In [17]:

```
import pandas as pd

df = pd.DataFrame({'Name':['Bhaskar','Bhaskar','Bhaskar','Zuhaire','Zuhaire','Zuhai
                   'UT':[1,2,3,1,2,3,1,2,3,1,2,3],
                   'Maths':[22,21,14,20,23,22,23,24,12,15,18,17],
                   'Science':[21,20,19,17,15,18,19,22,25,22,21,18],
                   'S.St':[18,17,15,22,21,19,20,24,19,25,25,20],
                   'Hindi':[20,22,24,24,25,23,15,17,21,22,24,25],
                   'Eng':[21,24,23,19,15,13,22,21,23,22,23,20]})

df
```

Out[17]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Bhaskar	1	22	21	18	20	21
1	Bhaskar	2	21	20	17	22	24
2	Bhaskar	3	14	19	15	24	23
3	Zuhaire	1	20	17	22	24	19
4	Zuhaire	2	23	15	21	25	15
5	Zuhaire	3	22	18	19	23	13
6	Ashravy	1	23	19	20	15	22
7	Ashravy	2	24	22	24	17	21
8	Ashravy	3	12	25	19	21	23
9	Mishti	1	15	22	25	22	22
10	Mishti	2	18	21	25	24	23
11	Mishti	3	17	18	20	25	20

In [18]:

```
print(str(df['Name'])+"\n") # return as a series.
print(str(df['UT'].sum())+'\n')
print(df.UT) # return as a series.
```

```
0     Bhaskar
1     Bhaskar
2     Bhaskar
3     Zuhaire
4     Zuhaire
5     Zuhaire
6     Ashravy
7     Ashravy
8     Ashravy
9      Mishti
10     Mishti
11     Mishti
Name: Name, dtype: object

24

0     1
1     2
2     3
3     1
4     2
5     3
6     1
7     2
8     3
9     1
10    2
11    3
Name: UT, dtype: int64
```

```
In [22]: print(df['Name']=='Bhaskar') # return a series only TRUE corresponding to name bhas

         df.loc[df['Name']=='Bhaskar']
```

```
0      True
1      True
2      True
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
Name: Name, dtype: bool
```

Out[22]:

	Name	UT	Maths	Science	S.St	Hindi	Eng
0	Bhaskar	1	22	21	18	20	21
1	Bhaskar	2	21	20	17	22	24
2	Bhaskar	3	14	19	15	24	23

```
In [25]: # If we want to access record or data from a data frame row wise or
         # column wise then iteration is used. Pandas provide 2 functions to
         # perform iterations 1. iterrows () 2. iteritems ()

         # iterrows() --> It is used to access the data row wise.

         for(row_index,row_values) in df.iterrows():
             print('\n Row index is ::',row_index)
             print('Row value is ::')
             print(row_values)
```

```
 Row index is :: 0
Row value is ::
Name       Bhaskar
UT               1
Maths           22
Science         21
S.St            18
Hindi           20
Eng             21
Name: 0, dtype: object

 Row index is :: 1
Row value is ::
Name       Bhaskar
UT               2
Maths           21
Science         20
S.St            17
Hindi           22
Eng             24
Name: 1, dtype: object

 Row index is :: 2
Row value is ::
Name       Bhaskar
UT               3
Maths           14
Science         19
S.St            15
Hindi           24
Eng             23
Name: 2, dtype: object

 Row index is :: 3
Row value is ::
Name       Zuhaire
UT               1
Maths           20
Science         17
S.St            22
Hindi           24
Eng             19
Name: 3, dtype: object

 Row index is :: 4
Row value is ::
Name       Zuhaire
UT               2
Maths           23
Science         15
S.St            21
Hindi           25
Eng             15
Name: 4, dtype: object

 Row index is :: 5
Row value is ::
Name       Zuhaire
UT               3
Maths           22
Science         18
S.St            19
Hindi           23
Eng             13
Name: 5, dtype: object

 Row index is :: 6
Row value is ::
Name       Ashravy
UT               1
Maths           23
Science         19
S.St            20
```

```
Hindi           15
Eng             22
Name: 6, dtype: object

 Row index is :: 7
Row value is ::
Name       Ashravy
UT               2
Maths           24
Science         22
S.St            24
Hindi           17
Eng             21
Name: 7, dtype: object

 Row index is :: 8
Row value is ::
Name       Ashravy
UT               3
Maths           12
Science         25
S.St            19
Hindi           21
Eng             23
Name: 8, dtype: object

 Row index is :: 9
Row value is ::
Name       Mishti
UT              1
Maths          15
Science        22
S.St           25
Hindi          22
Eng            22
Name: 9, dtype: object

 Row index is :: 10
Row value is ::
Name       Mishti
UT              2
Maths          18
Science        21
S.St           25
Hindi          24
Eng            23
Name: 10, dtype: object

 Row index is :: 11
Row value is ::
Name       Mishti
UT              3
Maths          17
Science        18
S.St           20
Hindi          25
Eng            20
Name: 11, dtype: object
```

In [29]:

```python
# iteritems() --> It is used to access the data column wise.

for(col_name,col_value) in df.iteritems():
   print('\n Row index is ::',col_name)
   print('Row value is ::')
   print(col_value)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2420\513167700.py in ?()
      1 # iteritems() --> It is used to access the data column wise.
      2 
----> 3 for(col_name,col_value) in df.iteritems():
      4     print('\n Row index is ::',col_name)
      5     print('Row value is ::')
      6     print(col_value)

c:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\cor
e\generic.py in ?(self, name)
   5985             and name not in self._accessors
   5986             and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'iteritems'
```

In [32]:
```
# Add or Rename a column in a dataframe.

s = pd.Series([10,15,18,22])
df_x=pd.DataFrame(s)
df_x.columns=['List1'] # To Rename the default column of DataFrame as List1
df_x['List2']=20 # To create a new column List2 with all values as 20
df_x['List3']=df_x['List1']+df_x['List2']
# Add Column1 and Column2 and store in
# New column List3
print(df_x)
```

```
   List1  List2  List3
0     10     20     30
1     15     20     35
2     18     20     38
3     22     20     42
```

In [34]:
```
# To delete a column in a DataFrame.

# We can delete the column from a data frame by using any of
# the the following -
# 1} del() 2} pop() 3} drop()

s = pd.Series([10,15,18,22])
df_x=pd.DataFrame(s)
df_x.columns=['List1']
df_x['List2']=20
df_x['List3']=df_x['List1']+df_x['List2']
print(df_x)
del df_x['List3']
print(df_x) # We can simply delete a column by passing column name in subscript wit

# df_x.pop('List2')  -- alternative.
```

```
   List1  List2  List3
0     10     20     30
1     15     20     35
2     18     20     38
3     22     20     42
   List1  List2
0     10     20
1     15     20
2     18     20
3     22     20
```

In [49]:
```
# To delete a column using drop.

import pandas as pd

se = pd.Series([10,20,30,40,50])
dfl = pd.DataFrame(se)

dfl.columns = ['List1']
dfl['List2'] = 40
```

```
print(dfl)

df1=dfl.drop('List2',axis=1) # (axis=1) means to delete data column-wise.
print(df1)

df2=dfl.drop(index=[2,3],axis=0) # (axis=0) means to delete data row wise with give
print(df2)
```

```
   List1  List2
0     10     40
1     20     40
2     30     40
3     40     40
4     50     40
   List1
0     10
1     20
2     30
3     40
4     50
   List1  List2
0     10     40
1     20     40
4     50     40
```

In [7]:

```
# DESCRIPTIVE STATISTICS

# Calculating maximum value.

print(df.max())

#Maximum value in name column (alphabetically)
#Maximum value in column UT
#Maximum value in column Maths
#Maximum value in column Science
#Maximum value in column S.St
#Maximum value in column Hindi
#Maximum value in column Eng

print(df.max(numeric_only=True))
#If we want to output maximum value for the columns
#having only numeric values, then we can set the
#parameter numeric_only=True in the max().
```

```
Name       Zuhaire
UT               3
Maths           24
Science         25
S.St            25
Hindi           25
Eng             24
dtype: object
UT          3
Maths      24
Science    25
S.St       25
Hindi      25
Eng        24
dtype: int64
```

In [8]:

```
# Write the statements to output the maximum marks obtained in each subject in Unit

df1 = df[df.UT == 2]
print('\nResult of Unit Test 2:\n\n',df1)
```

```
Result of Unit Test 2:

        Name  UT  Maths  Science  S.St  Hindi  Eng
1      Raman   2     21       20    17     22   24
4    Zuhaire   2     23       15    21     25   15
7    Ashravy   2     24       22    24     17   21
10    Mishti   2     18       21    25     24   23
```

In [54]:

```
import pandas as pd
```

```
       Name  UT  Maths  Science  S.St  Hindi  Eng
1  Bhaskar   2     21       20    17     22   24
2  Bhaskar   3     14       19    15     24   23
3  Zuhaire   1     20       17    22     24   19
1  Bhaskar   2     21       20    17     22   24
2  Bhaskar   3     14       19    15     24   23
3  Zuhaire   1     20       17    22     24   19
1  Bhaskar   2     21       20    17     22   24
2  Bhaskar   3     14       19    15     24   23
3  Zuhaire   1     20       17    22     24   19
1  Bhaskar   2     21       20    17     22   24
2  Bhaskar   3     14       19    15     24   23
3  Zuhaire   1     20       17    22     24   19
```

In [9]:
```
print('\nMaximum Mark obtained inEach Subject in Unit Test 2: \n\n',df1.max(numeric
```

```
Maximum Mark obtained inEach Subject in Unit Test 2:

 UT          2
Maths       24
Science     22
S.St        25
Hindi       25
Eng         24
dtype: int64
```

In [61]:
```
# Accessing the data frame through loc() and iloc() method or indexing using Labels

# It is used to access a group of rows and columns.

# Syntax --> Df.loc[StartRow : EndRow, StartColumn : EndColumn]
# The loc() function is label based data selecting method which
# means that we have to pass the name of the row or column which
# we want to select.

# This method includes the last element of the range passed in it,
# unlike iloc(). loc() can accept the boolean data unlike iloc().

import pandas as pd

data = pd.DataFrame({'Brand': ['Maruti', 'Hyundai', 'Tata',
                                                        'Mahindra', 'Maruti', 'Hyun
                                                        'Renault', 'Tata', 'Maruti'
                                        'Year': [2012, 2014, 2011, 2015, 2012,
                                                        2016, 2014, 2018, 2019],
                                        'Kms Driven': [50000, 30000, 60000,
                                                                        25000, 1000
                                                                        31000, 1500
                                        'City': ['Gurgaon', 'Delhi', 'Mumbai',
                                                        'Delhi', 'Mumbai', 'Delhi',
                                                        'Mumbai', 'Chennai', 'Ghazi
                                        'Mileage': [28, 27, 25, 26, 28,
                                                                29, 24, 21, 24]})

display(data)

# selecting cars with brand 'Maruti' and Mileage > 25
x1 = display(data.loc[(data.Brand == 'Maruti') & (data.Mileage > 25)])
print("\n\n",x1,"\n")

# selecting range of rows from 2 to 5
x2 = display(data.loc[2: 5])
print("\n\n",x2,"\n")


# updating values of Mileage if Year < 2015
data.loc[(data.Year < 2015), ['Mileage']] = 22
display(data)
```

	Brand	Year	Kms Driven	City	Mileage
0	Maruti	2012	50000	Gurgaon	28
1	Hyundai	2014	30000	Delhi	27
2	Tata	2011	60000	Mumbai	25
3	Mahindra	2015	25000	Delhi	26
4	Maruti	2012	10000	Mumbai	28
5	Hyundai	2016	46000	Delhi	29
6	Renault	2014	31000	Mumbai	24
7	Tata	2018	15000	Chennai	21
8	Maruti	2019	12000	Ghaziabad	24

	Brand	Year	Kms Driven	City	Mileage
0	Maruti	2012	50000	Gurgaon	28
4	Maruti	2012	10000	Mumbai	28

```
None
```

	Brand	Year	Kms Driven	City	Mileage
2	Tata	2011	60000	Mumbai	25
3	Mahindra	2015	25000	Delhi	26
4	Maruti	2012	10000	Mumbai	28
5	Hyundai	2016	46000	Delhi	29

```
None
```

	Brand	Year	Kms Driven	City	Mileage
0	Maruti	2012	50000	Gurgaon	22
1	Hyundai	2014	30000	Delhi	22
2	Tata	2011	60000	Mumbai	22
3	Mahindra	2015	25000	Delhi	26
4	Maruti	2012	10000	Mumbai	22
5	Hyundai	2016	46000	Delhi	29
6	Renault	2014	31000	Mumbai	22
7	Tata	2018	15000	Chennai	21
8	Maruti	2019	12000	Ghaziabad	24

In [63]:
```
# The iloc() function is an indexed-based selecting method which
# means that we have to pass an integer index in the method to
# select a specific row/column. This method does not include the
#last element of the range passed in it unlike loc().
#iloc() does not accept the boolean data unlike loc().
#Operations performed using iloc() are:

# selecting 0th, 2nd, 4th, and 7th index rows
display(data.iloc[[0, 2, 4, 7]])


# selecting rows from 1 to 4 and columns from 2 to 4
display(data.iloc[1: 5, 2: 5])
```

	Brand	Year	Kms Driven	City	Mileage
0	Maruti	2012	50000	Gurgaon	22
2	Tata	2011	60000	Mumbai	22
4	Maruti	2012	10000	Mumbai	22
7	Tata	2018	15000	Chennai	21

	Kms Driven	City	Mileage
1	30000	Delhi	22
2	60000	Mumbai	22
3	25000	Delhi	26
4	10000	Mumbai	22

In [1]:
```
# By default, the max() method finds the maximum
# value of each column (which means, axis=0). However,
# to find the maximum value of each row, we have to
# specify axis = 1 as its argument.

df.max(axis=1)
```