CREATIVE INSTITUTE DATA SCIENCE

Pandas INTORDUCTION

Pandas is one of the most popular Python libraries for Data Science and Analytics.

pandas is built on numpy. So, while importing pandas, import numpy as well.

Pandas is built on top of the Numpy package, means Numpy is required for operating the Pandas.

It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

You can install Pandas using the built-in Python tool `pip` and run the following command.

```
$ pip install pandas
```

Pandas Data Structures and Data Types

Data Types

- `object`: text or mixed numeric or non-numeric values
- `int64`: integer numbers
- `bool`: true/false vaues
- `float64`: floating point numbers
- `category`: finite list of text values
- `datetime64`: Date and time values
- `timedelta[ns]`: differences between two datetimes

CREATIVE INSTITUTE DATA SCIENCE

Pandas INTORDUCTION

Data Structure

A **data structure** is a particular way of organizing our data. Pandas has two data structures, and all operations are based on those two objects:

There are two core objects in pandas: the DataFrame and the Series.

- `Series`
- `DataFrame`

1) Series

It is defined as a one-dimensional array that is capable of storing various data types. The row labels of series are called the **index**.

Can easily convert the list, tuple, and dictionary into series using "series' method.

A Series cannot contain multiple columns. It has one parameter:

Data: It can be any list, dictionary, or scalar value.

Pandas Series is nothing but a column in an excel sheet.

Creating Series from Array:

```
import pandas as pd
import numpy as np
dt=np.array([1,2,3,4,5,6])
sr=pd.Series(dt)
print(sr)
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas INTORDUCTION

Accessing element of Series

There are two ways through which we can access element of series, they are :

- Accessing Element from Series with Position
- Accessing Element Using Label (index)

```
import pandas as pd
import numpy as np
dt=np.array([1,2,3,4,5,6])
sr=pd.Series(dt)
print(sr[1])
```

- Accessing Element Using Label (index)

```
import pandas as pd
import numpy as np
dt=np.array([1,2,3,4])
sr=pd.Series(dt,index=['alpha','beta','gama','delta'])
print(sr)
print(sr['delta'])
```

Create an Empty Series:

We can easily create an empty series in Pandas which means it will not have any value.

creates an Empty Series type object that has no values and having default datatype, i.e., **float64**.

```
import pandas as pd
a = pd.Series()
print (a)
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas INTORDUCTION

Create a Series from dict

We can also create a Series from dict.

dictionary object is being passed as an input and the index is not specified, then the dictionary keys are taken in a sorted order to construct the index**.**

If index is passed, then values correspond to a particular label in the index will be extracted from the dictionary.

```
import pandas as pd
import numpy as np
sr = {'name' : 'creative', 'branch' : 'katargam', 'no' : 333}
a = pd.Series(sr)
print (a)
```

Create a Series using Scalar:

If we take the scalar values, then the index must be provided.

The scalar value will be repeated for matching the length of the index.

```
import pandas as pd
import numpy as np
sr=pd.Series(4,index=[0,1,2,3])
print(sr)
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas INTORDUCTION

Series object attributes

The Series attribute is defined as any information related to the Series object such as size, datatype.

Attributes	Description
Series.index	Defines the index of the Series.
Series.shape	It returns a tuple of shape of the data.
Series.dtype	It returns the data type of the data.
Series.empty	It returns True if Series object is empty, otherwise returns false.
Series.hasnans	It returns True if there are any NaN values, otherwise returns false.
Series.nbytes	It returns the number of bytes in the data.
Series.ndim	It returns the number of dimensions in the data.

Retrieving Index array and data array of a series object

We can retrieve the index array and data array of an existing Series object by using the attributes index and values.

```
import numpy as np
import pandas as pd
a=pd.Series(data=[12,34,56,67,78])
b=pd.Series(data=[10.5,20.5,30.5], index=['a','b','c'])
print(a.index)
print(a.values)
print(b.index)
print(b.values)
```

CREATIVE INSTITUTE DATA SCIENCE

Pandas INTORDUCTION

Retrieving Types (dtype)

```
import numpy as np
import pandas as pd
a=pd.Series(data=[12,34,56,67,78])
b=pd.Series(data=[10.5,20.5,30.5], index=['a','b','c'])
print(a.dtype)
```

Retrieving Shape

The shape of the Series object defines total number of elements including missing or empty values(NaN).

```
import pandas as pd
import numpy as np
dt=np.array([1,2,3,4])
sr=pd.Series(dt,index=['alpha','beta','gama','delta'])
print(sr)
print(sr.shape)
```

Checking Emptiness and Presence of NaNs

```
import pandas as pd
import pandas as pd
import numpy as np
dt=np.array([1,2,3,4,np.nan])
sr=pd.Series(dt,index=['alpha','beta','gama','delta','Epsilon'])
print(sr)
print(sr.empty)
print(sr.hasnans)
print(len(sr))
print(sr.count)
```