Pandas

- It is an open source data analysis library written in python.
- It leverages power and speed of numpy to make data analysis and preprocessing. ***NumPy*** brings the computational power of languages like C and Fortran to Python, a language much easier to learn and use.
- It provides rich and highly robust data operations

Pandas has two types of data structures :

1. Series : It's a 1D array with indexes, it stores a single column or row of data in Dataframe.
2. Dataframe : It's a tabular spreadsheet like structure representing rows each of which contains one or multiple columns.

General functions — pandas 2.0.2 documentation

https://pandas.pydata.org/docs/reference/general_functions.html

```
import pandas as pd
import numpy as np

dict1={
"name":'[abc','xyz'],"marks":[20,67]
}

df=pd.DataFrame(dict1)
print(df)
```

To export dataframe to csv

```
df.to_csv('data.csv')
```

To remove the indexes of rows

```
df.to_csv('data.csv',index=False)
```

To see upper and lower respectively two or three or i rows

```
df.head(2)
df.tail(2)
```

Perform mathematical operations on numerical columns of csv (statiscal analysis)

```
df.describe()
```

To read data from a csv file

```
var_name= pd.read_csv('file_name.csv')
print(var_name)
```

To edit data in a csv

```
harry['speed'][0] = 40
harry
```

To write the changes to the csv

```
harry.to_csv('file_name.csv')
```

To change the index to another value, To access rows we have indexes and to access columns we have column name

```
harry.index = [’first’ ,’second’ , ‘third’ , ‘fourth’]
harry
```

Series:

```
#Generate 34 random number series
ser = pd.Series(np.random.rand(34)
ser
#print type of ser will print pandas.core.series.Series
type(ser)
```

Dataframe:

```
newdf = pd.DataFrame(np.random.rand(334,5),index=np.arange(334))
newdf.head()
type(newdf) #prints pandas.core.frame.DataFrame
newdf.descirbe()
#to print datatypes of columns
newdf.dtypes #strings have object as dtype
newdf.index()
newdf.columns()
```

To convert a dataframe into a numpy array

```
newdf.to_numpy()
```

To take transpose of the dataframe

```
newdf.T
newdf
```

To sort index in reverse order, for index sorting axis=0 , and for column sorting axis=1.

```
newdf.sort_index(axis=0, ascending=False)
```

A view (newdf2) : any changes in newdf2 will be visible in newdf too. Just like a pointer

```
newdf2=newdf
newdf2[0][0]= 1234
newdf
```

A copy

```
newdf3 = newdf.copy()
newdf3[0][0]= 97835
newdf
```

while chaining, it’s not guranteed that view will be returned or a copy will be returned. Pandas acc to it’s internal memory management decides whether to return a view or a copy. So, while setting a value we set it using loc function

```
newdf.loc[0,0] = 654
```

To drop a column

```
newdf.drop(column_name,axis=1)
```

To get specific rows and columns respectively. It will just print and will not make any changes to previous dataframe. It returns a copy

```
newdf.loc[[1,2],['c','d']]
#for all columns
newdf.loc[:,['c','d']]
#for all rows
newdf.loc[[1,2],:]
```

Complex Queries

```
#To use names us loc
newdf.loc[(newdf['A']<0.3) && (newdf['c']>0.1)]
```

```
#to use index use iloc
newdf.iloc[0,4]
```