Kmit

SONET

Data Science

(Level-I)

Ajeet K. Jain, M. Narsimlu

(ML TEAM)- SONET, KMIT, Hyderabad

Session - 20

This session deals with

Introduction to Pandas

Types of Pandas Data Structures

pandas Attributes and methods

Exercises on Numpy and Linear algebra

Introduction

Powerful and productive Python data analysis and management library

Rich data structures and functions to make working with structured data fast, easy, and expressive

Flexible data manipulation capabilities of **spreadsheets and relational databases**

Sophisticated indexing functionality

slice, dice, perform aggregations, select subsets of data

The ideal tool for Data Scientist

Munging data

Cleaning data

Analyzing data

Modeling data

Organizing the results of the analysis into a form suitable for plotting or tabular displa

Types Of Pandas Data Structure

Pandas deals with the following three data structures −

1.Series
2.DataFrame
3. Panel

Dimension & Description:

Data Structure	Dimensions	Description
Series	1	1D labeled homogeneous array, sizeimmutable.
Data Frames	2	General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns.
Panel	3	General 3D labeled, size-mutable array.

Series

Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56, …

10	14	15	25	30	45	55
Data						
Index	**0**	**1**	**2**	**3**	**4**	**5**

6

pandas. Series(data, index, d

```
import pandas as pd
s=pd.Series((10,14,15,25,30,45,55))
print(s)
```

```
0     10
1     14
2     15
3     25
4     30
5     45
6     55
dtype: int64
```

Key Points :

Homogeneous data

Size Immutable

Values of Data Mutable

Series

Attributes:

size
axes
dtypes
at
values
shape
ftypes
Loc
etc…

```
In [44]: import pandas as pd
         s=pd.Series((10,14,15,25,30,45,55),copy=False, dtype=float)
         print(s.size)
         print(s.axes)
         print(s.dtypes)
         print(s.at)
         print(s.values)
         print(s.shape)
         print(s.loc[:1])
         print(s.ftypes)
```

```
7
[RangeIndex(start=0, stop=7, step=1)]
float64
<pandas.core.indexing._AtIndexer object at 0x000001E040C67138>
[10. 14. 15. 25. 30. 45. 55.]
(7,)
0    10.0
1    14.0
dtype: float64
float64:dense
```

Series

Reindexing labels: changing indexing as per our choice

```
In [18]: import pandas as pd
         s=pd.Series((10,5,7),index=['a','b','c'])
         print(s.index)
```

```
Index(['a', 'b', 'c'], dtype='object')
```

```
In [16]: print(s.reindex(['c','b','a']))
```

```
c    -0.083656
b    -0.601870
a     0.051397
dtype: float64
```

Series

DATA SCIENCE

Methods:

abs()
add()
add_suffix()

```
import pandas as pd
s=pd.Series((10,14,15,25.674,30.45,55,55))
s1=pd.Series((10,14,15,None,30.45,55,55))
s2=pd.Series(('kmit','ngit','kmes'))
print(s.abs())
print(s.add(s1))
print(s2.add_suffix(1))
```

```
0     10.000
1     14.000
2     15.000
3     25.674
4     30.450
5     55.000
6     55.000
dtype: float64
0      20.0
1      28.0
2      30.0
3       NaN
4      60.9
5     110.0
6     110.0
dtype: float64
01     kmit
11     ngit
21     kmes
dtype: object
```

Series

Methods:

append()

dot()

```
In [13]: import pandas as pd
         s=pd.Series((10,5,5))
         s1=pd.Series((10,5,6))
         s2=pd.Series(('kmit','ngit','kmes'))
         print(s.append(s1))
         print(s1.dot(s))
```

```
0    10
1     5
2     5
0    10
1     5
2     6
dtype: int64
155
```

Series

Methods:

mean()

mad()

sort_values()

pop()

std()

```
import pandas as pd
s=pd.Series((10,5,7))
s1=pd.Series((10,5,6))
s2=pd.Series(('kmit','ngit','kmes'))
print(s.mean())
print(s.mad())
print(s.sort_values())
print(s.pop(2))
print(s.std())
```

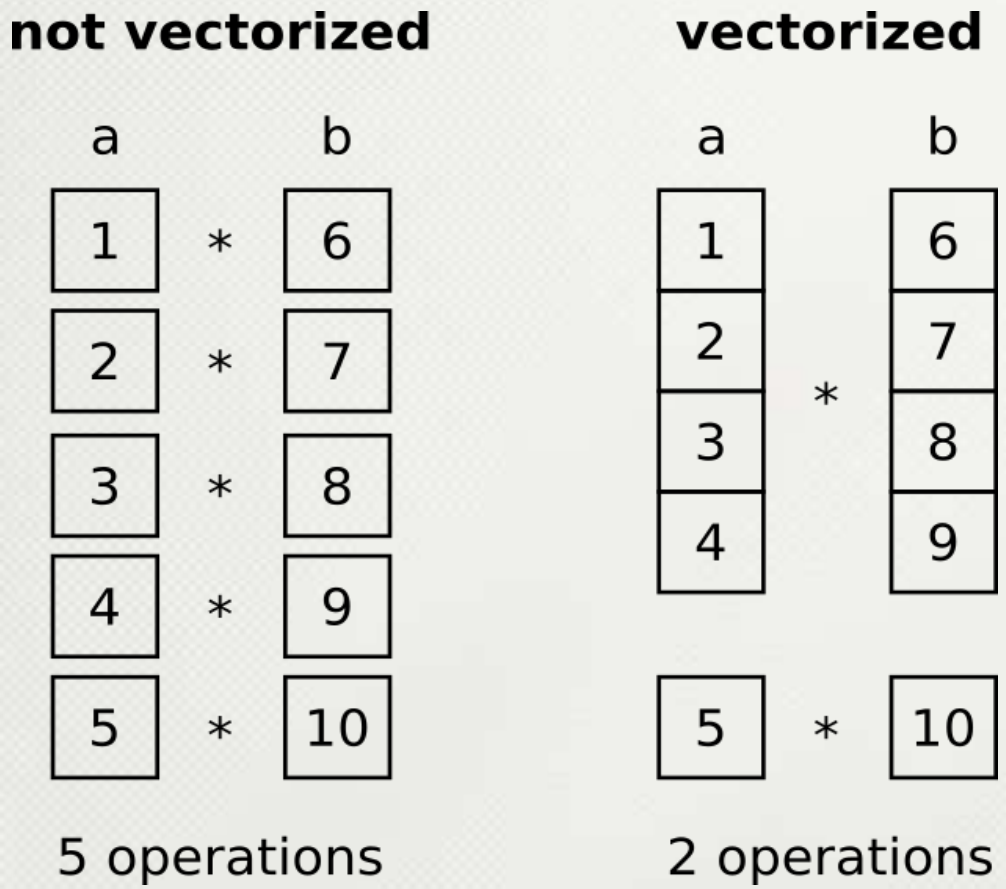
```
7.333333333333333
1.7777777777777777
1     5
2     7
0    10
dtype: int64
7
3.5355339059327378
```

Vectorization

The process of rewriting a loop so that instead of processing a single element of an array N times.

```
import pandas as pd
a=pd.Series((1,2,3,4,5))
b=pd.Series((6,7,8,9,10))
print(a.add(b))
print(a+b)
```

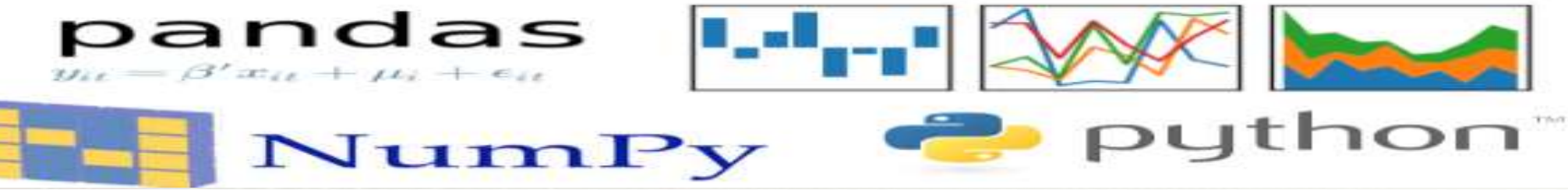
```
0     7
1     9
2    11
3    13
4    15
dtype: int64
0     7
1     9
2    11
3    13
4    15
dtype: int64
```



SONET

Pandas Data Types

DATA SCIENCE



Pandas dtype	Python type	NumPy type	Usage
object	str	string_, unicode_	Text
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	float	float_, float16, float32, float64	Floating point numbers
bool	bool	bool_	True/False values
datetime	datetime64	datetime64	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

DataFrame

Two-dimensional tabular data structure

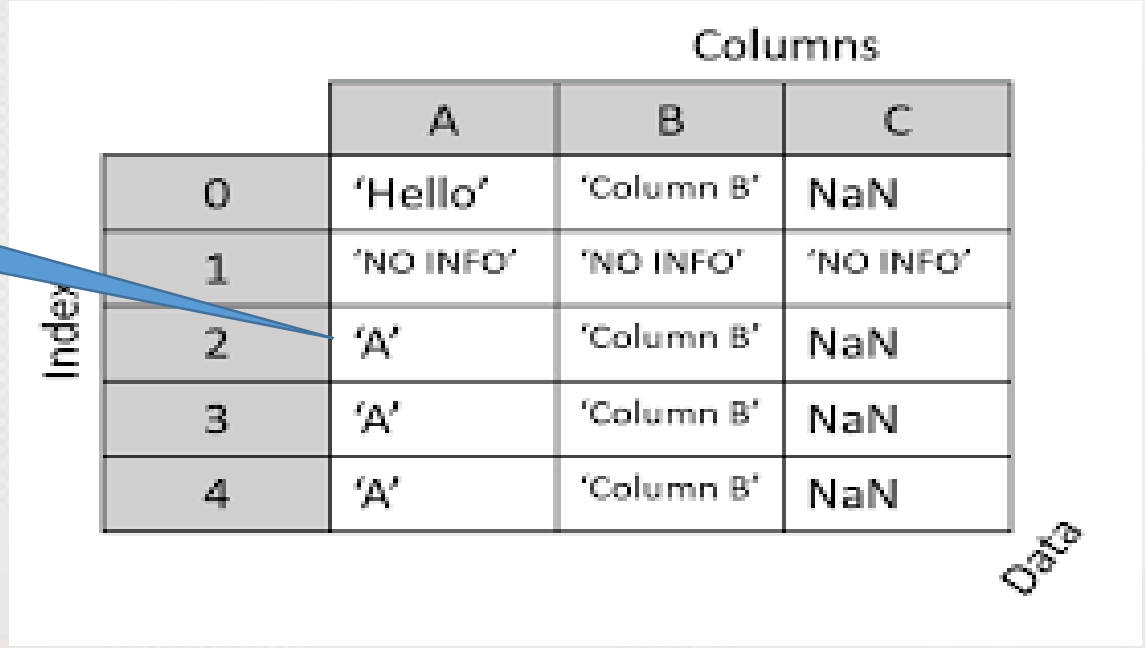
Data manipulation with integrated indexing

Support heterogeneous columns and Homogeneous columns

A pandas DataFrame→pandas. DataFrame(data, index, columns, dtype)

Data : ndarray\series\map\lists\dict\ DataFrame

Dtype:-Data type of each column
Size:- Mutable



Creating DataFrame

DATA SCIENCE

```
In [21]: #dataframe
         data={'one':[1,2,3],'two':[4,5,6]}
         d=pd.DataFrame(data)
         print(d)
```

```
   one  two
0    1    4
1    2    5
2    3    6
```

Adding label

```
In [22]: d=pd.DataFrame(data,index=['a','b','c'])
         print(d)
```

```
   one  two
a    1    4
b    2    5
c    3    6
```

add column to DataFrame

```
In [23]: d['three']=[7,8,9]
         print(d)
```

```
   one  two  three
a    1    4      7
b    2    5      8
c    3    6      9
```

Select row by label

```
In [25]: row=d.xs('a')
         print(row)
```

```
one      1
two      4
three    7
Name: a, dtype: int64
```

DataFrame

Attributes:

```
In [72]: import pandas as pd
         data={'one':[1,2,3],'two':[4,5,6]}
         df=pd.DataFrame(data)
         print(df)
         print(df.values)
         print(df.shape)
         print(df.size)
         print(df.dtypes)
         print(df.index)
         print(df.iat[0,1])
         print(df.columns)
         print(df.T)
         print(df.empty)
```

```
   one  two
0    1    4
1    2    5
2    3    6
[[1 4]
 [2 5]
 [3 6]]
(3, 2)
6
one     int64
two     int64
dtype: object
RangeIndex(start=0, stop=3, step=1)
4
Index(['one', 'two'], dtype='object')
     0  1  2
one  1  2  3
two  4  5  6
False
```

Methods

head() : To display top five records in the dataset

tail() : To display last five records in the dataset

isnull(): Detect missing values.

fillna([]):Fill NA/NaN values using the specified method

dropna():Remove missing values.

info()-It displays general information about the dataframe

astype() -It can change the column type

DataFrame

Methods :

describe() - statistical characteristics of each numerical feature such as number of non-missing values, mean, standard deviation, range, median, 0.25 and 0.75 quartiles.

isnull().sum(): Detect missing values in each feature.

Append(): adding features to data set.

DataFrame

```
In [63]: import pandas as pd
data={'one':[1,2,3],'two':[None,5,6],'three':[7,8,9]}
df=pd.DataFrame(data)
print(df.isnull())
print( df.fillna((df.mean())))
print(df.dropna())
print(df.append(df1))
```

```
     one    two  three
0  False   True  False
1  False  False  False
2  False  False  False
   one  two  three
0    1  5.5      7
1    2  5.0      8
2    3  6.0      9
   one  two  three
1    2  5.0      8
2    3  6.0      9
   five  four  one  six  three  two
0   NaN   NaN  1.0  NaN    7.0  NaN
1   NaN   NaN  2.0  NaN    8.0  5.0
2   NaN   NaN  3.0  NaN    9.0  6.0
0   NaN   1.0  NaN  7.0    NaN  NaN
1   5.0   2.0  NaN  7.0    NaN  NaN
2   5.0   3.0  NaN  8.0    NaN  NaN
```

Methods

The Pandas I/O API is a set of top level reader functions accessed like pd.read_csv() that generally return a Pandas object.

```
import pandas as pd
data_loan=pd.read_csv("E:\\KMIT\\SONET\\NPTEL_Python_DS\\datasets\\loan.csv")
df_loan=pd.DataFrame(data_loan)
print(df_loan.head())
print(df_loan.tail())
print(df_loan.isnull())
print(df_loan.isnull().sum())
df_loan["Credit Score"].fillna(df_loan["Credit Score"].mean(),inplace=True)
print(df_loan["Credit Score"].head())
```

Methods

```
import pandas as pd
data_loan=pd.read_csv("E:\\KMIT\\SONET\\NPTEL_Python_DS\\datasets\\loan.csv")
df_loan=pd.DataFrame(data_loan)
df_loan.dropna(inplace=True)
print(df_loan.info())
print(df_loan.describe())
print(df_loan.dtypes)
df_cat=df_loan.select_dtypes(include=['object']).copy()
print(df_loan["Term"].head())
df_loan["Term"]=df_loan["Term"].astype("category")
print(df_loan["Term"].head())
```

Indexing and selecting Data

Indexing	Description
.loc()	Label based
.iloc()	Integer based

```
In [11]: import pandas as pd
         import numpy as np

         df = pd.DataFrame(np.random.randn(4, 4),
         index = ['a','b','c','d'], columns = ['A', 'B', 'C', 'D'])
         print(df)
         print(df.loc[:,'A'])
         print(df.iloc[:,1])
         print(df.ix[:,'A'])
```

```
          A         B         C         D
a -0.903447  0.349539 -2.079854  2.141442
b  0.487952  0.512143  1.788810  1.288846
c  0.136634  0.674832 -1.900180 -0.828820
d  0.134507 -0.458892  0.667469  1.074890
a   -0.903447
b    0.487952
c    0.136634
d    0.134507
Name: A, dtype: float64
a    0.349539
b    0.512143
c    0.674832
d   -0.458892
Name: B, dtype: float64
a   -0.903447
b    0.487952
c    0.136634
d    0.134507
Name: A, dtype: float64
```

Conclusion

You are aware of

Pandas

Reading Data from Various Sources

We will proceed with

More on Pandas

Kmit

SONET

DATA SCIENCE

THANK YOU