DAY-2

Pandas is defined as an open-source library that provides high-performance data manipulation in python.
Data analytics requires lot of processing such as restructuring, cleaning, mergin, manipulating etc...We prefer Pandas to perform above functionalities coz it is fast, simple than other tools. Pandas is built on Numpy, Numpy is required for operating pandas.
Pandas Series is a data structure with one dimentional labelled array. It is a primary building block of Dataframe making its rows and columns.

```
import numpy as np
labels = ['a','b','c']
my_data = [10,20,30]
arr = np.array(my_data)
d = {'a':100,'b':200,'c':300}
```

#Syntax --> pandas.Series(data=None, index=None, dtype=None, name=None, copy=True or False)
#E.g-1

```
import pandas as pd
pd.Series(my_data)
0    10
1    20
2    30

dtype: int64

type(pd.Series(my_data))
```

```
pandas.core.series.Series
```

#Series with labels

```
pd.Series(data=my_data, index=labels)
#          or
pd.Series(my_data, labels)
```

```
a    10
b    20
c    30
dtype: int64
```

```
pd.Series(data=[print, len, sum])
```

```
0    <built-in function print>
1      <built-in function len>
2      <built-in function sum>
dtype: object
```

#decorator: Pandas does provide a concept called "dec orators" in the form of DataFrame/Series accessor fun ctions.

Accessors in Pandas are used to provide access to addi tional methods and attributes that are not directly ava ilable on DataFrame or Series objects. They allow you to extend the functionality of Pandas objects by defini ng custom methods that can be accessed through a spe cial syntax.

```
def decor(func):
   def inner():
      str1 = func()
```

```
        return str1.upper()
    return inner

@decor
def greet():
    return "good morning"
print(greet())
```

```
GOOD MORNING
```

```
ser1 = pd.Series([1,2,3,4,5],["USA","India","Canada","UK","
Egypt"])
ser1
```

```
USA        1
India      2
Canada     3
UK         4
Egypt      5
dtype: int64
```

#access values using index

```
ser1[0:3]
```

```
USA        1
India      2
Canada     3
dtype: int64
```

```
ser2 = pd.Series([5,6,7,8],["USA","Brazil","Canada","UK"])
ser1+ser2
```

```
Brazil      NaN
Canada     10.0
```

```
Egypt       NaN
India       NaN
UK         12.0
USA         6.0
dtype: float64
```

 # To create a new data in series

```
ser2['china'] = 'duplicate'
ser2
```

```
USA                 5
Brazil              6
Canada              7
UK                  8
china       duplicate
dtype: object
```

#Dataframe: DataFrame is a two-dimensional labeled data structure that is widely used for data manipulation and analysis. It is essentially a table with rows and columns, where each column can have a different data type (e.g., integer, float, string, etc.).
#pd.DataFrame(datas, row_label, col_label)

```
df = pd.DataFrame(np.random.randn(5,4))
df
```

	0	1	2	3
0	-0.743617	0.043163	1.257809	-0.441337
1	-0.634113	-0.099980	2.282786	-0.816584
2	0.166242	-0.432571	-1.512663	-0.235004

	0	1	2	3
3	-0.695917	1.006487	0.154597	-1.137518
4	0.715899	0.261169	-0.065419	-1.069397

```
df = pd.DataFrame(np.random.randn(5,4), ['A','B','C','D','E'],['w','x','y','z'])
df
```

	w	x	y	z
A	-1.390346	-0.617943	-0.235332	1.834664
B	0.776284	-2.006753	-1.595141	0.525834
C	0.044868	-0.131634	-0.026057	-0.599927
D	-1.316303	0.486794	-0.961434	-1.379597
E	0.456750	-1.923114	-0.885952	0.328777

 # To convert dictionary to DataFrame

```
d = {"col1":[1,2], "col2":[3,40], "col3":[5,6]}
print(d)
df = pd.DataFrame(d, ['row1','row2'])
print("\n",df)
```

```
{'col1': [1, 2], 'col2': [3, 40], 'col3': [5, 6]}

        col1  col2  col3
row1     1     3     5
row2     2    40     6
```

#Transpose
df.T

	A	B	C	D	E
w	-1.390346	0.776284	0.044868	-1.316303	0.456750
x	-0.617943	-2.006753	-0.131634	0.486794	-1.923114
y	-0.235332	-1.595141	-0.026057	-0.961434	-0.885952
z	1.834664	0.525834	-0.599927	-1.379597	0.328777

df.index **#-->get all the row names**
df.columns
type(df)
df.dtypes
df.info()
df.values
df.axes
df.ndim
df.size

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, A to E
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   w       5 non-null      float64
 1   x       5 non-null      float64
```

```
 2    y          5 non-null      float64
 3    z          5 non-null      float64
dtypes: float64(4)
memory usage: 372.0+ bytes
20
```

 # To access specific column in df

```
df['w']
type(df['w'])
```

#to access multiple columns

```
df[['w','x','y']]
```

	w	x	y
A	-1.390346	-0.617943	-0.235332
B	0.776284	-2.006753	-1.595141
C	0.044868	-0.131634	-0.026057
D	-1.316303	0.486794	-0.961434
E	0.456750	-1.923114	-0.885952

#to create new column

```
df['new'] = df['w']+df['y']
df
```

	w	x	y	z	new
A	-1.390346	-0.617943	-0.235332	1.834664	-1.625679
B	0.776284	-2.006753	-1.595141	0.525834	-0.818857

	w	x	y	z	new
C	0.044868	-0.131634	-0.026057	-0.599927	0.018811
D	-1.316303	0.486794	-0.961434	-1.379597	-2.277737
E	0.456750	-1.923114	-0.885952	0.328777	-0.429202

#to remove new column

```
df.drop('new', axis=1, inplace=True)
df
```

	w	x	y	z
A	-1.390346	-0.617943	-0.235332	1.834664
B	0.776284	-2.006753	-1.595141	0.525834
C	0.044868	-0.131634	-0.026057	-0.599927
D	-1.316303	0.486794	-0.961434	-1.379597
E	0.456750	-1.923114	-0.885952	0.328777

K. Tanuja

22A81A0623