

# pandas introduction.....

```
In [1]: #pip install pandas
```

```
In [2]: import pandas as pd
```

```
In [3]: print(pd.__version__)
```

2.1.3

## pandas series

a pandas series is like a column in a table it is 1-D array which holds data of any series.

```
In [4]: x1=[1,7,2]  
pd.Series(x1)
```

```
Out[4]: 0    1  
        1    7  
        2    2  
dtype: int64
```

### labelling

-label can be use to access a spcified values

```
In [5]: x1=[1,7,2]  
pd.Series(x1)[0]
```

```
Out[5]: 1
```

```
In [6]: x1=[1,7,2]  
pd.Series(x1)[2]
```

```
Out[6]: 2
```

with create label yo can create your own name labeles

```
In [7]: x1=[1,7,2]  
pd.Series(x1,index=["x","y","z"] )
```

```
Out[7]: x    1  
        y    7  
        z    2  
dtype: int64
```

label can be use to access a spcified values .(after creating own lables)

```
In [43]: x1=[1,7,2]
x2=pd.Series(x1,index=["x","y","z"] )
```

```
In [44]: x2["x"]
```

```
Out[44]: 1
```

you can also use a key or values object like a dictionary a series

here we will create a simple pandas series from a dictionary

```
In [46]: cal={"day1":1,"day2":2,"day3":3}
```

```
In [47]: pd.Series(cal)
```

```
Out[47]: day1    1
day2    2
day3    3
dtype: int64
```

now we will create a series using only data from day1 and day 2

```
In [ ]: cal={"day1":1,"day2":2,"day3":3}
```

```
In [50]: result=pd.Series(cal,index=["day1","day3"])
```

```
In [51]: result
```

```
Out[51]: day1    1
day3    3
dtype: int64
```

## DataFrame

data sets in pandas are usually multidimensional tables and they are called the DataFrames

series are like columns and DataFrame is the whole tables.

we will now create a dataframe from 2 series

```
In [61]: x1={"cal":[220,390,540],"duration":[50,40,45]}
```

```
In [62]: pd.DataFrame(x1)
```

```
Out[62]:
```

	cal	duration
0	220	50
1	390	40
2	540	45

locate row pandas use the loc attribute to return one or more specified row

```
In [8]: x1={"cal":[220,390,540],"duration":[50,40,45]}
```

```
In [9]: x1
```

```
Out[9]: {'cal': [220, 390, 540], 'duration': [50, 40, 45]}
```

```
In [15]: pd.DataFrame(x1)
```

```
Out[15]:
```

	cal	duration
0	220	50
1	390	40
2	540	45

```
In [25]: df=pd.DataFrame(x1)
```

```
In [27]: df
```

```
Out[27]:
```

	cal	duration
0	220	50
1	390	40
2	540	45

```
In [28]: df.loc[0]
```

```
Out[28]: cal      220  
duration    50  
Name: 0, dtype: int64
```

ex of returning row 0 and 1:

```
In [30]: df.loc[[0,1]]
```

```
Out[30]:
```

	cal	duration
0	220	50
1	390	40

name index: with the index arg, you can name your own index.

```
In [37]: df=pd.DataFrame(x1,index=["Day1","Day2","Day3"])
```

```
In [38]: df
```

```
Out[38]:
```

	cal	duration
Day1	220	50
Day2	390	40
Day3	540	45

locate the name index:-

```
In [47]: df.loc[["Day1","Day2"]]
```

```
Out[47]:
```

	cal	duration
Day1	220	50
Day2	390	40

## pandas\_csv

Load the data from the csv file in the dataframe i.e data.csv

csv(comma seperated file)

it is a simple way to store the big and biggest data sets. csv files contains plain text..

fileload=pd.read\_csv("")#path...

loading the csv into dataframe

```
In [59]: df=pd.read_csv(r'C:\Users\arnak\OneDrive\Pictures\sample1.csv')
```

```
In [60]: df
```

```
Out[60]:
```

	Name	age	Experience	Salary
0	jack	31	10	30000
1	alex	30	8	25000
2	caroline	29	4	20000
3	paul	24	3	20000
4	sandra	21	1	15000
5	casandra	23	2	18000

max rows:- you can check your system's maximum rows with :

```
In [67]: pd.options.display.max_rows
```

```
Out[67]: 60
```

yes we can increase the maximum number of rows to display the entire data...

example:- #pd.options.display.max\_rows=9999

## pandas Analyzing

viewing the data -: one of the most used method for a quick overview of the dataframe is the head() method . this methd specified number of rows

```
In [69]: df.head()
```

```
Out[69]:
```

	Name	age	Experience	Salary
0	jack	31	10	30000
1	alex	30	8	25000
2	caroline	29	4	20000
3	paul	24	3	20000
4	sandra	21	1	15000

here we will print the first 3 rows in the dataframe

```
In [73]: df.head(3)
```

```
Out[73]:
```

	Name	age	Experience	Salary
0	jack	31	10	30000
1	alex	30	8	25000
2	caroline	29	4	20000

here we will print the last 2 rows in the dataframe

```
In [74]: df.tail(2)
```

```
Out[74]:
```

	Name	age	Experience	Salary
4	sandra	21	1	15000
5	casandra	23	2	18000

what if you want the information about the data in the dataframe: via info()...

```
In [77]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Name        6 non-null     object
1   age         6 non-null     int64
2   Experience   6 non-null     int64
3   Salary      6 non-null     int64
dtypes: int64(3), object(1)
memory usage: 324.0+ bytes
```

## pandas cleaning data

cleaning data-: it means fixing the bad data in your dataset.

1]empty cell 2] data in wrong formate 3] duplicate dat 4]wrong data

Empty cell..-: it will give you wrong result always,we will have remove the rows always that contain the bad data

```
In [109... data=pd.read_csv(r"C:\Users\arnak\Downloads\sample4 - sample4.csv.csv")
```

```
In [110... data
```

Out[110]...

	Name	Departments	salary
0	jack	Data Science	10000.0
1	alex	ML	NaN
2	caroline	NaN	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	NaN	AI	10000.0
7	kathy	ML	NaN
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

here we will return a new data fream with no empty cell..

```
In [6]: data_1=data.dropna()
```

```
In [7]: data_1
```

Out[7]:

	Name	Departments	salary
0	jack	Data Science	10000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

if add any case you want to change the original dataframe , then use the inplace=true agument. it will remove the row containing the NULL(NaN ) values.

```
In [8]: data.dropna(inplace=True)
```

```
In [9]: data
```

```
Out[9]:
```

	<b>Name</b>	<b>Departments</b>	<b>salary</b>
<b>0</b>	jack	Data Science	10000.0
<b>3</b>	jack	AI	4000.0
<b>4</b>	sandra	Data Science	3000.0
<b>5</b>	jack	NLP	20000.0
<b>8</b>	andrew	Data Science	10000.0
<b>9</b>	jack	NLP	2000.0

replacing the empty value : we will use yhe fillna() method which will allow us to replace the empty cell with a value.

```
In [14]: data
```

```
Out[14]:
```

	<b>Name</b>	<b>Departments</b>	<b>salary</b>
<b>0</b>	jack	Data Science	10000.0
<b>1</b>	alex	ML	NaN
<b>2</b>	caroline	NaN	4000.0
<b>3</b>	jack	AI	4000.0
<b>4</b>	sandra	Data Science	3000.0
<b>5</b>	jack	NLP	20000.0
<b>6</b>	NaN	AI	10000.0
<b>7</b>	kathy	ML	NaN
<b>8</b>	andrew	Data Science	10000.0
<b>9</b>	jack	NLP	2000.0

```
In [49]: data.fillna(130,inplace=True)
```

```
In [50]: data
```



Out[50]:

	Name	Departments	salary
0	jack	Data Science	10000.0
1	alex	ML	130.0
2	caroline	130	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	130	AI	10000.0
7	kathy	ML	130.0
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

To replace only the empty value for one col, you need to specify the column name

In [52]: `data["salary"].fillna(130,inplace=True)`

In [53]: `data`

Out[53]:

	Name	Departments	salary
0	jack	Data Science	10000.0
1	alex	ML	130.0
2	caroline	NaN	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	NaN	AI	10000.0
7	kathy	ML	130.0
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

here we can also replace the empty cell using `mean()`, `median()`, `mode()`.

calculate the MEAN and replace the empty values with it..

In [60]: `data_2=data`

```
In [61]: data_2
```

```
Out[61]:
```

	Name	Departments	salary
--	------	-------------	--------

0	jack	Data Science	10000.0
1	alex	ML	NaN
2	caroline	NaN	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	NaN	AI	10000.0
7	kathy	ML	NaN
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

```
In [90]: data["salary"].mean()
```

```
Out[90]: 7875.0
```

```
In [91]: data_2=data["salary"].mean()
```

```
In [92]: data["salary"].fillna(data_2,inplace=True)
```

```
In [93]: data
```

```
Out[93]:
```

	Name	Departments	salary
--	------	-------------	--------

0	jack	Data Science	10000.0
1	alex	ML	7875.0
2	caroline	NaN	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	NaN	AI	10000.0
7	kathy	ML	7875.0
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

In [ ]:

In [95]: data

Out[95]:

	Name	Departments	salary
0	jack	Data Science	10000.0
1	alex	ML	7875.0
2	caroline	NaN	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	NaN	AI	10000.0
7	kathy	ML	7875.0
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

calculate the MEDIAN AND REPLACE ANY EMPTY VALUES IN IT..

In [111... data["salary"].median()

Out[111... 7000.0

In [112... data\_3=data["salary"].median()

In [113... data["salary"].fillna(data\_3,inplace=True)

In [114... data

Out[114...

	Name	Departments	salary
0	jack	Data Science	10000.0
1	alex	ML	7000.0
2	caroline	NaN	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	NaN	AI	10000.0
7	kathy	ML	7000.0
8	andrew	Data Science	10000.0
9	jack	NLP	2000.0

## Pandas Removing Duplicates

In [138...

```
data=pd.read_csv(r"C:\Users\arnak\Downloads\sample4 - sample4.csv (2).csv")
```

In [139...

```
data
```

Out[139...

	Name	Departments	salary
0	jack	Data Science	10000.0
1	alex	ML	NaN
2	caroline	NaN	4000.0
3	jack	AI	4000.0
4	sandra	Data Science	3000.0
5	jack	NLP	20000.0
6	NaN	AI	10000.0
7	jack	AI	4000.0
8	kathy	ML	NaN
9	andrew	Data Science	10000.0
10	jack	NLP	2000.0
11	jack	Data Science	10000.0

first you need to discover the duplicate values via duplicate() method.

return true for every row that is duplicate otherwise return false

```
In [141...] data.duplicated()
```

```
Out[141...] 0    False
            1    False
            2    False
            3    False
            4    False
            5    False
            6    False
            7     True
            8    False
            9    False
           10    False
           11     True
           dtype: bool
```

removing the duplicate from the data set. via drop\_duplicates()

```
In [143...] data.drop_duplicates(inplace=True)
```

```
In [144...] data
```

```
Out[144...]   Name Departments  salary
0    jack    Data Science  10000.0
1    alex           ML      NaN
2  caroline           NaN   4000.0
3    jack           AI   4000.0
4  sandra    Data Science   3000.0
5    jack           NLP  20000.0
6    NaN           AI  10000.0
8    kathy           ML      NaN
9  andrew    Data Science  10000.0
10   jack           NLP   2000.0
```

```
In [ ]:
```