DataFrame.rename() function

- It is used to rename the single column name, multiple columns and all columns by index position, with a list, with a dict etc.

Syntax:

> **pandas.DataFrame.rename(index, columns, axis, inplace)**

Paramerters:

1. index – dictionary or function to rename index.
2. columns – dictionary or function to rename column.
3. axis – Value can be either 0 or index | 1 or columns. Default set to '0'.
4. inplace – Used to specify the the DataFrame refered to be update. Default is False. When used True, copy property will be ignored.

> i.e. By default returns DataFrame after updating columns. When use inplace=True it updates the existing DataFrame inplace (self) and donot create the new Dataframe with updates.

Let's create a DataFrame with a dictionary of lists.

Our pandas DataFrame contains column names Courses, Fee, Duration.

```
import pandas as pd
technologies = ({
  'Courses':["Spark","PySpark","Hadoop","Python","pandas","Oracle","Java"],
  'Fee' :[20000,25000,26000,22000,24000,21000,22000],
  'Duration':['30day', '40days' ,'35days', '40days', '60days', '50days', '55days']
              })
df = pd.DataFrame(technologies)

print(df)
print(df.columns)
```

```
   Courses    Fee Duration
0    Spark  20000    30day
1  PySpark  25000   40days
2   Hadoop  26000   35days
3   Python  22000   40days
4   pandas  24000   60days
5   Oracle  21000   50days
6     Java  22000   55days
Index(['Courses', 'Fee', 'Duration'], dtype='object')
```

Rename Single Columns

- Pandas DataFrame.rename() accepts a dict(dictionary) as a parameter for columns you wanted to rename.
- So you just pass a dict with key-value pair; the key is an existing column you would like to rename and value would be your preferred column name.

```
# Rename a Single Column
a=df.rename(columns = {'Courses':'Courses_List'})

print(a.columns)
```

```
Index(['Courses_List', 'Fee', 'Duration'], dtype='object')
```

Yields below output. As you see it rename column from Courses to Courses_List.

Alternatively, you can also write the above statement by using axis=1 or axis='columns'.

```
# Alternatively you can write above using axis
a = df.rename({'Courses':'Courses_Lst'}, axis=0)

#df2=df.rename({'Courses':'Courses_Lst'}, axis='columns')

print(a.columns)
```

```
Index(['Courses', 'Fee', 'Duration'], dtype='object')
```

In order to change columns on the existing DataFrame without copying to the new DataFrame, you have to use inplace=True.

Rename Multiple Columns

You can also use the same approach to rename multiple columns of Pandas DataFrame.

All you need to specify multiple columns you wanted to rename in a dictionary mapping.

```
# Rename multiple columns
a = df.rename(columns = {'Courses':'Courses_List','Fee':'Courses_Fee','Duration':'Courses_Duration'})

print(a.columns)
```

```
Index(['Courses_List', 'Courses_Fee', 'Courses_Duration'], dtype='object')
```

As you see it renames multiple columns.

Additional Points:

Rename Column by index or position

- Rename column by Index/position can by done by using df.columns.values[index]='value' in pandas DataFrame.
- Index and position can be used interchangingly to access column at a given position. By using this you can raname first column, last column e.t.c.
- As you have seen above df.columns returns a column names as a Series and df.columns.values gets column names as list, now you can set the specific index/position with a new value.
- The below example updates column Courses to Courses_Duration at index 3. Note that index starts from zero.

```
# pandas rename column by index
df.columns.values[2] = "Courses_Duration"
```

Rename Columns with List

*Python list can be used to rename all columns in pandas DataFrame, when you doesn't want to rename any specific column then use the same name in the list.

```
#Rename columns wiht list
column_names = ['Courses','Fee','Duration']

df.columns = column_names

print(df.columns)
```

```
    Index(['Courses', 'Fee', 'Duration'], dtype='object')
```

Rename Columns in place

- By defaults, rename() function returns a new DataFrame after updating the column names.
- You can change this behaviour by using inplace=True prameter.
- i.e. By default rename() returns DataFrame after updating columns. When use inplace=True it updates the existing DataFrame inplace (self) and donot create the new Dataframe with updates.

```
# Rename multiple columns
df.rename(columns = {'Courses':'Courses_List','Fee':'Courses_Fee',
   'Duration':'Courses_Duration'}, inplace = True)

print(df.columns)
```

```
    Index(['Courses_List', 'Courses_Fee', 'Courses_Duration'], dtype='object')
```

This renames column names on DataFrame in place and returns None type.

Rename All Columns by adding Suffix or Prefix

- Sometimes you may need to add a string text to the suffix or prefix of all column names.
- You can use pandas.DataFrame.add_prefix() and pandas.DataFrame.add_suffix() to add prefix and suffix respectively to the pandas DataFrame column names.

```
# Add prefix to the column names
df2=df.add_prefix('col_')
print(df2.columns)

# Add suffix to the column names
df2=df.add_suffix('_col')
print(df2.columns)
```

```
    Index(['col_col_Courses', 'col_col_Fee', 'col_col_Duration'], dtype='object')
    Index(['col_Courses_col', 'col_Fee_col', 'col_Duration_col'], dtype='object')
```

Rename or Convert All Columns to Lower or Upper Case

When column names are mixed with lower and upper case, it would be best practice to convert/update all columns names to either lower or upper case.

```
# Change to all lower case
df = pd.DataFrame(technologies)
df2=df.rename(str.lower, axis='columns')
```

```
print(df2.columns)

# Change to all upper case
df = pd.DataFrame(technologies)
df2=df.rename(str.upper, axis='columns')
print(df2.columns)
```

```
Index(['courses', 'fee', 'duration'], dtype='object')
Index(['COURSES', 'FEE', 'DURATION'], dtype='object')
```

Colab paid products - Cancel contracts here