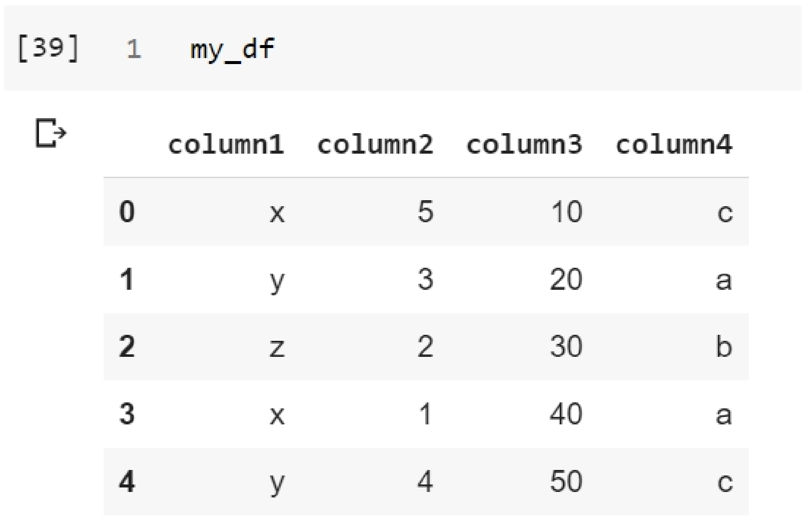
<u>Method 1 : Drop()</u>
This method is used to drop columns/rows by names or indices(locations)

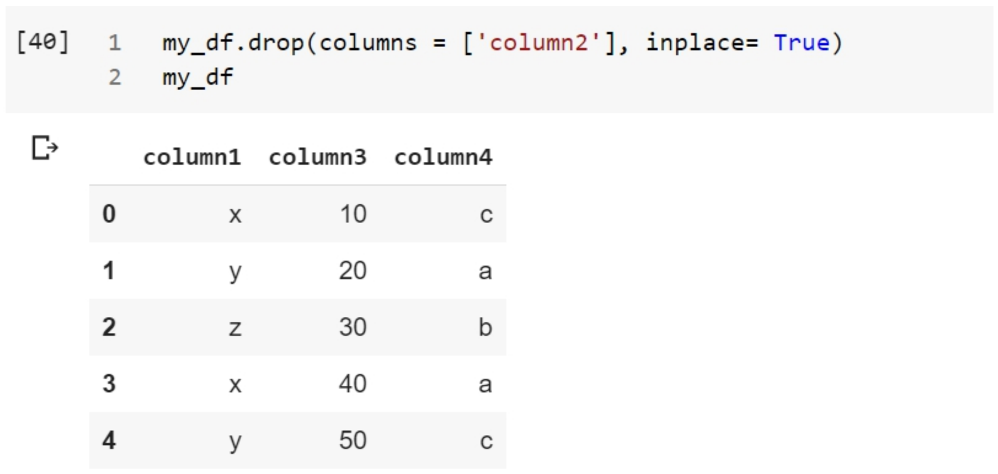
A. **Dropping a column from a dataframe** :
   1. Create a dataframe



```
[39]  my_df
```

	column1	column2	column3	column4
0	x	5	10	c
1	y	3	20	a
2	z	2	30	b
3	x	1	40	a
4	y	4	50	c

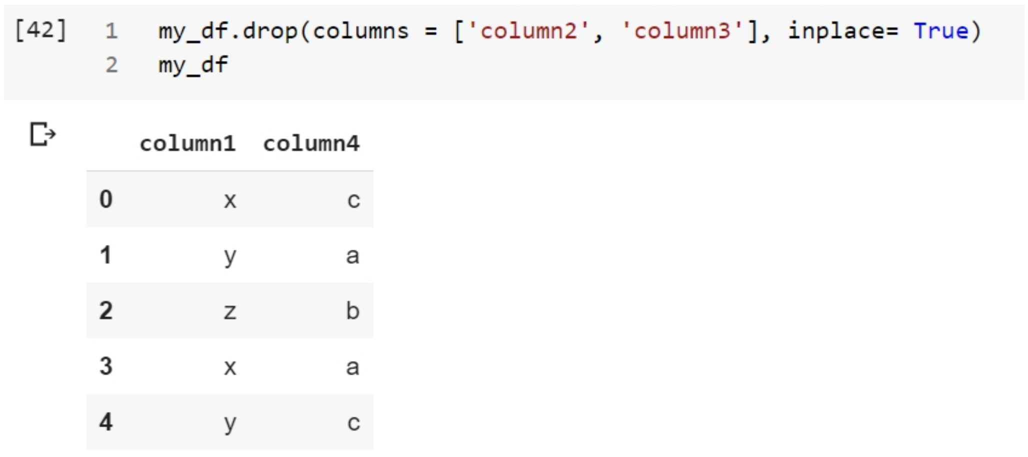
   2. Use the function drop() and pass the list of columns to the **parameter 'columns'**
      i. **Dropping a single column** :



```
[40]  my_df.drop(columns = ['column2'], inplace= True)
      my_df
```

	column1	column3	column4
0	x	10	c
1	y	20	a
2	z	30	b
3	x	40	a
4	y	50	c

ii. **Dropping multiple columns** :

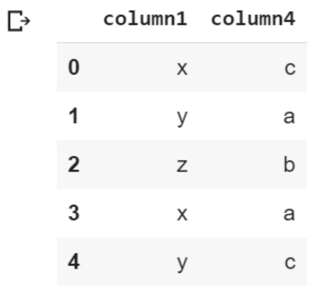


```
[42]
my_df.drop(columns = ['column2', 'column3'], inplace= True)
my_df
```

```
   column1  column4
0        x        c
1        y        a
2        z        b
3        x        a
4        y        c
```

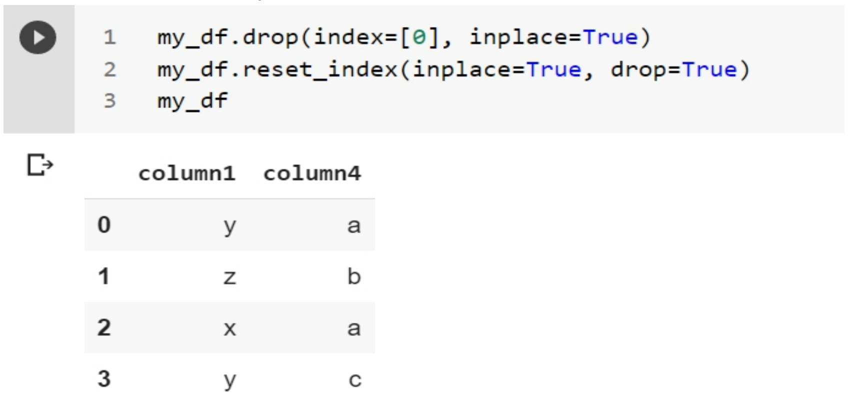
B. Dropping rows from a dataframe :

a. Create a dataframe



```
   column1  column4
0        x        c
1        y        a
2        z        b
3        x        a
4        y        c
```

b. To drop rows by their indices, use the **parameter ‘index**’ and pass a list of indices which rows you want to drop



```
my_df.drop(index=[0], inplace=True)
my_df.reset_index(inplace=True, drop=True)
my_df
```

```
   column1  column4
0        y        a
1        z        b
2        x        a
3        y        c
```

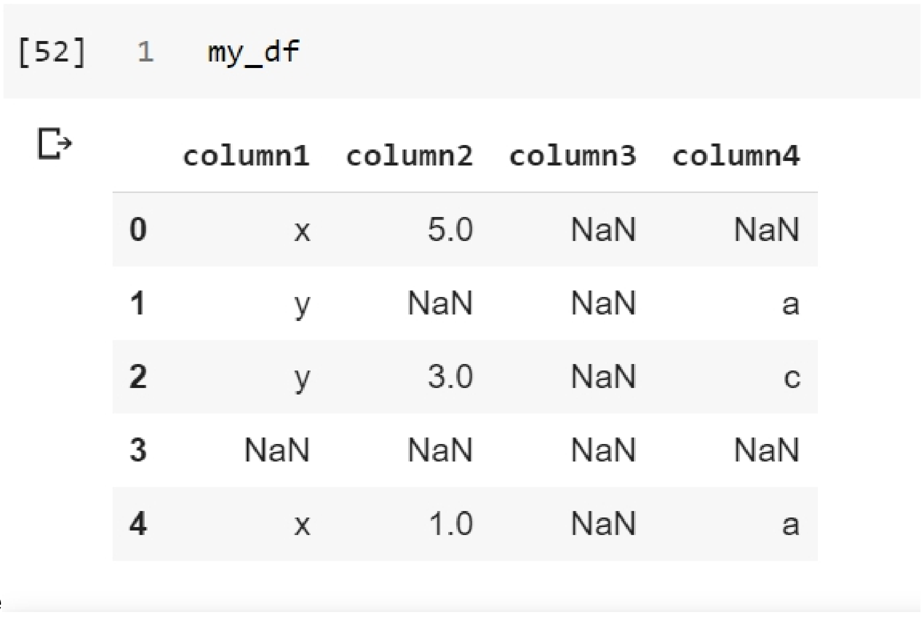
(I have only passed a single value [0] but you can pass a list containing multiple indices)

<u>Method 2 : Dropna()</u>

This method is used to delete rows/columns which have null (missing) values.

A. **Drop a row/column with 100% null values** :

1. Create a

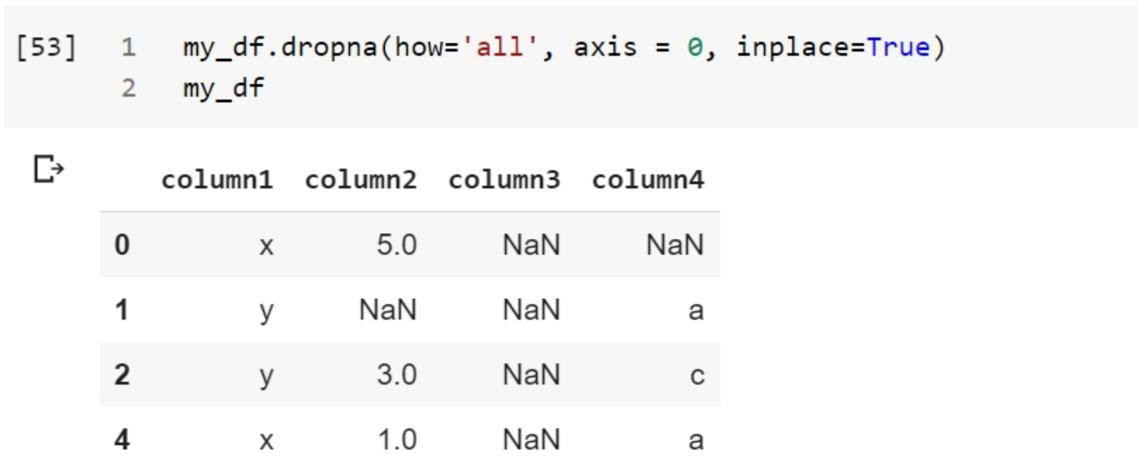


```
[52]  my_df
```

	column1	column2	column3	column4
0	x	5.0	NaN	NaN
1	y	NaN	NaN	a
2	y	3.0	NaN	c
3	NaN	NaN	NaN	NaN
4	x	1.0	NaN	a

dataframe
(As you can see the 3rd column is fully null and the 4th row is fully null. Thus our goal is to remove them from our dataframe)

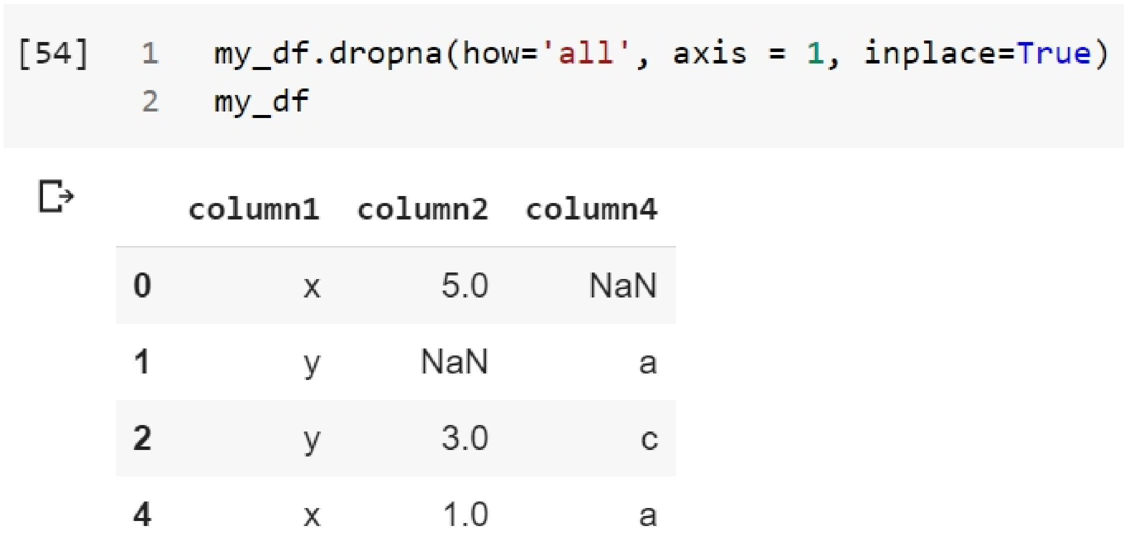
2. For dropping we'll use the **parameter 'how'** and specify its value as 'all'. This means only rows/columns with all values as null will be dropped. For specifying whether we want to drop the columns or rows, we'll use the **parameter 'axis'.** For **dropping rows we'll use axis=0**, and for **dropping columns, axis=1**
   i. Dropping fully null
   rows :



```
[53]  my_df.dropna(how='all', axis = 0, inplace=True)
      my_df
```

	column1	column2	column3	column4
0	x	5.0	NaN	NaN
1	y	NaN	NaN	a
2	y	3.0	NaN	c
4	x	1.0	NaN	a

ii. Dropping fully null columns :

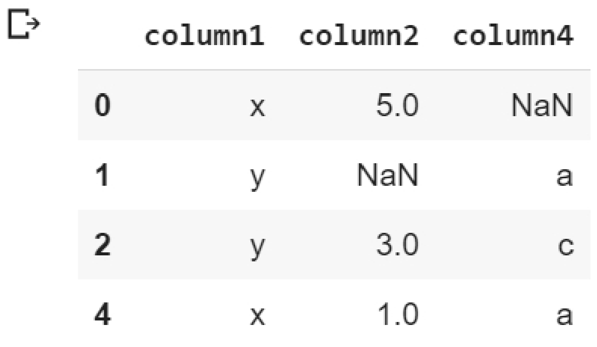


```
[54] my_df.dropna(how='all', axis = 1, inplace=True)
     my_df
```

	column1	column2	column4
0	x	5.0	NaN
1	y	NaN	a
2	y	3.0	c
4	x	1.0	a

B. Dropping a row/column if they contain any nulls :

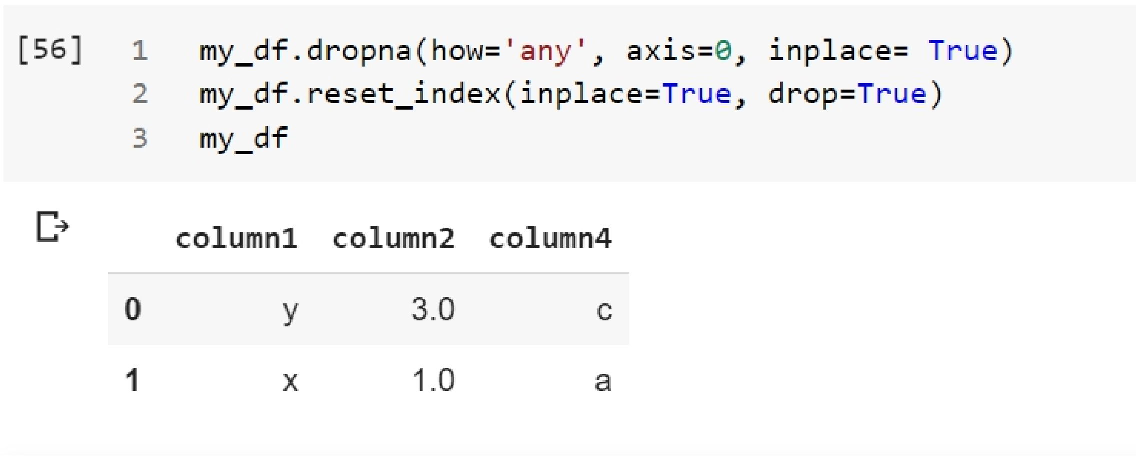
1. Create a dataframe



	column1	column2	column4
0	x	5.0	NaN
1	y	NaN	a
2	y	3.0	c
4	x	1.0	a

2. For dropping a row/column with any null values we'll specify the parameter **'how' = 'any'**. As we did previously, **for specifying rows,axis=0**, and **for columns, axis=1**.

   i. **Dropping rows with any null values** :

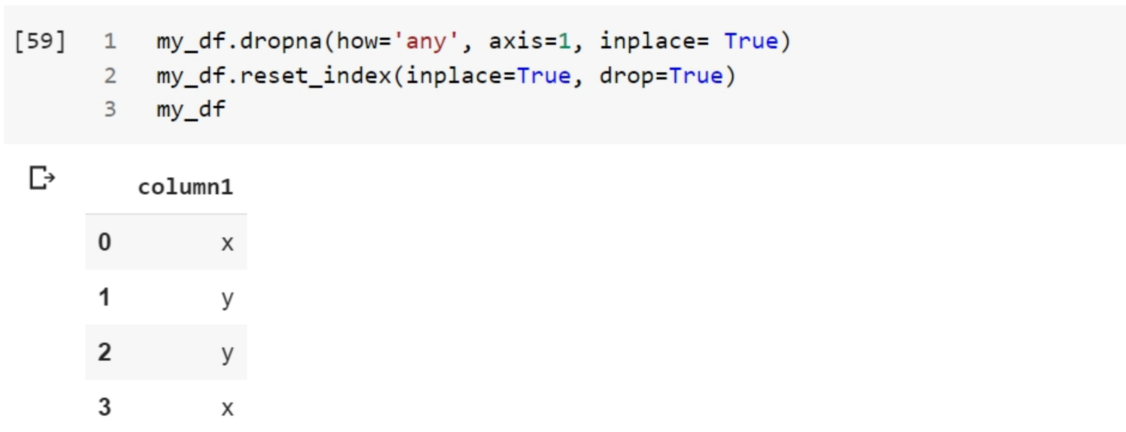


```
[56]  my_df.dropna(how='any', axis=0, inplace= True)
      my_df.reset_index(inplace=True, drop=True)
      my_df
```

```
     column1  column2  column4
0          y      3.0        c
1          x      1.0        a
```

(the first and second rows are dropped as they had nulls)

ii. **Dropping columns with any null values** :



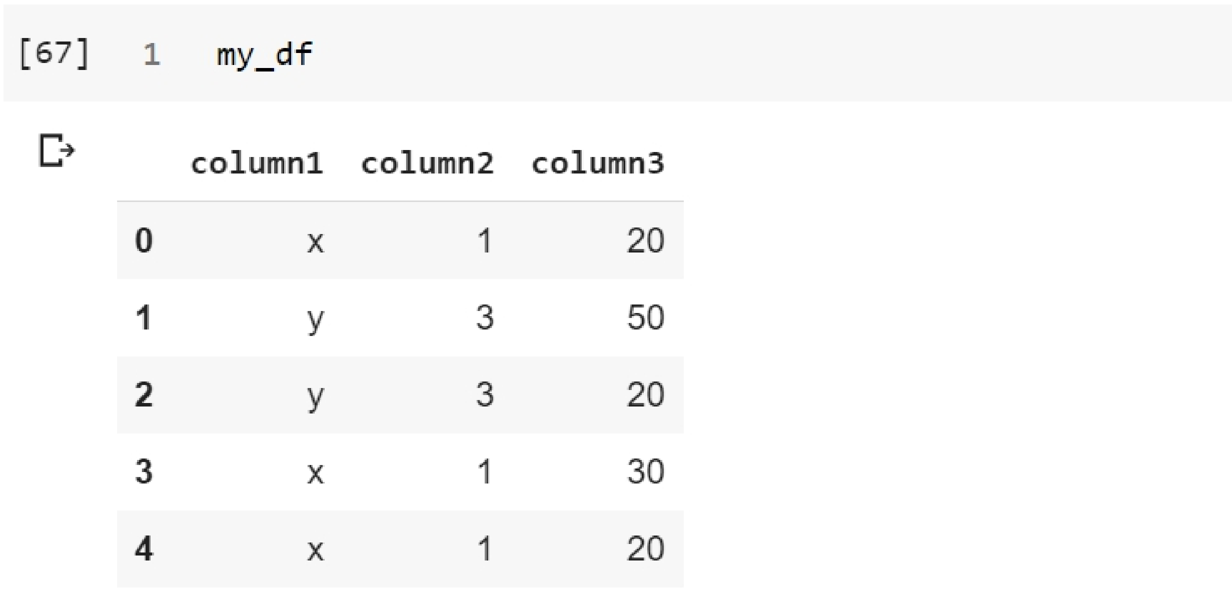
```
[59]  my_df.dropna(how='any', axis=1, inplace= True)
      my_df.reset_index(inplace=True, drop=True)
      my_df
```

```
   column1
0        x
1        y
2        y
3        x
```

(I used this code on the original dataframe, shown in step 1. Column2 and column4 are dropped as they had null values)

<u>Method 3 : Drop duplicates()</u>
This method is used to drop rows which are duplicate.

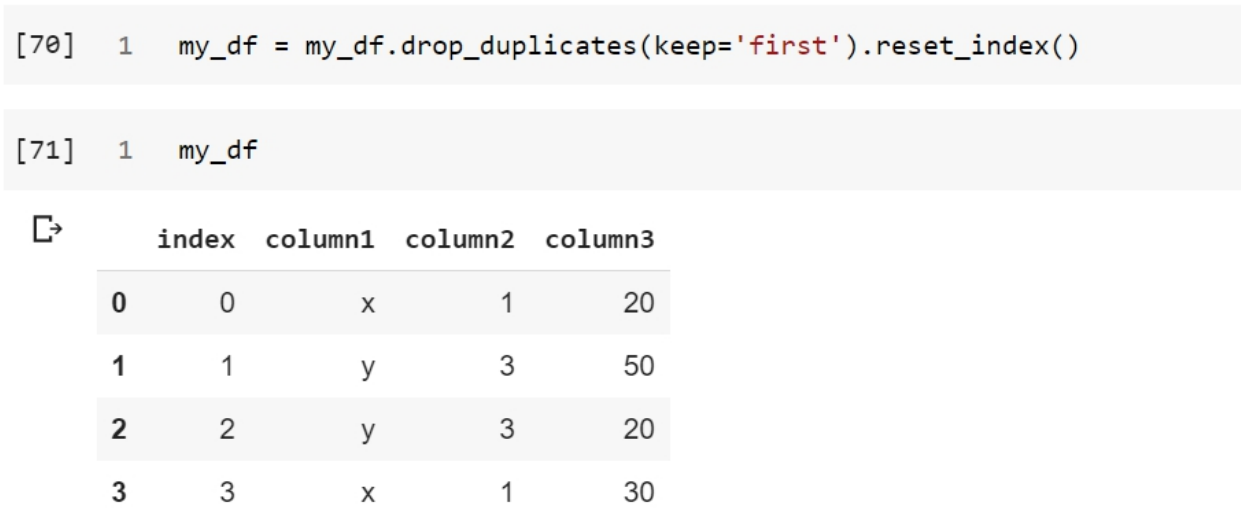
A. **To drop rows which are duplicates based on all columns** (all columns of the rows have same value) :
   a. Create a
      dataframe :



```
[67]   my_df
```

```
    column1  column2  column3
0         x        1       20
1         y        3       50
2         y        3       20
3         x        1       30
4         x        1       20
```

   b. Dropping :



```
[70]   my_df = my_df.drop_duplicates(keep='first').reset_index()
```

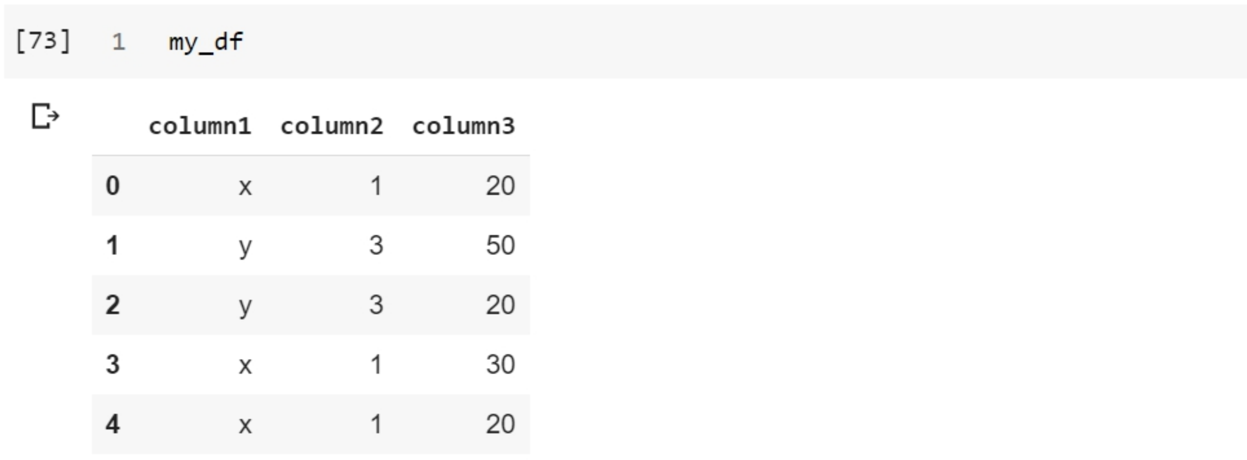
```
[71]   my_df
```

```
   index  column1  column2  column3
0      0        x        1       20
1      1        y        3       50
2      2        y        3       20
3      3        x        1       30
```

(the first and the last rows were duplicates. Since we used **parameter ‘keep’ = ‘first’**, only the first occurrence of the duplicates was kept. By specifying’ **keep’ = ‘last**’, we can save the last occurrence. Additionally, by using **‘keep’ = False,** we can drop both occurrences)

B. **To drop rows which are duplicates based on 1 or multiple columns** of our choosing:

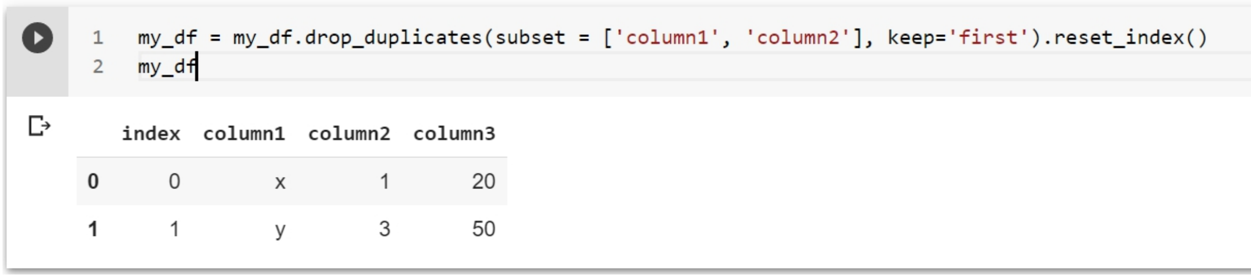
a. Create a dataframe :



```
[73]  my_df
```

```
   column1  column2  column3
0        x        1       20
1        y        3       50
2        y        3       20
3        x        1       30
4        x        1       20
```

b. We will use a **parameter ‘subset**’. Subset contains a list of columns based on which duplicate rows are to be found. Parameter ‘keep’ will also be used as specified in the previous point.



```python
my_df = my_df.drop_duplicates(subset = ['column1', 'column2'], keep='first').reset_index()
my_df
```

```
   index  column1  column2  column3
0      0        x        1       20
1      1        y        3       50
```

(We have provided subset as column1 and column2, duplicates will be found based on these two columns. Rows - 0, 3 and 4 were found to be duplicates so 0 was kept as keep was specified as first. Rows 1 and 2 were duplicates so row 1 was kept.)

EXTRA RESOURCES :

1. https://www.w3resource.com/pandas/dataframe/dataframe-drop.php
2. https://hackersandslackers.com/pandas-dataframe-drop/
3. https://www.geeksforgeeks.org/python-pandas-dataframe-dropna/
4. https://www.w3resource.com/pandas/series/series-drop_duplicates.php