# pandas1

November 27, 2023

[1]:
```
L=[1,2,3,4,5]
```

[2]:
```
from pandas import DataFrame as DF
```

[3]:
```
Data1=DF(L)
```

[4]:
```
Data1
```

[4]:

| | 0 |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |

[5]:
```
L=[(1,2,3),(4,5,6),(7,8)]
```

[6]:
```
Data2=DF(L)
```

[7]:
```
Data2
```

[7]:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 2 | 3.0 |
| 1 | 4 | 5 | 6.0 |
| 2 | 7 | 8 | NaN |

[8]:
```
T=([1,2,3],[4,5,6])
```

[9]:
```
Data3=DF(T)
```

[10]:
```
Data3
```

[10]:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 |

[11]:
```
S={(1,2,3),(4,5,6)}
```

[12]:
```
Data4=DF(S)
```

[13]:
```
Data4
```

[13]:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 |

[14]:
```
print(type(Data4))
```

```
<class 'pandas.core.frame.DataFrame'>
```

[15]:
```
Data1.index
```

[15]:
```
RangeIndex(start=0, stop=5, step=1)
```

[16]:
```
Data2.index
```

[16]:
```
RangeIndex(start=0, stop=3, step=1)
```

[17]:
```
for i in Data1.index:
    print(i)
```

```
0
1
2
3
4
```

[18]:
```
type(i)
```

[18]:
```
int
```

[19]:
```
Data1.index.dtype
```

[19]:
```
dtype('int64')
```

[20]:
```
Data2.columns
```

[20]:
```
RangeIndex(start=0, stop=3, step=1)
```

[21]:
```
for j in Data2.columns:
    print(j)
```

```
0
1
2
```

[22]:
```
Data2.columns.dtype
```

[22]:
```
dtype('int64')
```

[23]:
```
Data1.values
```

[23]:
```
array([[1],
       [2],
       [3],
       [4],
       [5]], dtype=int64)
```

[24]:
```
Data2.values
```

[24]:
```
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.],
       [ 7.,  8., nan]])
```

[25]:
```
type(Data2.values)
```

[25]:
```
numpy.ndarray
```

[26]:
```
Dict1={'Name':'Zachariah','empno':103,'Salary':22000}
```

[27]:
```
Data11=DF(Dict1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[27], line 1
----> 1 Data11=DF(Dict1)

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame
 ↪py:736, in DataFrame.__init__(self, data, index, columns, dtype, copy)
    730     mgr = self._init_mgr(
    731         data, axes={"index": index, "columns": columns}, dtype=dtype,␣
 ↪copy=copy
    732     )
    734 elif isinstance(data, dict):
    735     # GH#38939 de facto copy defaults to False only in non-dict cases
--> 736     mgr =␣
 ↪dict_to_mgr(data, index, columns, dtype=dtype, copy=copy, typ=manager)
    737 elif isinstance(data, ma.MaskedArray):
    738     from numpy.ma import mrecords

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\inter als\construct
 ↪py:503, in dict_to_mgr(data, index, columns, dtype, typ, copy)
```

```
    499     else:
    500         # dtype check to exclude e.g. range objects, scalars
    501         arrays = [x.copy() if hasattr(x, "dtype") else x for x in array
--> 503 return␣
 ↪arrays_to_mgr(arrays, columns, index, dtype=dtype, typ=typ, consolidate=copy)

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\inter als\construct
 ↪py:114, in arrays_to_mgr(arrays, columns, index, dtype, verify_integrity, typ␣
 ↪consolidate)
    111 if verify_integrity:
    112     # figure out the index, if necessary
    113     if index is None:
--> 114         index = _extract_index(arrays)
    115     else:
    116         index = ensure_index(index)

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\inter als\construct
 ↪py:667, in _extract_index(data)
    664         raise ValueError("Per-column arrays must each be 1-dimensional"
    666 if not indexes and not raw_lengths:
--> 667     raise ValueError("If using all scalar values, you must pass an␣
 ↪index")
    669 if have_series:
    670     index = union_indexes(indexes)

ValueError: If using all scalar values, you must pass an index
```

[28]:
```
Data11=DF(Dict1,index=[1])
```

[29]:
```
Data11
```

[29]:

| | Name | empno | Salary |
|---|---|---|---|
| 1 | Zachariah | 103 | 22000 |

[30]:
```
Dict2={'Name':{'one':'Mohan','two':'Shahana','three':'Zachariah'},'empno':
 ↪{'one':101,'two':102,'three':103},'Salary':{'one':14000,'two':20000,'three':
 ↪22000}}
```

[31]:
```
Data12=DF(Dict2)
```

[32]:
```
Data12
```

[32]:

| | Name | empno | Salary |
|---|---|---|---|
| one | Mohan | 101 | 14000 |
| two | Shahana | 102 | 20000 |

```
three  Zachariah    103   22000
```

[33]:
```
Dict3={'Name':['Mohan','Shahana','Zachariah'],'empno':[101,102,103],'Salary':
↪[14000,20000,22000]}
```

[34]:
```
Data13=DF(Dict3)
```

[35]:
```
Data13
```

[35]:
```
         Name  empno  Salary
0       Mohan    101   14000
1     Shahana    102   20000
2  Zachariah    103   22000
```

[36]:
```
Data=DF()
```

[37]:
```
Data
```

[37]:
```
Empty DataFrame
Columns: []
Index: []
```

[38]:
```
type(Data)
```

[38]:
```
pandas.core.frame.DataFrame
```

[39]:
```
L11=[1]
```

[40]:
```
Dat11=DF(L11)
```

[41]:
```
Dat11
```

[41]:
```
   0
0  1
```

[42]:
```
S11={1}
```

[43]:
```
Dat12=DF(S11)
```

[44]:
```
Dat12
```

[44]:
```
   0
0  1
```

[45]:
```
T11=1,
```

[46]:
```
Dat13=DF(T11)
```

[47]:
```
Dat13
```

[47]:
```
   0
0  1
```

[48]:
```
import pandas as pd
```

[49]:
```
DataSet1=pd.read_csv('emp.csv')
```

[50]:
```
DataSet1
```

[50]:
```
            E_Name  E_ID    E_Desig  E_Salary
0            K.Raj   101  scientist     54000
1     Mohan Raj M.   102   engineer     50000
2            M.Ram   103  scientist     80000
3        Ram Kumar   104  scientist     66000
4  Mohan Babu K.K.   106    officer     57000
5          K.Gopal   107  scientist     50000
6      Anil Raj M.   108   engineer     62000
7         Amala P.   201  scientist     70000
8             Uma    204    officer     60000
9             Suma   206   engineer     57000
```

[51]:
```
DataSet1.values
```

[51]:
```
array([['K.Raj', 101, 'scientist', 54000],
       ['Mohan Raj M.', 102, 'engineer', 50000],
       ['M.Ram', 103, 'scientist', 80000],
       ['Ram Kumar', 104, 'scientist', 66000],
       ['Mohan Babu K.K.', 106, 'officer', 57000],
       ['K.Gopal', 107, 'scientist', 50000],
       ['Anil Raj M.', 108, 'engineer', 62000],
       ['Amala P.', 201, 'scientist', 70000],
       ['Uma ', 204, 'officer', 60000],
       ['Suma', 206, 'engineer', 57000]], dtype=object)
```

[52]:
```
DataSet1.size
```

[52]:
```
40
```

[53]:
```
DataSet1.shape
```

[53]:
```
(10, 4)
```

[54]:
```
DataSet1.ndim
```

[54]:
```
2
```

[55]:
```
DataSet1.reshape(2,5,4)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5868\4106453542.py in ?()
----> 1 DataSet1.reshape(2,5,4)

~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\generic
 ↪py in ?(self, name)
   6200             and name not in self._accessors
   6201             and self._info_axis.
 ↪_can_hold_identifiers_and_holds_name(name)
   6202         ):
   6203             return self[name]
-> 6204         return object.__getattribute__(self, name)

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

[56]:
```
DataSet1.values.reshape(2,5,4)
```

[56]:
```
array([[['K.Raj', 101, 'scientist', 54000],
         ['Mohan Raj M.', 102, 'engineer', 50000],
         ['M.Ram', 103, 'scientist', 80000],
         ['Ram Kumar', 104, 'scientist', 66000],
         ['Mohan Babu K.K.', 106, 'officer', 57000]],

        [['K.Gopal', 107, 'scientist', 50000],
         ['Anil Raj M.', 108, 'engineer', 62000],
         ['Amala P.', 201, 'scientist', 70000],
         ['Uma ', 204, 'officer', 60000],
         ['Suma', 206, 'engineer', 57000]]], dtype=object)
```

[57]:
```
DataSet1.index
```

[57]:
```
RangeIndex(start=0, stop=10, step=1)
```

[58]:
```
for col in DataSet1.columns:
    print(col)
```

```
E_Name
E_ID
E_Desig
E_Salary
```

[59]:
```
DataSet1.columns
```

[59]:
```
Index(['E_Name', 'E_ID', 'E_Desig', 'E_Salary'], dtype='object')
```

[60]:
```
DataSet1['E_Name']
```

[60]:
```
0              K.Raj
1       Mohan Raj M.
2              M.Ram
3          Ram Kumar
4    Mohan Babu K.K.
5            K.Gopal
6        Anil Raj M.
7           Amala P.
8               Uma
9               Suma
Name: E_Name, dtype: object
```

[61]:
```
DataSet1['E_ID']
```

[61]:
```
0    101
1    102
2    103
3    104
4    106
5    107
6    108
7    201
8    204
9    206
Name: E_ID, dtype: int64
```

[62]:
```
DataSet1['1']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\index
 ↪py:3790, in Index.get_loc(self, key)
   3789 try:
-> 3790     return self._engine.get_loc(casted_key)
   3791 except KeyError as err:

File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()
```

```
File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

KeyError: '1'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[62], line 1
----> 1 DataSet1['1']

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame
 ↪py:3896, in DataFrame.__getitem__(self, key)
   3894 if self.columns.nlevels > 1:
   3895     return self._getitem_multilevel(key)
-> 3896 indexer = self.columns.get_loc(key)
   3897 if is_integer(indexer):
   3898     indexer = [indexer]

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\index
 ↪py:3797, in Index.get_loc(self, key)
   3792     if isinstance(casted_key, slice) or (
   3793         isinstance(casted_key, abc.Iterable)
   3794         and any(isinstance(x, slice) for x in casted_key)
   3795     ):
   3796         raise InvalidIndexError(key)
-> 3797     raise KeyError(key) from err
   3798 except TypeError:
   3799     # If we have a listlike key, _check_indexing_error will raise
   3800     #  InvalidIndexError. Otherwise we fall through and re-raise
   3801     #  the TypeError.
   3802     self._check_indexing_error(key)

KeyError: '1'
```

[63]:
```
DataSet1.loc[1]
```

[63]:
```
E_Name      Mohan Raj M.
E_ID                 102
E_Desig         engineer
E_Salary            50000
Name: 1, dtype: object
```

[64]:
```
DataSet1.iloc[1]
```

[64]:
```
E_Name      Mohan Raj M.
E_ID                 102
E_Desig         engineer
E_Salary           50000
Name: 1, dtype: object
```

[65]:
```
DataSet1.iloc[:,1]
```

[65]:
```
0    101
1    102
2    103
3    104
4    106
5    107
6    108
7    201
8    204
9    206
Name: E_ID, dtype: int64
```

[66]:
```
DataSet1.loc[[1,3,5]]
```

[66]:

| | E_Name | E_ID | E_Desig | E_Salary |
|---|---|---|---|---|
| 1 | Mohan Raj M. | 102 | engineer | 50000 |
| 3 | Ram Kumar | 104 | scientist | 66000 |
| 5 | K.Gopal | 107 | scientist | 50000 |

[67]:
```
DataSet1.loc[1:3]
```

[67]:

| | E_Name | E_ID | E_Desig | E_Salary |
|---|---|---|---|---|
| 1 | Mohan Raj M. | 102 | engineer | 50000 |
| 2 | M.Ram | 103 | scientist | 80000 |
| 3 | Ram Kumar | 104 | scientist | 66000 |

[68]:
```
DataSet1.iloc[1,3]
```

[68]:
```
50000
```

[69]:
```
DataSet1.loc[2,'E_ID']
```

[69]:
```
103
```

[70]:
```
DataSet1.loc[2,'E_ID']=110
```

[71]:
```
DataSet1
```

[71]:

| | E_Name | E_ID | E_Desig | E_Salary |
|---|---|---|---|---|
| 0 | K.Raj | 101 | scientist | 54000 |
| 1 | Mohan Raj M. | 102 | engineer | 50000 |
| 2 | M.Ram | 110 | scientist | 80000 |
| 3 | Ram Kumar | 104 | scientist | 66000 |
| 4 | Mohan Babu K.K. | 106 | officer | 57000 |
| 5 | K.Gopal | 107 | scientist | 50000 |
| 6 | Anil Raj M. | 108 | engineer | 62000 |
| 7 | Amala P. | 201 | scientist | 70000 |
| 8 | Uma | 204 | officer | 60000 |
| 9 | Suma | 206 | engineer | 57000 |

[72]:
```
DataSet1.drop('E_ID')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[72], line 1
----> 1 DataSet1.drop('E_ID')

File␣
 ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame
 ↪py:5347, in DataFrame.drop(self, labels, axis, index, columns, level, inplace␣
 ↪errors)
   5199 def drop(
   5200     self,
   5201     labels: IndexLabel | None = None,
   (…)
   5208     errors: IgnoreRaise = "raise",
   5209 ) -> DataFrame | None:
   5210     """
   5211     Drop specified labels from rows or columns.
   5212
   (…)
   5345             weight  1.0     0.8
   5346     """
-> 5347     return super().drop(
   5348         labels=labels,
   5349         axis=axis,
   5350         index=index,
   5351         columns=columns,
   5352         level=level,
   5353         inplace=inplace,
   5354         errors=errors,
   5355     )
```

```
File␣
  ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\gener .c.
  ↪py:4711, in NDFrame.drop(self, labels, axis, index, columns, level, inplace,␣
  ↪errors)
    4709 for axis, labels in axes.items():
    4710     if labels is not None:
-> 4711         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
    4713 if inplace:
    4714     self._update_inplace(obj)

File␣
  ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\gener .c.
  ↪py:4753, in NDFrame._drop_axis(self, labels, axis, level, errors, only_slice)
    4751         new_axis = axis.drop(labels, level=level, errors=errors)
    4752     else:
-> 4753         new_axis = axis.drop(labels, errors=errors)
    4754     indexer = axis.get_indexer(new_axis)
    4756 # Case for non-unique axis
    4757 else:

File␣
  ↪~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\index s\base.
  ↪py:6992, in Index.drop(self, labels, errors)
    6990 if mask.any():
    6991     if errors != "ignore":
-> 6992         raise KeyError(f"{labels[mask].tolist()} not found in axis")
    6993     indexer = indexer[~mask]
    6994 return self.delete(indexer)

KeyError: "['E_ID'] not found in axis"
```

[73]:
```
DataSet1.drop('E_ID',axis=1)
```

[73]:

| | E_Name | E_Desig | E_Salary |
|---|---|---|---|
| 0 | K.Raj | scientist | 54000 |
| 1 | Mohan Raj M. | engineer | 50000 |
| 2 | M.Ram | scientist | 80000 |
| 3 | Ram Kumar | scientist | 66000 |
| 4 | Mohan Babu K.K. | officer | 57000 |
| 5 | K.Gopal | scientist | 50000 |
| 6 | Anil Raj M. | engineer | 62000 |
| 7 | Amala P. | scientist | 70000 |
| 8 | Uma | officer | 60000 |
| 9 | Suma | engineer | 57000 |

[74]:
```
DataSet1
```

[74]:

| | E_Name | E_ID | E_Desig | E_Salary |
|---|---|---|---|---|
| 0 | K.Raj | 101 | scientist | 54000 |
| 1 | Mohan Raj M. | 102 | engineer | 50000 |
| 2 | M.Ram | 110 | scientist | 80000 |
| 3 | Ram Kumar | 104 | scientist | 66000 |
| 4 | Mohan Babu K.K. | 106 | officer | 57000 |
| 5 | K.Gopal | 107 | scientist | 50000 |
| 6 | Anil Raj M. | 108 | engineer | 62000 |
| 7 | Amala P. | 201 | scientist | 70000 |
| 8 | Uma | 204 | officer | 60000 |
| 9 | Suma | 206 | engineer | 57000 |

[75]:
```
DataSet1.drop('E_ID',axis=1,inplace=True)
```

[76]:
```
DataSet1
```

[76]:

| | E_Name | E_Desig | E_Salary |
|---|---|---|---|
| 0 | K.Raj | scientist | 54000 |
| 1 | Mohan Raj M. | engineer | 50000 |
| 2 | M.Ram | scientist | 80000 |
| 3 | Ram Kumar | scientist | 66000 |
| 4 | Mohan Babu K.K. | officer | 57000 |
| 5 | K.Gopal | scientist | 50000 |
| 6 | Anil Raj M. | engineer | 62000 |
| 7 | Amala P. | scientist | 70000 |
| 8 | Uma | officer | 60000 |
| 9 | Suma | engineer | 57000 |

[77]:
```
PhoneNo=[932287278, 962287278, 914287275, 914287335,␣
↪962387278,932287278,962287278, 89287275,94287335, 962387278]
```

[78]:
```
PhoneNo
```

[78]:
```
[932287278,
 962287278,
 914287275,
 914287335,
 962387278,
 932287278,
 962287278,
 89287275,
 94287335,
 962387278]
```

[79]:
```
DataSet1['PhoneNo']=PhoneNo
```

[80]:
```
DataSet1
```

[80]:

| | E_Name | E_Desig | E_Salary | PhoneNo |
|---|---|---|---|---|
| 0 | K.Raj | scientist | 54000 | 932287278 |
| 1 | Mohan Raj M. | engineer | 50000 | 962287278 |
| 2 | M.Ram | scientist | 80000 | 914287275 |
| 3 | Ram Kumar | scientist | 66000 | 914287335 |
| 4 | Mohan Babu K.K. | officer | 57000 | 962387278 |
| 5 | K.Gopal | scientist | 50000 | 932287278 |
| 6 | Anil Raj M. | engineer | 62000 | 962287278 |
| 7 | Amala P. | scientist | 70000 | 89287275 |
| 8 | Uma | officer | 60000 | 94287335 |
| 9 | Suma | engineer | 57000 | 962387278 |

[81]:
```
DataSet1.assign(Mobile=PhoneNo)
```

[81]:

| | E_Name | E_Desig | E_Salary | PhoneNo | Mobile |
|---|---|---|---|---|---|
| 0 | K.Raj | scientist | 54000 | 932287278 | 932287278 |
| 1 | Mohan Raj M. | engineer | 50000 | 962287278 | 962287278 |
| 2 | M.Ram | scientist | 80000 | 914287275 | 914287275 |
| 3 | Ram Kumar | scientist | 66000 | 914287335 | 914287335 |
| 4 | Mohan Babu K.K. | officer | 57000 | 962387278 | 962387278 |
| 5 | K.Gopal | scientist | 50000 | 932287278 | 932287278 |
| 6 | Anil Raj M. | engineer | 62000 | 962287278 | 962287278 |
| 7 | Amala P. | scientist | 70000 | 89287275 | 89287275 |
| 8 | Uma | officer | 60000 | 94287335 | 94287335 |
| 9 | Suma | engineer | 57000 | 962387278 | 962387278 |

[82]:
```
DataSet1.assign(Mobile=PhoneNo,inplace=True)
```

[82]:

| | E_Name | E_Desig | E_Salary | PhoneNo | Mobile | inplace |
|---|---|---|---|---|---|---|
| 0 | K.Raj | scientist | 54000 | 932287278 | 932287278 | True |
| 1 | Mohan Raj M. | engineer | 50000 | 962287278 | 962287278 | True |
| 2 | M.Ram | scientist | 80000 | 914287275 | 914287275 | True |
| 3 | Ram Kumar | scientist | 66000 | 914287335 | 914287335 | True |
| 4 | Mohan Babu K.K. | officer | 57000 | 962387278 | 962387278 | True |
| 5 | K.Gopal | scientist | 50000 | 932287278 | 932287278 | True |
| 6 | Anil Raj M. | engineer | 62000 | 962287278 | 962287278 | True |
| 7 | Amala P. | scientist | 70000 | 89287275 | 89287275 | True |
| 8 | Uma | officer | 60000 | 94287335 | 94287335 | True |
| 9 | Suma | engineer | 57000 | 962387278 | 962387278 | True |

[83]:
```
DataSet1
```

[83]:

| | E_Name | E_Desig | E_Salary | PhoneNo |
|---|---|---|---|---|
| 0 | K.Raj | scientist | 54000 | 932287278 |
| 1 | Mohan Raj M. | engineer | 50000 | 962287278 |
| 2 | M.Ram | scientist | 80000 | 914287275 |
| 3 | Ram Kumar | scientist | 66000 | 914287335 |

```
4  Mohan Babu K.K.    officer     57000  962387278
5          K.Gopal  scientist     50000  932287278
6      Anil Raj M.   engineer     62000  962287278
7         Amala P.  scientist     70000   89287275
8             Uma     officer     60000   94287335
9             Suma   engineer     57000  962387278
```

[84]:
```
DataSet1.drop(1)
```

[84]:
```
            E_Name    E_Desig  E_Salary    PhoneNo
0            K.Raj  scientist     54000  932287278
2            M.Ram  scientist     80000  914287275
3        Ram Kumar  scientist     66000  914287335
4  Mohan Babu K.K.    officer     57000  962387278
5          K.Gopal  scientist     50000  932287278
6      Anil Raj M.   engineer     62000  962287278
7         Amala P.  scientist     70000   89287275
8             Uma     officer     60000   94287335
9             Suma   engineer     57000  962387278
```

[85]:
```
import pandas as pd
```

[86]:
```
DataSet2=pd.read_excel('Marks.xlsx')
```

[87]:
```
DataSet2
```

[87]:
```
    Sr. No        Name of Student  Registration No   Marks
0        1     PAPPU JNANA MADHURI       2022UG1001     42
1        2              Ritesh Raj       2022UG1002     36
2        3         SHIVANSH RAJPUT       2022UG1003     48
3        4               ARION DAS       2022UG1004     44
4        5  KATRAVATH ROSHAN NAYAK       2022UG1005     20
..     …                       …              …    …
67      68            VISHAL VERMA       2022UG1068     44
68      69          DEVANSH MISHRA       2022UG1069     38
69      70     Amit kumar kushwaha       2022UG1070     30
70      71              YASH MEHTA       2022UG1071     24
71      72             ANKIT KUMAR       2022UG1072     54

[72 rows x 4 columns]
```

[88]:
```
DataSet1=pd.read_excel('Marks.xlsx',sheet_name=0)
```

[89]:
```
DataSet1
```

[89]:
```
    Sr. No        Name of Student  Registration No   Marks
0        1     PAPPU JNANA MADHURI       2022UG1001     42
```

```
1        2              Ritesh Raj       2022UG1002     36
2        3         SHIVANSH RAJPUT       2022UG1003     48
3        4               ARION DAS       2022UG1004     44
4        5  KATRAVATH ROSHAN NAYAK       2022UG1005     20
..     …                       …              …    …
67      68            VISHAL VERMA       2022UG1068     44
68      69          DEVANSH MISHRA       2022UG1069     38
69      70     Amit kumar kushwaha       2022UG1070     30
70      71              YASH MEHTA       2022UG1071     24
71      72             ANKIT KUMAR       2022UG1072     54

[72 rows x 4 columns]
```

[90]:
```
DataSet2=pd.read_excel('Marks.xlsx',sheet_name=1)
```

[91]:
```
DataSet2
```

[91]:
```
    Sr. No.  Name of Student  Registration No   Marks
0         1        AMIT KUMAR       2022UG1073     26
1         2       YATIN KUMAR       2022UG1074     40
2         3  Shivank Tripathi       2022UG1075     52
3         4     Anant Awasthi       2022UG1076     38
4         5        Khush Shah       2022UG1077     50
..      …               …              …    …
68       69   Rishabh Kashyap       2022UG3027     52
69       70           SAKSHAM       2022UG3028     40
70       71      MANISH MEENA       2022UG3029     24
71       72    SHREYANSH GOEL       2022UG2029     58
72       73    PRATHAM DIWEDI       2022UG4019     48

[73 rows x 4 columns]
```

[92]:
```
DataSet2.to_string()
```

[92]:
```
'    Sr. No.                    Name of Student  Registration No   Marks\n0
1                           AMIT KUMAR       2022UG1073     26\n1         2
YATIN KUMAR       2022UG1074     40\n2         3                     Shivank
Tripathi       2022UG1075     52\n3         4                      Anant
Awasthi       2022UG1076     38\n4         5                        Khush
Shah       2022UG1077     50\n5         6                    SHIVANSH SHUKLA
2022UG1078     30\n6         7                       RAJEEV RANJAN
2022UG1079     32\n7         8                  Abhishek kumar jha
2022UG1080     22\n8         9                       ABHINAV PATEL
2022UG1081     32\n9        10                      SAHIL REPURIYA
2022UG1082     42\n10       11                        SAHIL SHUKLA
2022UG1083     45\n11       12                       vishal mishra
2022UG1084     30\n12       13                  ABHISHEK SINGH NEGI
```

| | | | |
|---|---|---|---|
| 2022UG1085 | 52\n13 | 14 | VIKAS CHAURASIA |
| 2022UG1086 | 44\n14 | 15 | MOHIT KUMAR |
| 2022UG1087 | 48\n15 | 16 | PRAVEEN KUMAR GUPTA |
| 2022UG1088 | 56\n16 | 17 | GAUTAM SHARMA |
| 2022UG1089 | 54\n17 | 18 | TIWARI LOKENDRA PURUSHOTTAM |
| 2022UG1090 | 42\n18 | 19 | UTKARSH |
| 2022UG1091 | 32\n19 | 20 | SAMUDRALA DINESH NAVEEN KUMAR |
| 2022UG1092 | 42\n20 | 21 | Kushagra Saxena |
| 2022UG1093 | 38\n21 | 22 | VIBEK PRASAD |
| 2022UG1094 | 36\n22 | 23 | ARYAN |
| 2022UG1095 | 32\n23 | 24 | ANKIT KUMAR |
| 2022UG1096 | 30\n24 | 25 | Navneet Patel |
| 2022UG1097 | 40\n25 | 26 | Shirish |
| 2022UG1098 | 10\n26 | 27 | SHIVAM KUMAR RAH |
| 2022UG1099 | 18\n27 | 28 | SANDEEP GUPTA |
| 2022UG1100 | 30\n28 | 29 | NITIN KUMAR |
| 2022UG1101 | 32\n29 | 30 | Aditya Pandey |
| 2022UG1102 | 24\n30 | 31 | Yash Pundhir |
| 2022UG1103 | 36\n31 | 32 | Koushik Jalan |
| 2022UG1104 | 42\n32 | 33 | NAYAN KUMAR CHOUHAN |
| 2022UG1105 | 6\n33 | 34 | DIPURANJAN SETHY |
| 2022UG1106 | 32\n34 | 35 | ANKIT KUMAR |
| 2022UG1107 | 40\n35 | 36 | BOMMASAMUDRAM SANTHOSH |
| 2022UG1108 | 24\n36 | 37 | SATVIK MISHRA |
| 2022UG1109 | 20\n37 | 38 | Punya Chadha |
| 2022UG1110 | 46\n38 | 39 | NIKHIL SONKAR |
| 2022UG1111 | 34\n39 | 40 | AKASH MISHRA |
| 2022UG1112 | 40\n40 | 41 | SANDEEP KUMAR |
| 2022UG1113 | 36\n41 | 42 | GUGULOTH NITHIN |
| 2022UG1114 | 14\n42 | 43 | GAURANG AGARWAL |
| 2022UG3001 | 52\n43 | 44 | Sagnik Taraphdar |
| 2022UG3002 | 48\n44 | 45 | ANIKET KUMAR |
| 2022UG3003 | 20\n45 | 46 | KOPPISETTI J S VENKATA DURGA MANESH |
| 2022UG3004 | 18\n46 | 47 | VARAD GUPTA |
| 2022UG3005 | 50\n47 | 48 | Md Khateebur Rab |
| 2022UG3006 | 32\n48 | 49 | ADITYA RAJPUT |
| 2022UG3007 | 32\n49 | 50 | ALOK GUPTA |
| 2022UG3008 | 42\n50 | 51 | SHUBH SHUBHANJAL |
| 2022UG3009 | 38\n51 | 52 | NISHANT |
| 2022UG3010 | 46\n52 | 53 | Ayushman Singh |
| 2022UG3011 | 56\n53 | 54 | ADITYA HARSHIT |
| 2022UG3012 | 32\n54 | 55 | AYUSH SHAURYA JHA |
| 2022UG3013 | 44\n55 | 56 | VIVEK KUMAR |
| 2022UG3014 | 32\n56 | 57 | SAMBHAV SRIVASTAVA |
| 2022UG3015 | 26\n57 | 58 | DEVAM SINGH |
| 2022UG3016 | 38\n58 | 59 | SHARAD PRAKASH |
| 2022UG3017 | 58\n59 | 60 | KRISH SRIVASTAVA |

```
2022UG3018     50\n60       61                        AKSHAT KUMAR
2022UG3019     40\n61       62                 ADARSH KUMAR BHARDWAJ
2022UG3020     50\n62       63                         Abhinav Singh
2022UG3021     52\n63       64                          SAJAL NAMDEO
2022UG3022     55\n64       65                    PRITAM ROY SARKAR
2022UG3023     30\n65       66                        ANUBHAV SINGH
2022UG3024     30\n66       67                      TANISHQ RAJORIA
2022UG3025     42\n67       68                       VISHWAS TIWARI
2022UG3026     40\n68       69                     Rishabh Kashyap
2022UG3027     52\n69       70                              SAKSHAM
2022UG3028     40\n70       71                        MANISH MEENA
2022UG3029     24\n71       72                       SHREYANSH GOEL
2022UG2029     58\n72       73                       PRATHAM DIWEDI
2022UG4019     48'
```

[93]:
```
DataSet1.head()
```

[93]:

| | Sr. No | Name of Student | Registration No | Marks |
|---|---|---|---|---|
| 0 | 1 | PAPPU JNANA MADHURI | 2022UG1001 | 42 |
| 1 | 2 | Ritesh Raj | 2022UG1002 | 36 |
| 2 | 3 | SHIVANSH RAJPUT | 2022UG1003 | 48 |
| 3 | 4 | ARION DAS | 2022UG1004 | 44 |
| 4 | 5 | KATRAVATH ROSHAN NAYAK | 2022UG1005 | 20 |

[94]:
```
DataSet1.tail()
```

[94]:

| | Sr. No | Name of Student | Registration No | Marks |
|---|---|---|---|---|
| 67 | 68 | VISHAL VERMA | 2022UG1068 | 44 |
| 68 | 69 | DEVANSH MISHRA | 2022UG1069 | 38 |
| 69 | 70 | Amit kumar kushwaha | 2022UG1070 | 30 |
| 70 | 71 | YASH MEHTA | 2022UG1071 | 24 |
| 71 | 72 | ANKIT KUMAR | 2022UG1072 | 54 |

[95]:
```
DataSet1['Marks'].fillna(0,inplace=True)
```

[96]:
```
DataSet1['Marks']
```

[96]:
```
0     42
1     36
2     48
3     44
4     20
      ..
67    44
68    38
69    30
70    24
```

```
71    54
Name: Marks, Length: 72, dtype: int64
```

[97]:
```
DataSet1['Marks'].sum()
```

[97]:
```
2603
```

[98]:
```
DataSet1['Marks'].mean()
```

[98]:
```
36.15277777777778
```

[99]:
```
DataSet1['Marks'].mode()[0]
```

[99]:
```
48
```

[100]:
```
import numpy as np
```

[101]:
```
A=np.arange(1,17).reshape(4,4)
```

[102]:
```
A
```

[102]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

[103]:
```
from pandas import DataFrame as DF
```

[104]:
```
D_arr=DF(A,index='r1 r2 r3 r4'.split(),columns='c1 c2 c3 c4'.split())
```

[105]:
```
D_arr
```

[105]:
```
    c1  c2  c3  c4
r1   1   2   3   4
r2   5   6   7   8
r3   9  10  11  12
r4  13  14  15  16
```

[106]:
```
B=np.arange(21,37).reshape(4,4)
```

[107]:
```
B
```

[107]:
```
array([[21, 22, 23, 24],
       [25, 26, 27, 28],
       [29, 30, 31, 32],
       [33, 34, 35, 36]])
```

[108]:
```
D_arr2=DF(B,index='r1 r2 r3 r4'.split(),columns='c1 c2 c3 c4'.split())
```

[109]:
```
D_arr2
```

[109]:
```
    c1  c2  c3  c4
r1  21  22  23  24
r2  25  26  27  28
r3  29  30  31  32
r4  33  34  35  36
```

[110]:
```
D_arr+D_arr2
```

[110]:
```
    c1  c2  c3  c4
r1  22  24  26  28
r2  30  32  34  36
r3  38  40  42  44
r4  46  48  50  52
```

[111]:
```
D_arr.add(D_arr2)
```

[111]:
```
    c1  c2  c3  c4
r1  22  24  26  28
r2  30  32  34  36
r3  38  40  42  44
r4  46  48  50  52
```

[112]:
```
D_arr-D_arr2
```

[112]:
```
    c1  c2  c3  c4
r1 -20 -20 -20 -20
r2 -20 -20 -20 -20
r3 -20 -20 -20 -20
r4 -20 -20 -20 -20
```

[113]:
```
D_arr*D_arr2
```

[113]:
```
     c1   c2   c3   c4
r1   21   44   69   96
r2  125  156  189  224
r3  261  300  341  384
r4  429  476  525  576
```

[114]:
```
D_arr/D_arr2
```

[114]:
```
           c1        c2        c3        c4
r1  0.047619  0.090909  0.130435  0.166667
r2  0.200000  0.230769  0.259259  0.285714
```

```
r3  0.310345  0.333333  0.354839  0.375000
r4  0.393939  0.411765  0.428571  0.444444
```

[115]:
```
D_arr.values@D_arr2.values
```

[115]:
```
array([[ 290,  300,  310,  320],
       [ 722,  748,  774,  800],
       [1154, 1196, 1238, 1280],
       [1586, 1644, 1702, 1760]])
```

[116]:
```
D_arr.loc['r1']
```

[116]:
```
c1    1
c2    2
c3    3
c4    4
Name: r1, dtype: int32
```

[117]:
```
D_arr.to_excel('Samp3.xlsx')
```

[118]:
```
D_arr.to_excel('Samp4.xlsx',index=False)
```

[119]:
```
D_arr
```

[119]:
```
    c1  c2  c3  c4
r1   1   2   3   4
r2   5   6   7   8
r3   9  10  11  12
r4  13  14  15  16
```

[120]:
```
New_row={'c1':10,'c2':20,'c3':30,'c4':40}
```

[121]:
```
D_arr=D_arr.add(New_row)
```

[122]:
```
D_arr
```

[122]:
```
    c1  c2  c3  c4
r1  11  22  33  44
r2  15  26  37  48
r3  19  30  41  52
r4  23  34  45  56
```

[123]:
```
D_arr_new=pd.concat([D_arr,D_arr2],ignore_index=True)
```

[124]:
```
D_arr_new
```

[124]:

| | c1 | c2 | c3 | c4 |
|---|---|---|---|---|
| 0 | 11 | 22 | 33 | 44 |
| 1 | 15 | 26 | 37 | 48 |
| 2 | 19 | 30 | 41 | 52 |
| 3 | 23 | 34 | 45 | 56 |
| 4 | 21 | 22 | 23 | 24 |
| 5 | 25 | 26 | 27 | 28 |
| 6 | 29 | 30 | 31 | 32 |
| 7 | 33 | 34 | 35 | 36 |

[ ]:

[ ]: