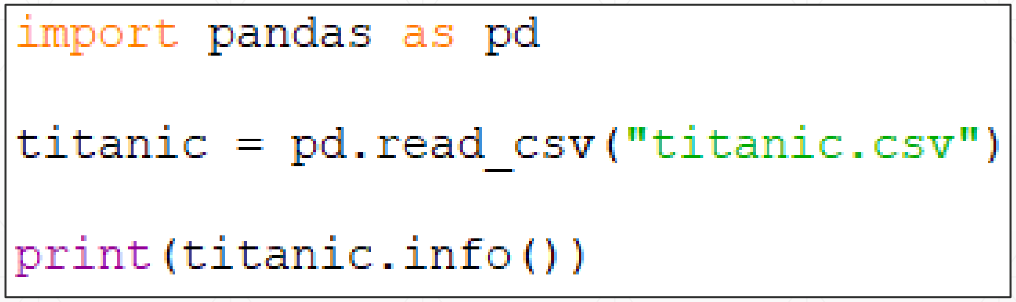
```
import pandas as pd
import openpyxl

#titanic = pd.read_excel("titanic.xlsx", sheet_name="passengers")
titanic = pd.read_excel("titanic.xlsx", sheet_name="passengers", engine="openpyxl")

print(titanic)
```

Read/Convert data files

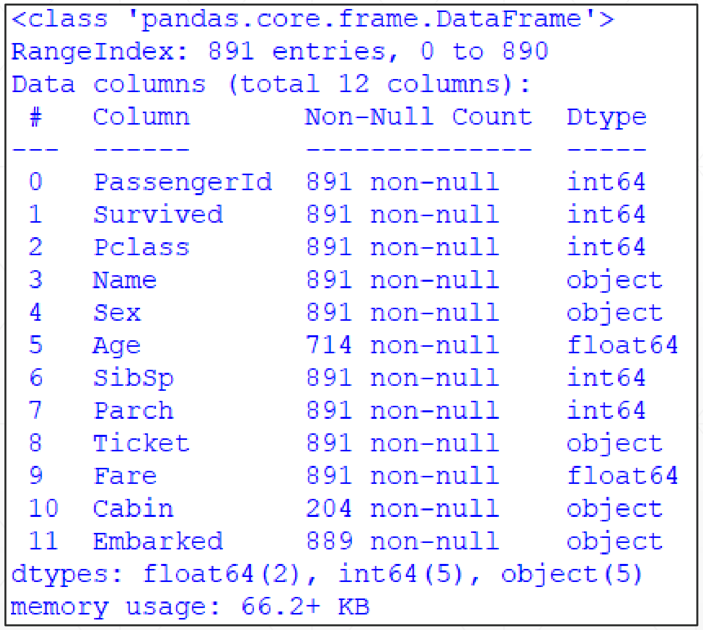
- **Syntax of reading DataFrame info**
```
pd.info()
```

- **Example**
```
titanic = pd.read_csv("titanic.csv")
print(titanic.info())
```

- **Full Example**



```
import pandas as pd

titanic = pd.read_csv("titanic.csv")

print(titanic.info())
```



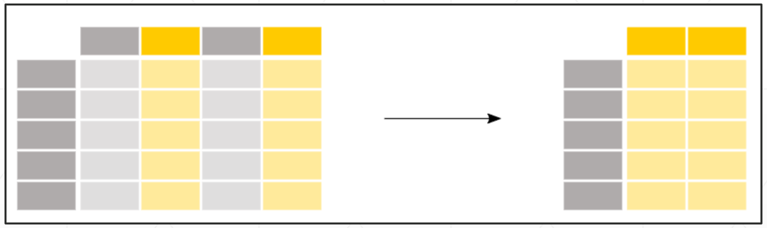
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    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: 66.2+ KB
```

Simple Data Processing

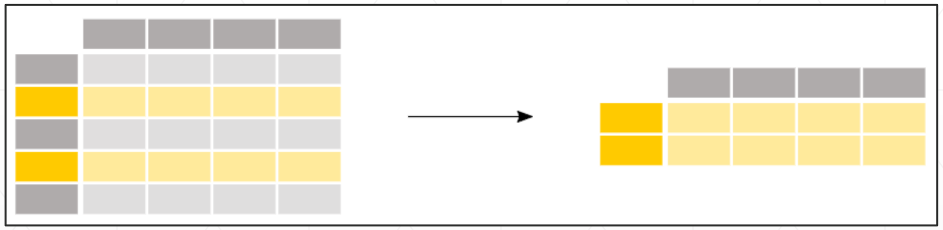
Simple Data Processing

- **<u>Select a subset from DataFrame</u>**
  - Case1: select specific columns from a DataFrame
  - Case2: select specific rows from a DataFrame

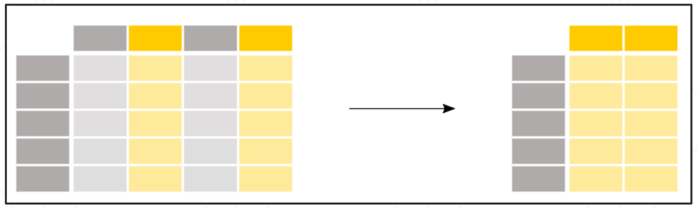
Case1



Case2



Simple Data Processing



- **Select specific columns from a DataFrame**

```
df["column"]
```

- **Example – single column**

```
titanic = pd.read_csv("titanic.csv")
ages = titanic["Age"]
print(ages.head(10))
```

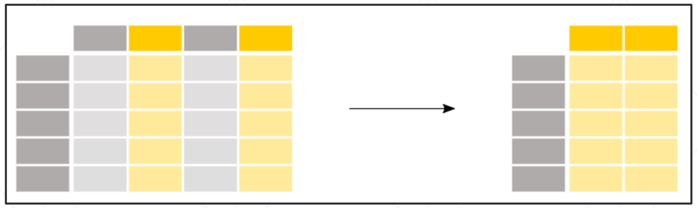
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
Name: Age, dtype: float64
```

- **Example – multiple columns**

```
titanic = pd.read_csv("titanic.csv")
age_sex = titanic[["Age", "Sex"]]
print(age_sex.head(10))
```

```
    Age     Sex
0  22.0    male
1  38.0  female
2  26.0  female
3  35.0  female
4  35.0    male
5   NaN    male
6  54.0    male
7   2.0    male
8  27.0  female
9  14.0  female
```

Simple Data Processing



- **Select specific columns from a DataFrame**
- **Use of `shape`**
  check the (num_row, num_column)
- **Example**

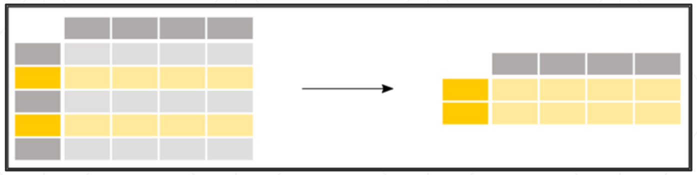
```
titanic = pd.read_csv("titanic.csv")
ages = titanic["Age"]
print(ages.shape)
```

```
(891, )
```

```
age_sex = titanic[["Age", "Sex"]]
print(age_sex.shape)
```

```
(891, 2)
```

Simple Data Processing



- **Select specific rows from a DataFrame**

```
df["column"] > 100
df["column"] == "value"
df["column"] != 50
```

- **Example – single criteria**

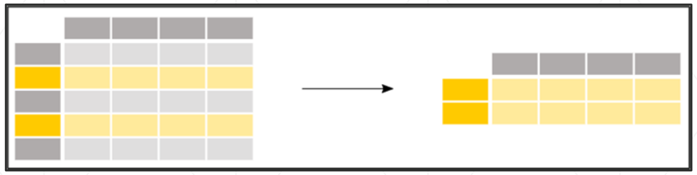
```
titanic = pd.read_csv("titanic.csv")
above_35 = titanic[titanic["Age"] > 35]
print(above_35.head(10))
```

```
    PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
1             2         1       1  ...  71.2833   C85         C
6             7         0       1  ...  51.8625   E46         S
11           12         1       1  ...  26.5500  C103         S
13           14         0       3  ...  31.2750   NaN         S
15           16         1       2  ...  16.0000   NaN         S
25           26         1       3  ...  31.3875   NaN         S
30           31         0       1  ...  27.7208   NaN         C
33           34         0       2  ...  10.5000   NaN         S
35           36         0       1  ...  52.0000   NaN         S
40           41         0       3  ...   9.4750   NaN         S
```

How to display all columns?

```
pd.set_option('display.max_columns', None)
```

Simple Data Processing



- **Select specific rows from a DataFrame**
- **Example – multiple criteria (Use `isin()`)**

```
titanic = pd.read_csv("titanic.csv")
class_23 = titanic[titanic["Pclass"].isin([2, 3])]
print(class_23.head(10))
```

- **Example – multiple criteria (Use |)**

| means or
& means and

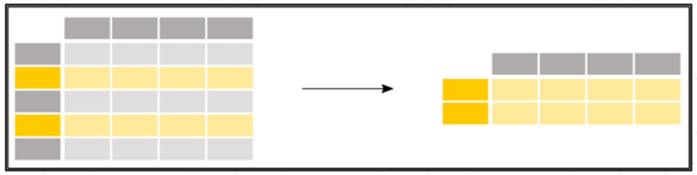
```
titanic = pd.read_csv("titanic.csv")
class_23 = titanic[(titanic["Pclass"] == 2) |
(titanic["Pclass"] == 3)]
print(class_23.head(10))
```

- **Example – select not NA rows (Use `notna()`)**

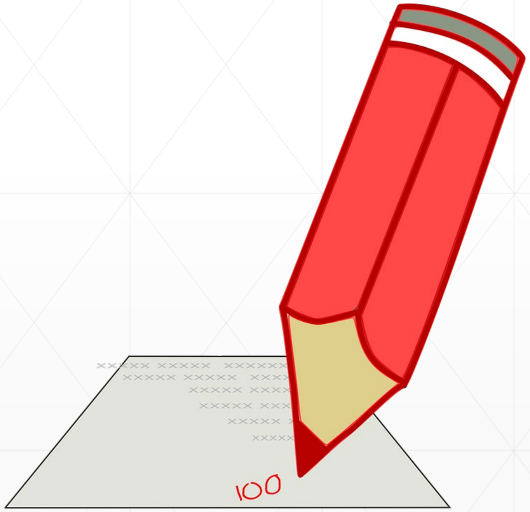
```
titanic = pd.read_csv("titanic.csv")
age_no_na = titanic[titanic["Age"].notna()]
print(age_no_na.head(10))
```

`isna()/isnull()`
finds NaN rows

Simple Data Processing



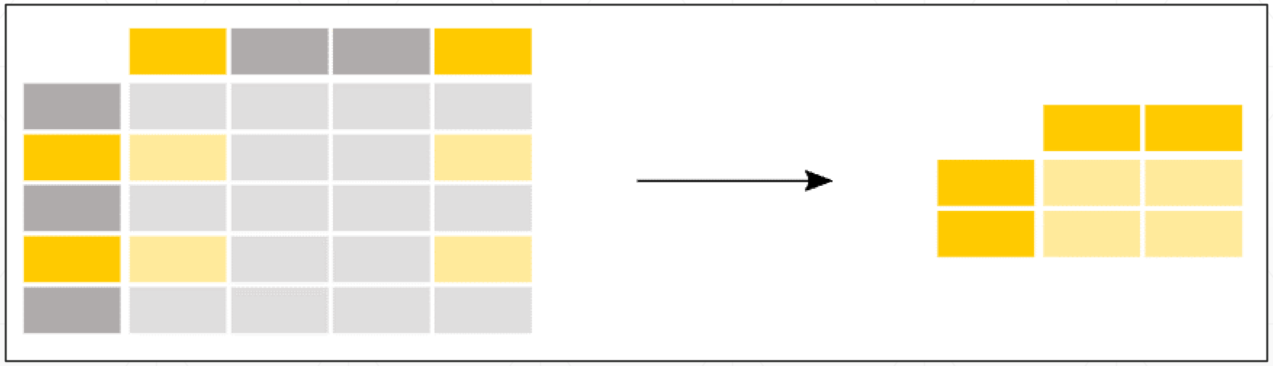
- Select specific rows from a DataFrame
- Try yourself! Find the `shape` of previous examples



Simple Data Processing

- **How about selecting specific columns and rows from a DataFrame?**

Specific columns and rows



Simple Data Processing

- **<u>Select specific columns and rows from a DataFrame Using `loc`</u>**

- Specify with **the row and column names**

```
df.loc["row_name","column_name"]
```

- **<u>Example – using `loc`</u>**

```
titanic = pd.read_csv("titanic.csv")
adult_names = titanic.loc[titanic["Age"] > 35, "Name"]
print(adult_names.head(10))
```

```
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) 
25    Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...
30                             Uruchurtu, Don. Manuel E
33                                Wheadon, Mr. Edward H
35                         Holverson, Mr. Alexander Oskar
40       Ahlin, Mrs. Johan (Johanna Persdotter Larsson)
```

Explanation:
Select the names of the passengers older than 35 years

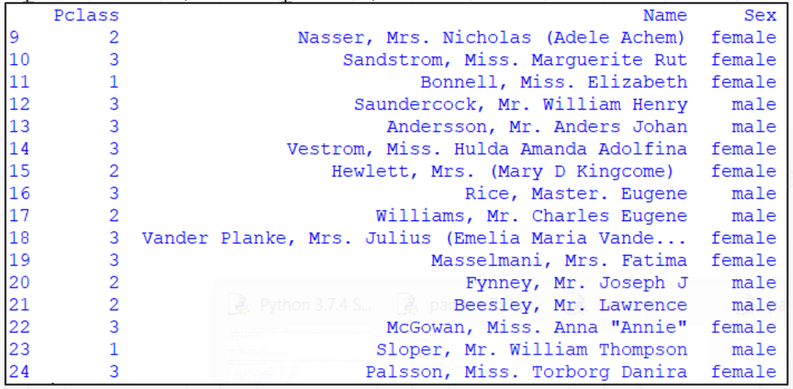
Simple Data Processing

- **<u>Select specific columns and rows from a DataFrame Using `iloc`</u>**
- Specify with **the row and column index**

```
df.iloc["row_index","column_index"]
```

- **<u>Example – using `iloc`</u>**

```
titanic = pd.read_csv("titanic.csv")
sample = titanic.iloc[9:25, 2:5]
print(sample)
```



```
   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
14      3        Vestrom, Miss. Hulda Amanda Adolfina  female
15      2            Hewlett, Mrs. (Mary D Kingcome)   female
16      3                        Rice, Master. Eugene    male
17      2                Williams, Mr. Charles Eugene    male
18      3 Vander Planke, Mrs. Julius (Emelia Maria Vande...  female
19      3                     Masselmani, Mrs. Fatima  female
20      2                        Fynney, Mr. Joseph J    male
21      2                       Beesley, Mr. Lawrence    male
22      3                 McGowan, Miss. Anna "Annie"  female
23      1                Sloper, Mr. William Thompson    male
24      3               Palsson, Miss. Torborg Danira  female
```

Explanation:
Select the rows 10 till 25 and columns 3 to 5 by using index

Simple Data Processing

- **Select specific columns and rows from a DataFrame Using `iloc`**

- Try to assign value to specific columns and rows

- **Example – assign value with `iloc`**

```
titanic = pd.read_csv("titanic.csv")
titanic.iloc[0:5, 3] = "anonymous"
id_names = titanic[["PassengerId","Name"]]
print(id_names.head(10))
```

```
   PassengerId                                               Name
0            1                                          anonymous
1            2                                          anonymous
2            3                                          anonymous
3            4                                          anonymous
4            5                                          anonymous
5            6                                   Moran, Mr. James
6            7                            McCarthy, Mr. Timothy J
7            8                     Palsson, Master. Gosta Leonard
8            9  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9           10                Nasser, Mrs. Nicholas (Adele Achem)
```

Explanation:
Select the rows 1 till 5 and columns “Name” and assign the value

References

- References/Examples of Pandas
  - https://pandas.pydata.org/
- Basic about `describe()` of Pandas
  - https://pandas.pydata.org/docs/user_guide/basics.html#basics-describe
- Troubleshoot for pandas cannot read xlsx file
  - https://stackoverflow.com/questions/65250207/pandas-cannot-open-an-excel-xlsx-file

Thank you