In [9]:
```python
import numpy as np
```

1.Creating a numpy array

In [36]:
```python
#array from a python list
a=[2,4,6,8,10]
b=np.array(a)
```

In [37]:
```python
b
```

Out[37]:
```
array([ 2,  4,  6,  8, 10])
```

In [38]:
```python
a
```

Out[38]:
```
[2, 4, 6, 8, 10]
```

In [39]:
```python
type(a)
```

Out[39]:
```
list
```

In [40]:
```python
type(b)
```

Out[40]:
```
numpy.ndarray
```

In [42]:
```python
list1=a
arr1=b
```

In [76]:
```python
 list1+1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-76-3611460f1106> in <module>
----> 1 list1+1

TypeError: can only concatenate list (not "int") to list
```

```
In [77]: arr1+1
```

```
Out[77]: array([ 3,  5,  7,  9, 11])
```

```
In [78]: list2=[[9,0,4,6],
               [4,6,2,1]]
         arr2=np.array(list2)
```

```
In [79]: arr2
```

```
Out[79]: array([[9, 0, 4, 6],
                [4, 6, 2, 1]])
```

```
In [80]: #float array
         arr2=np.array(list2,dtype='float')
```

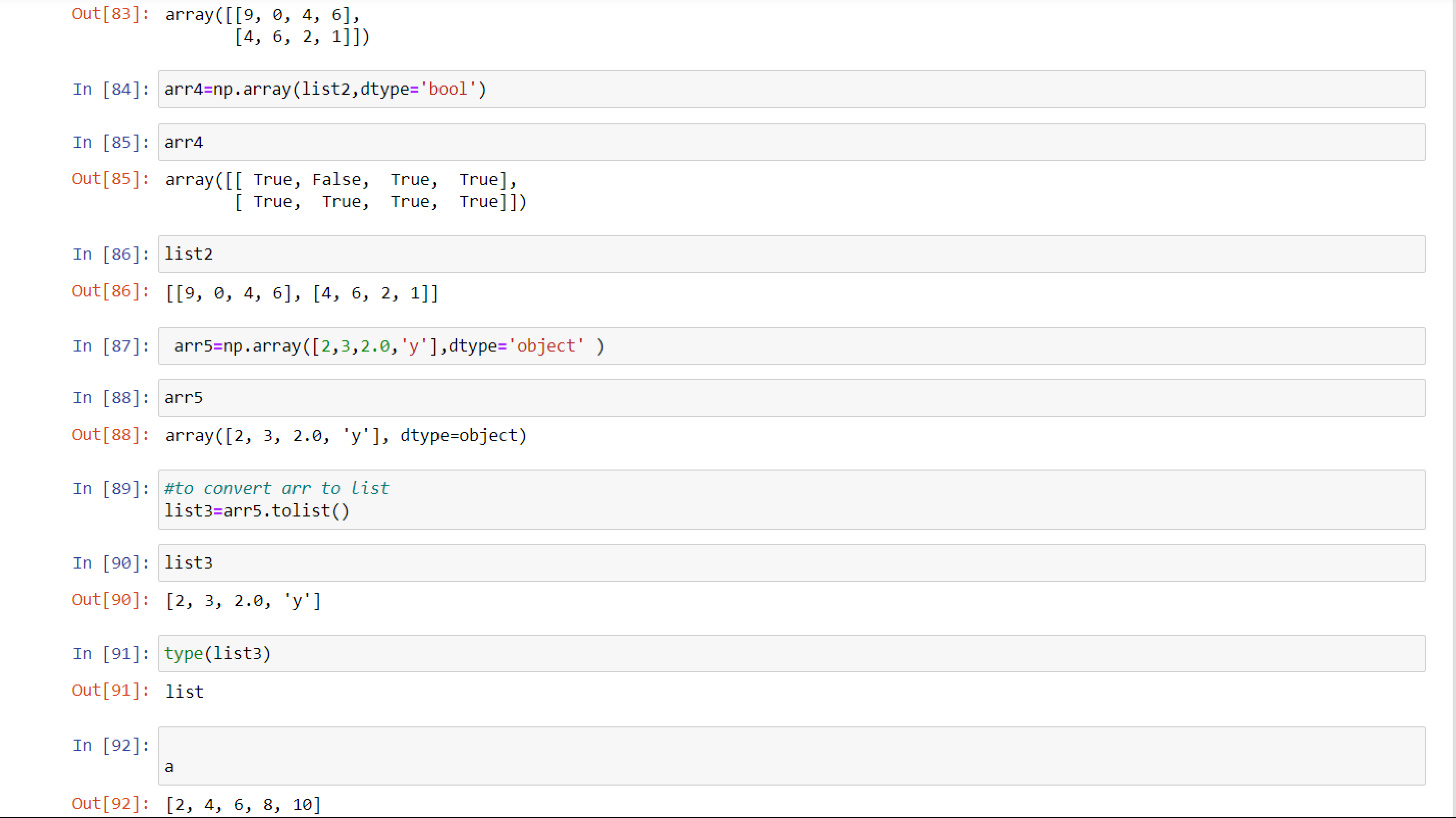
```
In [81]: arr2
```

```
Out[81]: array([[9., 0., 4., 6.],
                [4., 6., 2., 1.]])
```

```
In [82]: arr3=arr2.astype('int')
```

```
In [83]: arr3
```

```
Out[83]: array([[9, 0, 4, 6],
```



```
Out[83]: array([[9, 0, 4, 6],
                [4, 6, 2, 1]])

In [84]: arr4=np.array(list2,dtype='bool')

In [85]: arr4

Out[85]: array([[ True, False,  True,  True],
                [ True,  True,  True,  True]])

In [86]: list2

Out[86]: [[9, 0, 4, 6], [4, 6, 2, 1]]

In [87]:  arr5=np.array([2,3,2.0,'y'],dtype='object' )

In [88]: arr5

Out[88]: array([2, 3, 2.0, 'y'], dtype=object)

In [89]: #to convert arr to list
         list3=arr5.tolist()

In [90]: list3

Out[90]: [2, 3, 2.0, 'y']

In [91]: type(list3)

Out[91]: list

In [92]: 
         a

Out[92]: [2, 4, 6, 8, 10]
```

L2 ARRAY DIMENSION

```
In [94]: arr2

Out[94]: array([[9., 0., 4., 6.],
                [4., 6., 2., 1.]])

In [95]: #shape of array
         arr2.shape

Out[95]: (2, 4)

In [96]: #no. of elements
         arr2.size

Out[96]: 8

In [97]: #type
         arr2.dtype

Out[97]: dtype('float64')

In [98]:  arr2.ndim

Out[98]: 2
```

L3 REVERSING ROWS AND COLUMNS

```
In [100]: arr2

Out[100]: array([[9., 0., 4., 6.],
                 [4., 6., 2., 1.]])
```

```
In [101]: #reverse rows
          arr2[::-1]

Out[101]: array([[4., 6., 2., 1.],
                 [9., 0., 4., 6.]])

In [102]: arr2[::-1,::-1]

Out[102]: array([[1., 2., 6., 4.],
                 [6., 4., 0., 9.]])
```

L4 SPECIFIC ELEMENT EXTRACTION

```
In [104]: arr2

Out[104]: array([[9., 0., 4., 6.],
                 [4., 6., 2., 1.]])

In [105]: arr2[0,:]

Out[105]: array([9., 0., 4., 6.])

In [106]: arr2[:1,:]

Out[106]: array([[9., 0., 4., 6.]])

In [107]: arr2[:2,:]

Out[107]: array([[9., 0., 4., 6.],
                 [4., 6., 2., 1.]])

In [108]: arr2[:-1,:]

Out[108]: array([[9., 0., 4., 6.]])
```

```
In [108]: arr2[:-1,:]

Out[108]: array([[9., 0., 4., 6.]])
```

```
In [109]: #last column
          arr2[:,:3]

Out[109]: array([[9., 0., 4.],
                 [4., 6., 2.]])
```

```
In [110]:  arr2[:,:-1]

Out[110]: array([[9., 0., 4.],
                 [4., 6., 2.]])
```

```
In [111]: arr2[:1,:3] #1,2

Out[111]: array([[9., 0., 4.]])
```

I5 BASIC STATISTICS

```
In [112]:  arr2

Out[112]: array([[9., 0., 4., 6.],
                 [4., 6., 2., 1.]])
```

```
In [113]: arr2.min()

Out[113]: 0.0
```

```
In [114]: arr2.max()

Out[114]: 9.0
```

```
In [115]: arr2.sum()
```

```
Out[115]: 32.0

In [116]: arr2.mean()

Out[116]: 4.0

In [118]: np.median(arr2)

Out[118]: 4.0

In [120]: np.average(arr2)

Out[120]: 4.0

In [121]: #variance
          np.var(arr2)

Out[121]: 7.75
```

I-6 RESHAPING AND FLATENNING

```
In [122]: arr2

Out[122]: array([[9., 0., 4., 6.],
                 [4., 6., 2., 1.]])

In [123]: arr2.shape

Out[123]: (2, 4)

In [126]: arr2.reshape(4,2)
```

```
Out[126]: array([[9., 0.],
                 [4., 6.],
                 [4., 6.],
                 [2., 1.]])
```

```
In [127]: arr2.reshape(1,8)
```

```
Out[127]: array([[9., 0., 4., 6., 4., 6., 2., 1.]])
```

```
In [128]: arr2.reshape(8,1)
```

```
Out[128]: array([[9.],
                 [0.],
                 [4.],
                 [6.],
                 [4.],
                 [6.],
                 [2.],
                 [1.]])
```

```
In [129]: arr2.reshape(2,2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-129-7e35622d1c82> in <module>
----> 1 arr2.reshape(2,2)

ValueError: cannot reshape array of size 8 into shape (2,2)
```

```
In [131]: #single dimension
          f1=arr2.flatten()
```

```
In [133]: f1.ndim

Out[133]: 1
```

L7 RANDOM ARRAYS AND SEQUENCES

```
In [135]: np.arange(10)

Out[135]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [136]: np.arange(2,10)

Out[136]: array([2, 3, 4, 5, 6, 7, 8, 9])

In [137]: np.arange(0,10,2)

Out[137]: array([0, 2, 4, 6, 8])

In [138]: np.arange(10,0)

Out[138]: array([], dtype=int32)

In [139]: #starting from 10 not 0 in descending order

          np.arange(10,0,-1)

Out[139]: array([10,  9,  8,  7,  6,  5,  4,  3,  2,  1])

In [141]: #mention start,stop,total no of elements
          #equal spaces

          np.linspace(1,10,3)

Out[141]: array([ 1. ,  5.5, 10. ])
```

```
In [142]: np.linspace(1,10,3)

Out[142]: array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

```
In [143]: np.linspace(1,10,4)

Out[143]: array([ 1.,  4.,  7., 10.])
```

```
In [145]: np.zeros([2,3,4])

Out[145]: array([[[0., 0., 0., 0.],
                  [0., 0., 0., 0.],
                  [0., 0., 0., 0.]],

                 [[0., 0., 0., 0.],
                  [0., 0., 0., 0.],
                  [0., 0., 0., 0.]]])
```

I-8 UNIQUE ITEMS AND COUNT

```
In [155]: arr=[[1,4,5,2,2,5],
          [4,4,1,7,4,5]]
```

```
In [156]: arr

Out[156]: [[1, 4, 5, 2, 2, 5], [4, 4, 1, 7, 4, 5]]
```

```
In [157]: u_val, count= np.unique(arr, return_counts=True)
```

```
In [158]: u_val

Out[158]: array([1, 2, 4, 5, 7])
```

```
In [159]:   count
In [159]: count
```

```
Out[159]: array([2, 2, 4, 3, 1], dtype=int64)
```

2.PANDAS TUTORIAL I-9

```
In [ ]:
```

```
In [1]: #LOADING LIBRARY
        import pandas as pd
        import numpy as np
```

```
In [2]: #1.create Dataframe
        data={
            'roll_no':[3,2,7,11],
            'ppr_id':[34,21,10,11],
            'marks':[30,23,17,27]
        }
```

```
In [3]: data
```

```
Out[3]: {'roll_no': [3, 2, 7, 11],
         'ppr_id': [34, 21, 10, 11],
         'marks': [30, 23, 17, 27]}
```

```
In [4]: df1=pd.DataFrame(data)
```

```
In [5]: df1
```

Out[5]:

	roll_no	ppr_id	marks
0	3	34	30
1	2	21	23
2	7	10	17

Out[5]:

	roll_no	ppr_id	marks
0	3	34	30
1	2	21	23
2	7	10	17
3	11	11	27

In [6]:
```
#2 Setting index

df2=pd.DataFrame(data,index=['ab','ef','xy','uv'])
```

In [7]:
```
df2
```

Out[7]:

	roll_no	ppr_id	marks
ab	3	34	30
ef	2	21	23
xy	7	10	17
uv	11	11	27

In [8]:
```
#3 Extracting info
df2.loc['xy']
```

Out[8]:
```
roll_no     7
ppr_id     10
marks      17
Name: xy, dtype: int64
```

In [9]:
```
#3 Extracting info(col based,want last column data values)
df2.iloc[:,-1]
```

```
Out[9]: ab    30
        ef    23
        xy    17
        uv    27
        Name: marks, dtype: int64
```

```
In [11]: df2.iloc[0:2,2:3] #intersection of rollno and 2 column it returns
```

Out[11]:

	marks
ab	30
ef	23

I-10 working on csv file

```
In [ ]: #1loading data
        #csv file download link- https://www.kaggle.com/datasets/saurabh00007/iriscsv?resource=download
```

```
In [17]: import pandas as pd
```

```
In [28]: df = pd.read_csv('C:\\Users\\kriti\\Downloads\\archive\\Iris.csv')
```

```
In [29]: df.head()
         #print first 5 entries
```

Out[29]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [30]:
```
#for more entries
df.head(10)
```

Out[30]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [31]:
```
df.info
#shows info
```

Out[31]:
```
<bound method DataFrame.info of        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
```

Out[31]:
```
<bound method DataFrame.info of       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

In [32]:
```
#data description
df.describe()
df.describe()
```

Out[32]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [33]:

```
#data selection
```

In [39]:

```
df['SepalWidthCm'][:5]
# print first 5 columns of sepal_width
```

Out[39]:

```
0    3.5
1    3.0
2    3.2
3    3.1
4    3.6
Name: SepalWidthCm, dtype: float64
```

In [41]:

```
df[['SepalWidthCm']][:5]
#prints in form of data frame
```

Out[41]:

	SepalWidthCm
0	3.5
1	3.0
2	3.2
3	3.1
4	3.6

In [42]:

```
#two columns
df[['SepalWidthCm','PetalWidthCm']].head()
```

Out[42]:

	SepalWidthCm	PetalWidthCm
0	3.5	0.2
1	3.0	0.2
2	3.2	0.2
3	3.1	0.2
4	3.6	0.2

In [37]:

```
#till 3 but excluding 3
df.iloc[:10,1:3]
```

Out[37]:

	SepalLengthCm	SepalWidthCm
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1

3	4.6	3.1
4	5.0	3.6
5	5.4	3.9
6	4.6	3.4
7	5.0	3.4
8	4.4	2.9
9	4.9	3.1

```
In [38]: df.iloc[:10,[1,3]]
```

Out[38]:

	SepalLengthCm	PetalLengthCm
0	5.1	1.4
1	4.9	1.4
2	4.7	1.3
3	4.6	1.5
4	5.0	1.4
5	5.4	1.7
6	4.6	1.4
7	5.0	1.5
8	4.4	1.4
9	4.9	1.5

5.Missing values I-11

5.Missing values I-11

In [45]:
```
import numpy as np
data={
    'roll_no':[3,2,7,11],
    'ppr_id':[34,21,10,11],
    'marks':[np.nan,23,17,27]
}
#np.nan is null value in python
```

In [46]:
```
df1=pd.DataFrame(data)
```

In [47]:
```
df1
```

Out[47]:

	roll_no	ppr_id	marks
0	3	34	NaN
1	2	21	23.0
2	7	10	17.0
3	11	11	27.0

In [52]:
```
# to check prrescence of null value in data set
df1.isnull()
```

Out[52]:

	roll_no	ppr_id	marks
0	False	False	True
1	False	False	False
2	False	False	False
3	False	False	False

```
In [53]: df1.isnull().sum()
```

```
Out[53]: roll_no    0
         ppr_id     0
         marks      1
         dtype: int64
```

```
In [55]: #fillna()   to get rid of null values
         df2 = df1.fillna(1)
```

```
In [56]: df2
         #gets filled with 1
```

Out[56]:

	roll_no	ppr_id	marks
0	3	34	1.0
1	2	21	23.0
2	7	10	17.0
3	11	11	27.0

```
In [ ]: #dropping NULL values
```

```
In [57]: a=df1.dropna() # drop entire row
```

```
In [58]: df1
```

Out[58]:

	roll_no	ppr_id	marks
0	3	34	NaN
1	2	21	23.0
2	7	10	17.0
3	11	11	27.0

In [59]:
```
a
```

Out[59]:

	roll_no	ppr_id	marks
1	2	21	23.0
2	7	10	17.0
3	11	11	27.0

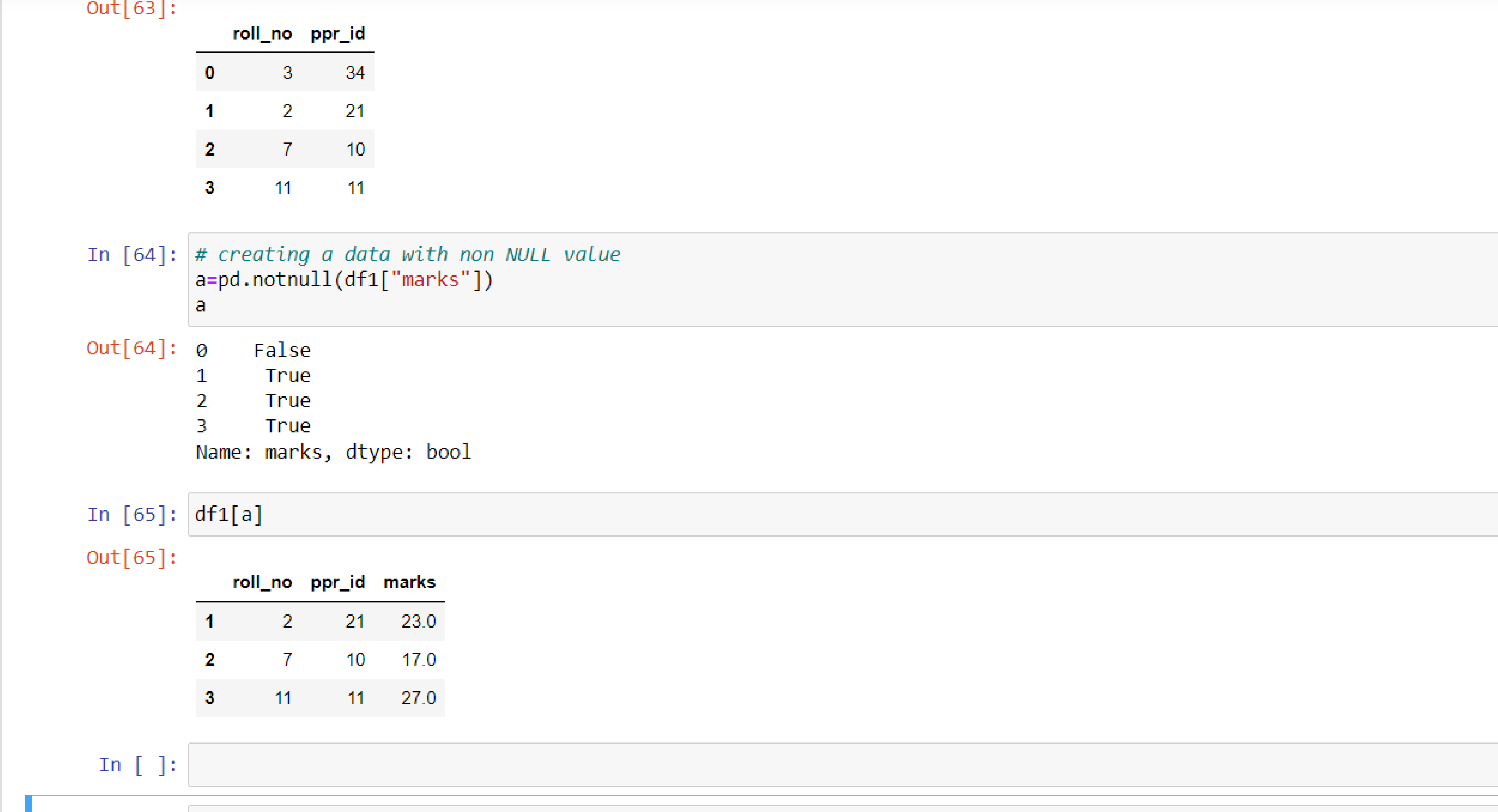
In [60]:
```
#drow rows
```

In [62]:
```
a=df1.dropna(axis=0)
a
```

Out[62]:

	roll_no	ppr_id	marks
1	2	21	23.0
2	7	10	17.0
3	11	11	27.0

In [63]:
```
# drop column
a=df1.dropna(axis=1)
a
```



Out[63]:

	roll_no	ppr_id
0	3	34
1	2	21
2	7	10
3	11	11

In [64]:
```
# creating a data with non NULL value
a=pd.notnull(df1["marks"])
a
```

Out[64]:
```
0    False
1     True
2     True
3     True
Name: marks, dtype: bool
```

In [65]:
```
df1[a]
```

Out[65]:

	roll_no	ppr_id	marks
1	2	21	23.0
2	7	10	17.0
3	11	11	27.0

In [ ]: