Practical : 2

Aim: Study About Numpy And Pandas Libraries Of Python

Numpy:

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.
- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.
- This behavior is called locality of reference in computer science.
- This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

Creating Arrays:

0-D Arrays: 0-D arrays, or Scalars, are the elements in an array. Each value in an array is a 0-D array.

Example:

Code:

```
import numpy as np

arr = np.array(42)

print(arr)
```

output:

```
42
```

1-D Arrays: An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array.

Example:

Code:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

output:

```
[1 2 3 4 5]
```

2-D Arrays: An array that has 1-D arrays as its elements is called a 2-D array.

Example:

Code:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr)
```

output:

```
[[1 2 3]
 [4 5 6]]
```

3-D Arrays: An array that has 2-D arrays (matrices) as its elements is called 3-D array.

Example:

Code:

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(arr)
```

output:

```
[[[1 2 3]
 [4 5 6]]
 [[1 2 3]
 [4 5 6]]]
```

Slicing Arrays:

- Slicing in python means taking elements from one given index to another given index.
- We pass slice instead of index like this: [*start*:*end*].
- We can also define the step, like this: [*start*:*end*:*step*].

Example:

Code:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5])
```

Output:

```
[2 3 4 5]
```

Shape Of an Array:

- NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

Example:

Code:

```
import numpy as np
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)
```

Output:

(2,4)

Reshaping Arrays:

- Reshaping means changing the shape of an array.
- By reshaping we can add or remove dimensions or change number of elements in each dimension.

Example:

Code:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(4, 3)

print(newarr)
```

Output:

```
[[1 2 3]

 [4 5 6]

 [7 8 9]

 [10 11 12]]
```

Pandas:

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

Series:

- A Pandas Series is like a column in a table.
- It is a one-dimensional array holding data of any type.

Example:

Code:

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

Output:

```
0    1

1    7

2    2

dtype:int64
```

Labels:

- If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.
- This label can be used to access a specified value.

Example:

Code:

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

Output:

```
X       1

Y       7

Z       2

dtype:int64
```

Dataframes:

- A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Example:

Code:

```
import pandas as pd

data = {
 "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

Output:

```
        Calories        duration

0       420             50
```

1	380	40
2	390	45

Read CSV Files:

- A simple way to store big data sets is to use CSV files (comma separated files).
- CSV files contains plain text and is a well know format that can be read by everyone including Pandas.
- In our examples we will be using a CSV file called 'data.csv'.

Example:

Code:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.to_string())
```

Output:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
3	60	103	135	340.0
4	45	109	175	282.4
5	30	109	133	195.1

Viewing The Data:

- One of the most used method for getting a quick overview of the DataFrame, is the head() method.
- The head() method returns the headers and a specified number of rows, starting from the top.

Example:

Code:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head(10))
```

Output:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.5

- There is also a tail() method for viewing the *last* rows of the DataFrame.
- The tail() method returns the headers and a specified number of rows, starting from the bottom.

Example:

Code:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.tail())
```

Output:

	Duration	Pulse	Maxpulse	Calories
164	60	105	140	290.8
165	60	110	145	300.4

166 60 115 145 310.2

167 75 120 150 320.4

168 75 125 150 330.4

Info about data:

- The DataFrames object has a method called info(), that gives you more information about the data set.

Example:

Code:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.info())
```

Output:

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 169 entries, 0 to 168

Data columns (total 4 columns):

 #   Column    Non-Null Count  Dtype

---  ------    --------------  -----

 0   Duration  169 non-null    int64

 1   Pulse     169 non-null    int64

2   Maxpulse  169 non-null    int64

 3   Calories  164 non-null    float64

dtypes: float64(1), int64(3)

memory usage: 5.4 KB

None
```