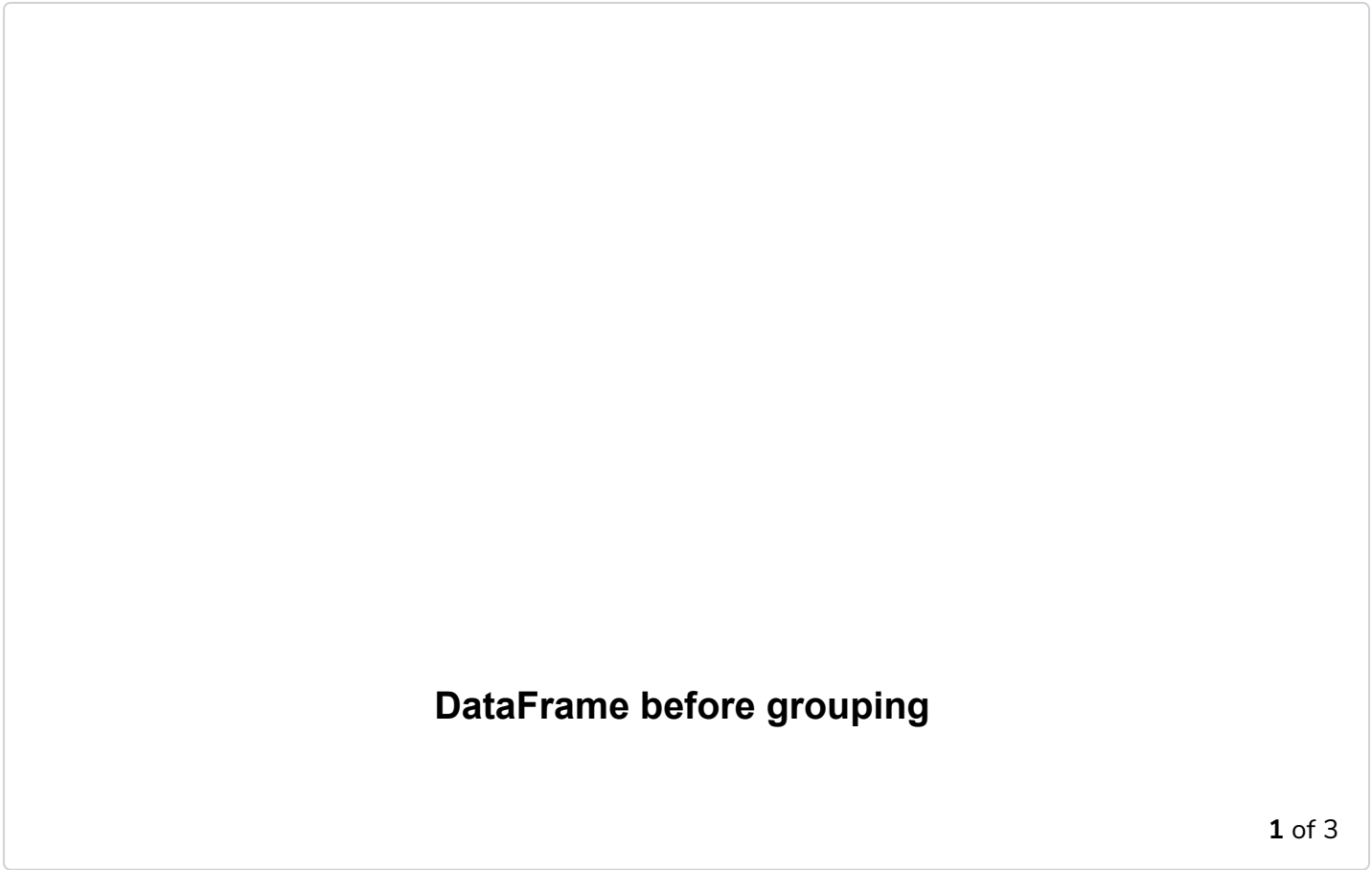
Data Grouping

In this lesson, an explanation is provided on data grouping and its various techniques.

We'll cover the following

Grouping

Grouping arranges similar data of a `DataFrame` in groups. If a value occurs in multiple rows of a single column, the data related to that value in other columns can be grouped together. This method also enables us to perform various *aggregation* operations on the grouped data like *sum*, *avg*, *mean*, etc.



DataFrame before grouping

DataFrame after grouping with column Col1

2 of 3

DataFrame after grouping with column Col2

3 of 3

In the above illustration, the two columns are grouped, and the rows with common values are combined. This helps in obtaining the results of each individual group.

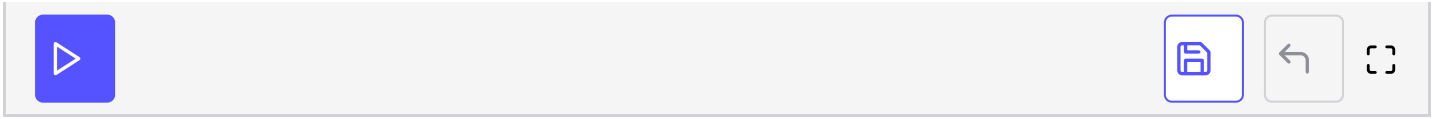
Example

The `groupby` function is used for this task. The values of the `DataFrame` can be grouped on both single and multiple columns. Let's look at an example of grouping some data.

```
import numpy as np
import pandas as pd

df = pd.DataFrame({'p1':['A','A','B','B','C','C'],'p2':['G1','G2','G1','G2','G1','G2'],
    'val_1':np.random.randn(6),'val_2':np.random.randn(6)})


print("DataFrame after using groupby")
print(df.groupby('p1'))
```



It can be seen from the output that instead of printing a grouped `DataFrame`, a grouped by object instance is returned and "displayed".

For a valid-grouped `DataFrame` to be returned, an operation such as *summation* or *averaging* needs to be performed on the grouped data. Let's assume that we want to sum the data based on the groups of `p1` and `p2` from the above example.

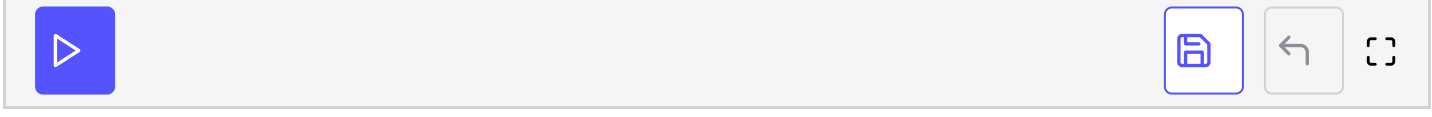
```
import numpy as np
import pandas as pd

df = pd.DataFrame({'p1':['A','A','B','B','C','C'],'p2':['G1','G2','G1','G2','G1','G2'],
    'val_1':np.arange(1,7,1),'val_2':np.arange(7,13,1)})

print("The original DataFrame")
print(df, '\n*****************************************')

print("DataFrame after using groupby on p1 and summing the values")
print(df.groupby('p1').sum(), '\n*****************************************')

print("DataFrame after using groupby on p2 and summing the values")
print(df.groupby('p2').sum(), '\n*****************************************')
```



The `.sum()` function is called in succession to the `groupby()` clause to sum the values of each newly created group.

On **line 11**, the `p1` column is grouped. As a result, the output shows that three groups of `A`, `B`, and `C` are created, and all their respective values are summed for

both `val_1` and `val_2` columns.

On **line 14**, the `p2` column is grouped. As a result, the output shows that two groups of `G1` and `G2` were created, and all their respective values are summed for both `val_1` and `val_2` columns.

Grouping on multiple columns

The same `groupby()` clause can be used to group data based on multiple columns too. All the rules are the same as before, except instead of passing one column to the function, a list of columns, on which data should be grouped, is passed. Let's look at an example.

```
import numpy as np
import pandas as pd

df = pd.DataFrame({'p1':['A','A','B','B','C','C'],'p2':['G1','G2','G1','G2','G1','G2'],
    'val_1':np.arange(1,7,1),'val_2':np.arange(7,13,1)})

print("The original DataFrame")
print(df, '\n*****************************************')

print("DataFrame after using groupby on p1 & p2 and Summing their values")
print(df.groupby(['p1','p2']).sum(), '\n*****************************************')
```

The revised statement with two columns is on **line 11**. It passes `p1` and `p2` as a list to the `groupby()` function. Now, in the output, which element belongs to which group can be seen clearly. Keep in mind that for values to be summed, multiple occurrences of a `(p1,p2)` pair need to be present in the `DataFrame`. As all `(p1, p2)` pairs only occur once, no value is summed; instead the values are arranged in a multi-indexed form.

For more information on the group by function and advanced features, refer here.

In the next lesson, data aggregation is discussed.