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

In [2]:
```
dict1 = {'Names':["Aravind","Samar",np.nan,"Siri"],
         'Age':[np.nan,21,32,43],
         'City':['Hyd','Blr','Chennai',np.nan]}
dict1
```

Out[2]:
```
{'Names': ['Aravind', 'Samar', nan, 'Siri'],
 'Age': [nan, 21, 32, 43],
 'City': ['Hyd', 'Blr', 'Chennai', nan]}
```

In [ ]:
```

```

In [23]:
```
d1=pd.DataFrame(dict1)
d1
```

Out[23]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	NaN	32.0	Chennai
3	Siri	43.0	NaN

In [6]:
```
d1.dtypes
```

Out[6]:
```
Names     object
Age      float64
City      object
dtype: object
```

In [7]:
```
dict2 = {'Names':["Aravind","Samar",None,"Siri"],
         'Age':[None,21,32,43],
         'City':['Hyd','Blr','Chennai',None]}
dict2
```

Out[7]:
```
{'Names': ['Aravind', 'Samar', None, 'Siri'],
 'Age': [None, 21, 32, 43],
 'City': ['Hyd', 'Blr', 'Chennai', None]}
```

In [8]:
```
d2=pd.DataFrame(dict2)
d2
```

Out[8]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	None	32.0	Chennai
3	Siri	43.0	None

In [9]:
```
d1.isnull()
```

Out[9]:

	Names	Age	City
0	False	True	False
1	False	False	False
2	True	False	False
3	False	False	True

In [14]:
```
d1.isnull().sum()
```

Out[14]:
```
Names    1
Age      1
City     1
dtype: int64
```

In [ ]:
```
# nan - not a number
```

$Method - 1$

Fill with random values

Method name : fill na

In [13]:
```
d1.isnull().sum()*100/len(d1)
```

Out[13]:
```
Names    25.0
Age      25.0
City     25.0
dtype: float64
```

In [15]:
```
d1.fillna(40)
```

Out[15]:

	Names	Age	City
0	Aravind	40.0	Hyd
1	Samar	21.0	Blr
2	40	32.0	Chennai
3	Siri	43.0	40

$Method - 2$

We can fill values WRT columns also

In [17]:
```
d1['Age'].fillna(40)   # it wont update
```

Out[17]:
```
0    40.0
1    21.0
2    32.0
3    43.0
Name: Age, dtype: float64
```

In [18]:
```
d1    # bcz inplace is default false
```

Out[18]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	NaN	32.0	Chennai
3	Siri	43.0	NaN

In [19]:
```
d1['Age'].fillna(40,inplace=True)
```

In [20]:
```
d1
```

Out[20]:

	Names	Age	City
0	Aravind	40.0	Hyd
1	Samar	21.0	Blr
2	NaN	32.0	Chennai
3	Siri	43.0	NaN

In [21]:
```
d1['Names'].fillna('Anuja',inplace=True)
d1
```

Out[21]:

	Names	Age	City
0	Aravind	40.0	Hyd
1	Samar	21.0	Blr
2	Anuja	32.0	Chennai
3	Siri	43.0	NaN

In [22]:
```
d1['City'].fillna('Pune',inplace=True)
d1
```

Out[22]:

	Names	Age	City
0	Aravind	40.0	Hyd
1	Samar	21.0	Blr
2	Anuja	32.0	Chennai
3	Siri	43.0	Pune

$Method - 3$

- bfill
- ffill
- pad
- backfill

In [24]:
```
d1=pd.DataFrame(dict1)
d1
```

Out[24]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	NaN	32.0	Chennai
3	Siri	43.0	NaN

In [31]:
```
d1.fillna(method='bfill')
```

```
C:\Users\Anuja_PC\AppData\Local\Temp\ipykernel_5356\1914124427.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future versio
n. Use obj.ffill() or obj.bfill() instead.
  d1.fillna(method='bfill')
```

Out[31]:

	Names	Age	City
0	Aravind	21.0	Hyd
1	Samar	21.0	Blr
2	Siri	32.0	Chennai
3	Siri	43.0	NaN

- Names - index 3 is missing , it is filled with index 2 value
- Age - index 0 is missing , it is filled with index 1
- City - index 3 is missing, no next value to fill with

In [26]:
```
d1.fillna(method='ffill')
```

```
C:\Users\Anuja_PC\AppData\Local\Temp\ipykernel_5356\4088926743.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future versio
n. Use obj.ffill() or obj.bfill() instead.
  d1.fillna(method='ffill')
```

Out[26]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	Samar	32.0	Chennai
3	Siri	43.0	Chennai

In [27]:
```
d1.fillna(method='pad')
```

```
C:\Users\Anuja_PC\AppData\Local\Temp\ipykernel_5356\3282249208.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future versio
n. Use obj.ffill() or obj.bfill() instead.
  d1.fillna(method='pad')
```

Out[27]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	Samar	32.0	Chennai
3	Siri	43.0	Chennai

In [28]:
```
d1.fillna(method='backfill')
```

```
C:\Users\Anuja_PC\AppData\Local\Temp\ipykernel_5356\907203477.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future versio
n. Use obj.ffill() or obj.bfill() instead.
  d1.fillna(method='backfill')
```

Out[28]:

	Names	Age	City
0	Aravind	21.0	Hyd
1	Samar	21.0	Blr
2	Siri	32.0	Chennai
3	Siri	43.0	NaN

In [32]:
```
d1.fillna(method='bfill', axis=1) # replacing with corresponding column,
```

```
C:\Users\Anuja_PC\AppData\Local\Temp\ipykernel_5356\877457416.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future versio
n. Use obj.ffill() or obj.bfill() instead.
  d1.fillna(method='bfill', axis=1)
```

Out[32]:

	Names	Age	City
0	Aravind	Hyd	Hyd
1	Samar	21.0	Blr
2	32.0	32.0	Chennai
3	Siri	43.0	NaN

In [33]:
```
d1.fillna(method='pad', axis=1)
```

```
C:\Users\Anuja_PC\AppData\Local\Temp\ipykernel_5356\3030213872.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future versio
n. Use obj.ffill() or obj.bfill() instead.
  d1.fillna(method='pad', axis=1)
```

Out[33]:

	Names	Age	City
0	Aravind	Aravind	Hyd
1	Samar	21.0	Blr
2	NaN	32.0	Chennai
3	Siri	43.0	43.0

- backfill and bfill - fill with next vale
- pad and ffill fill with prev value
- but it depends on axis

In [34]:
```
d1.fillna(method='backfill', axis=1)
```

```
C:\Users\Anuja_PC\AppData\Local\Temp\ipykernel_5356\1171026241.py:1: FutureWarnin
g: DataFrame.fillna with 'method' is deprecated and will raise in a future versio
n. Use obj.ffill() or obj.bfill() instead.
  d1.fillna(method='backfill', axis=1)
```

Out[34]:

	Names	Age	City
0	Aravind	Hyd	Hyd
1	Samar	21.0	Blr
2	32.0	32.0	Chennai
3	Siri	43.0	NaN

$Method - 4$

- Mean - Numerical vaue filled by mean value , but mean is affected by outliers, if we dont have outliers it is best one'
- Median - Numerical values can fill with median value, we know that it doesnt affect with outliers, so if utliers are there we can go with median
- Mode - Mode is useful for categorical data**

In [35]:
```
dict1 = {'Names':["Aravind","Samar",np.nan,"Siri"],
         'Age':[np.nan,21,32,43],
         'City':['Hyd','Blr','Chennai',np.nan]}
dict1
```

Out[35]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	Anuja	32.0	Chennai
3	Siri	43.0	NaN

In [36]:
```
d1=pd.DataFrame(dict1)
d1
```

Out[36]:

	Names	Age	City
0	Aravind	NaN	Hyd
1	Samar	21.0	Blr
2	NaN	32.0	Chennai
3	Siri	43.0	NaN

In [38]:
```
age_mean = d1['Age'].mean()

d1['Age'].fillna(age_mean)
```

Out[38]:
```
0    32.0
1    21.0
2    32.0
3    43.0
Name: Age, dtype: float64
```

In [39]:
```
age_median = d1['Age'].median()

d1['Age'].fillna(age_median)
```

Out[39]:
```
0    32.0
1    21.0
2    32.0
3    43.0
Name: Age, dtype: float64
```

In [40]:
```
age_mode = d1['Age'].mode()

d1['Age'].fillna(age_mode)
```

Out[40]:
```
0    21.0
1    21.0
2    32.0
3    43.0
Name: Age, dtype: float64
```

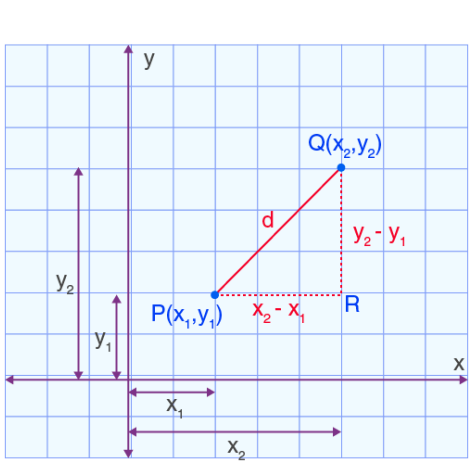
In [41]:
```
d1['City'].mode()
```

Out[41]:
```
0        Blr
1    Chennai
2        Hyd
Name: City, dtype: object
```

$Method - 5$

KNN imputer - K nearest neibour [applicable to numeric data]

- K is a hyper parameter means user can choose
- it is a distance metric : Euclidian distance - [distance between 2 points]



- KNN imputer takes mean of the neighbour values,
- the neibours value can provided by using value=k , if k=2 , take 2 values mean
- It is under sklearn package
- under sklearn we have impute method
- under imputer we have KNN imputer

In [42]:
```
from sklearn.impute import KNNImputer

knn_data= KNNImputer()

knn_data.fit_transform(d1[['Age']])
```

Out[42]:
```
array([[32.],
       [21.],
       [32.],
       [43.]])
```