


Method - Rank():

Rank method is used to rank a series(column) by it's value in ascending or descending order.

1. Create a dataframe :


```
[80] 1 my_df
```



	column1	column2	column3
0	x	1	20
1	y	3	55
2	y	3	12
3	x	1	35
4	x	1	40

2. Making ranks from column3 in ascending order :

```
[82] 1 my_df['column3_rank'] = my_df.column3.rank()  
    2 my_df
```



	column1	column2	column3	column3_rank
0	x	1	20	2.0
1	y	3	55	5.0
2	y	3	12	1.0
3	x	1	35	3.0
4	x	1	40	4.0

(Since parameter ascending isn't specified it's default value is used which is true. Thus the smallest value of column3 is given rank1 and so on)

3. Making ranks from column3 in descending order :

```
[83] 1 my_df['column3_rank'] = my_df.column3.rank(ascending=False)
      2 my_df
```

↗

	column1	column2	column3	column3_rank
0	x	1	20	4.0
1	y	3	55	1.0
2	y	3	12	5.0
3	x	1	35	3.0
4	x	1	40	2.0

By using the **parameter 'ascending' = False**, we can use ranks in descending order. Thus greatest value of column3 (55) is given rank1

4. How to handle duplicates in ranks?

Parameter method is designed for this purpose. (Visit [this page](#) for more details)

- Default value of **'method' = 'average'** (will take the average rank of the group)

```
[88] 1 my_df['column2_rank'] = my_df.column2.rank(ascending=False)
      2 my_df
```

↗

	column1	column2	column3	column2_rank
0	x	1	20	4.0
1	y	3	55	1.5
2	y	3	12	1.5
3	x	1	35	4.0
4	x	1	40	4.0

- Using **'method' = 'first'** (first occurrence will get higher rank)

```
[89] 1 my_df['column2_rank'] = my_df.column2.rank(ascending=False, method = 'first')
      2 my_df
```

↗

	column1	column2	column3	column2_rank
0	x	1	20	3.0
1	y	3	55	1.0
2	y	3	12	2.0
3	x	1	35	4.0
4	x	1	40	5.0

- c. There are three more methods, try them on your own.

VIDEO TO WATCH :

<https://www.youtube.com/watch?v=SeFu3Noexzs&feature=youtu.be>