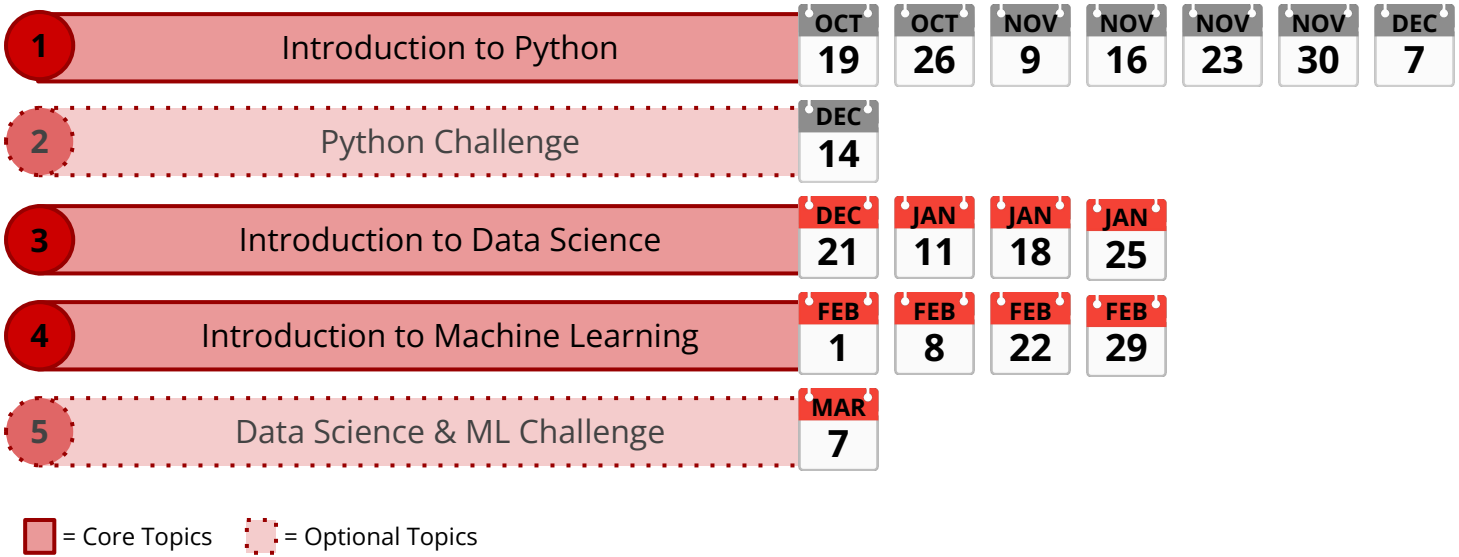
Python for Data Science and Machine Learning

School Year 2023-2024

IST

Course Structure



Jupyter Notebook Setup

In a browser:

```
192.168.10.4:8888
```

Password: **ist**

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.

9.0

```
import pandas as pd
```

Series

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

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

9.1

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

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

DataFrame

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

9.2

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

Exploring a DataFrame

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

9.3

```
df.head()
```

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]
```

Exercise

Complete the **9.4** , **9.5** & **9.6** programs.

- **9.4**: Use `loc` to select the passenger at index 15
- **9.5**: Use `loc` to select the passengers between index 6 and index 9
- **9.6**: Use `loc` to select the last passenger in the dataframe `df`

Solution 9.4

```
df.loc[15]
```

Solution 9.5

```
df.loc[6:9]
```

Solution 9.6

```
df.loc[len(df) - 1]
```

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"]]
```

Exercise

Complete the **9.7** , **9.8** & **9.9** programs.

- **9.7**: Use `loc` to select the first 10 rows of `df`.
- **9.8**: Use `loc` to select the first 10 rows of `df` **and** only include the 'Name' and 'Age' columns in your selection.
- **9.9**: Use `loc` to select all rows of `df` **and** only include the 'Age', 'Fare' and `Pclass` columns in your selection.

Solution 9.7

```
df.loc[:9]
```

Solution 9.8

```
df.loc[:9, ["Name", "Age"]]
```

Solution 9.9

```
df.loc[:, ["Age", "Fare", "Pclass"]]
```

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
```

9.10

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

9.11

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

Exercise

Complete the **9.12** , **9.13** & **9.14** programs.

- **9.12**: Using boolean indexing, select passengers who have no siblings/spouses (`SibSp` is `0`)
- **9.13**: Using boolean indexing, select passengers who paid a fare greater than `$100`
- **9.14**: Using boolean indexing, select passengers who survived the Titanic disaster (`Survived` is 1)

Solution 9.12

```
df[df.loc[:, "SibSp"] == 0]
```

Solution 9.13

```
df[df.loc[:, "Fare"] > 100]
```

Solution 9.14

```
df[df.loc[:, "Survived"] == 1]
```

End of Class

See you all next week!