Python for Data Science and Machine Learning

School Year 2023-2024

IST

Course Structure

1	Introduction to Python	OCT 19	OCT 26	NOV 9	NOV 16	NOV 23	NOV 30	DEC 7
2	Python Challenge	DEC 14						
3	Introduction to Data Science	DEC 21	JAN 11	JAN 18	JAN 25			
4	Introduction to Machine Learning	FEB 1	FEB 8	FEB 22	FEB 29			
5	Data Science & ML Challenge	MAR 7						

■ = Core Topics ▨ = Optional Topics

Jupyter Notebook Setup

In a browser:

```
192.168.10.4:8888
```

Password: **ist**

Recap: Pandas

Pandas is a powerful Python data analysis toolkit.

It provides flexible data structures like **Series** and **DataFrame**.

Widely used in data science, finance, and many other fields.

10.0

```
import pandas as pd
```

Recap: Series

A **Series** in Pandas is similar to a **dictionary**.

Each element in a Series has a unique label, which is its index.

```
data = [1, 3, 5, 7, 9]
letters = ["A", "F", "H", "L", "Z"]

series = pd.Series(data, index=letters)
series
```

```
A    1
F    3
H    5
L    7
Z    9
dtype: int64
```

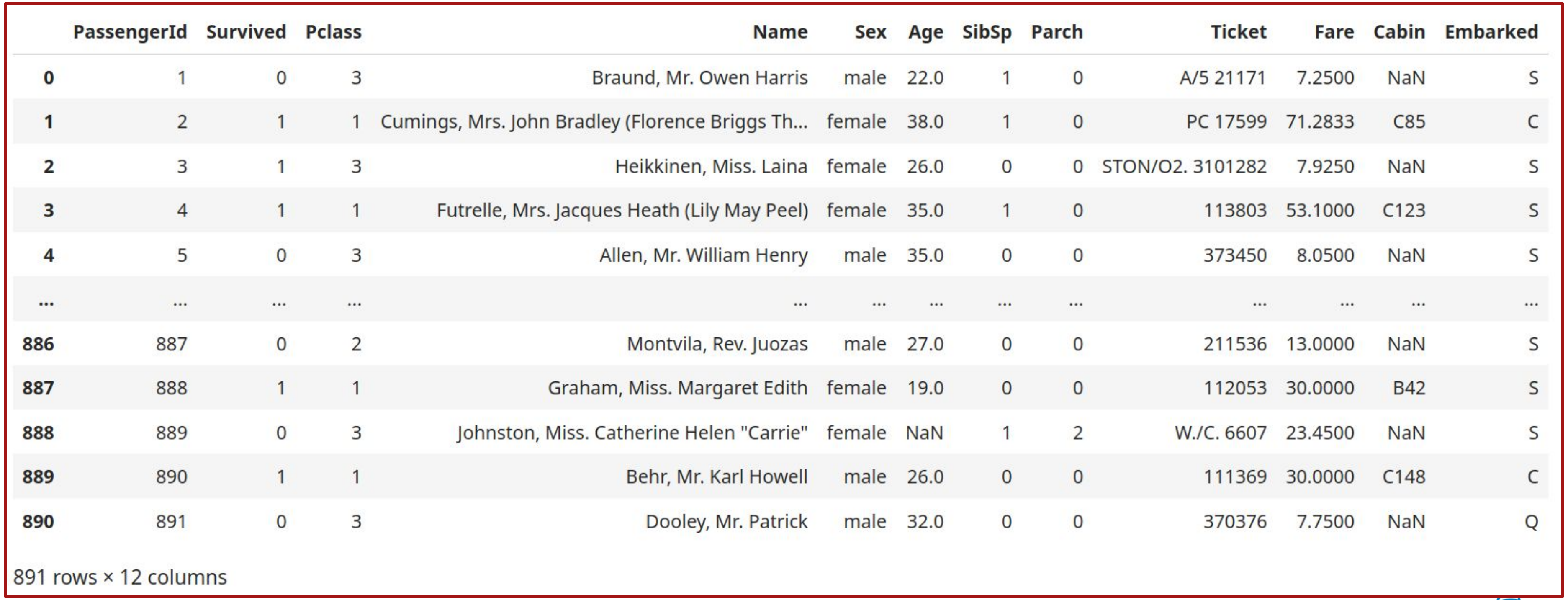
Recap: DataFrame

A **DataFrame** is a two-dimensional data structure with labeled axes (rows and columns).

10.1

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

Recap: DataFrame



	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

891 rows × 12 columns

Recap: Exploring a DataFrame

- Use the **head()** method to display the first 5 rows of the DataFrame **df**.
- Explore what information each column contains.

10.2

```
df.head()
```

10.3

```
df.head(1)
```

Recap: Selecting DataFrame Rows

- The **loc** attribute allows us to select rows and columns by labels.
- **loc** works based on labels of the index.
- To use **loc**, you need to know the index label of the rows and the column names you want to select.

```
df.loc[0]
```

```
df.loc[5:10]
```

Recap: Exercise

Complete the **10.4** , **10.5** & **10.6** programs.

- **10.4**: Use `loc` to select the row at index 3
- **10.5**: Use `loc` to select the rows between index 2 and index 6 (inclusive)
- **10.6**: Use `loc` to select the first 5 rows of dataframe `df`

Recap: Selecting DataFrame Columns

- The **loc** method in Pandas is not only for selecting rows but also for columns.
- By specifying the row and column labels, you can access specific portions of the dataset.

```
df.loc[0, "Name"]
```

```
df.loc[0:4, "Name"]
```

```
df.loc[:4, "Name"]
```

```
df.loc[4, ["Name", "Age"]]
```

```
df.loc[0:4, ["Name", "Age"]]
```

```
df.loc[:, ["Name", "Age"]]
```

Recap Exercise

Complete the **10.7** , **10.8** & **10.9** programs.

- **10.7:** Use `loc` to select the first 10 rows of `df` **and** only include the 'Name' column in your selection.
- **10.8:** Use `loc` to select all rows after index 400 of `df` **and** only include the 'Name' and 'Age' columns in your selection.
- **10.9:** Use `loc` to select all rows of `df` **and** only include the 'Age', 'Fare' and 'Pclass' columns in your selection.

Recap: Boolean Indexing

- **Boolean indexing** in Pandas allows you to select data subsets based on the actual values in the data.
- You can filter the data to match specific criteria.

```
df.loc[:, 'Age']
```

```
df.loc[:, "Age"] > 30
```

10.10

```
df[df.loc[:, "Age"] > 30]
```

10.11

```
df[df.loc[:, 'Pclass'] == 1]
```

Exercise

Complete the **10.12** , **10.13** & **10.14** programs.

- **10.12**: Using boolean indexing, select all passengers who are in `Cabin "G6"`.
- **10.13**: Using boolean indexing, select passengers who paid a fare lower than `$100`
- **10.14**: Using boolean indexing, select passengers who survived the Titanic disaster (`Survived` is 1)

Shorthand!

When using boolean indexing if you wish to **select a column** across **all rows** you can use the following shorthand:

```
df.loc[:, 'Age']
```

$\rightarrow$

```
df['Age']
```

10.15

```
df[df.loc[:, "Age"] > 30]
```

$\rightarrow$

```
df[df["Age"] > 30]
```

Chaining Indexing

You can **chain** multiple boolean indexing operations by using:

- | for “or”
- **&** for “and”

IMPORTANT! You must use **brackets**!

```
df[(df["Pclass"] == 1) | (df["Pclass"] == 2)]
```

10.16

```
df[(df["Pclass"] == 1) & (df["Age"] < 18)]
```

Exercise

Complete the **10.17** , **10.18** & **10.19** programs.

- **10.17:** Using boolean indexing chaining select all the passengers that were either under `18` or over `60` years of age.
- **10.18:** Using boolean indexing chaining find all passengers who embarked from Southampton (`'S'`) and paid a fare less than `$50`.
- **10.19:** Select passengers who are between `20` and `30` years old and paid a fare greater than `$100`.

Quiz Time!

https://ahaslides.com/MWSIB

End of Class

See you all next week!