Data Viusalisation with Python Programming

- Presentation By Uplatz
- Contact us: https://training.uplatz.com
- Email: info@uplatz.com
- Phone: +44 7836 212635

Uplatz

Numpy and Pandas

Learning outcomes:

NumPy and Pandas

- What is numpy?
- Why use numpy?
- Installation of numpy
- Example of numpy
- What is pandas?
- Key features of pandas
- Python Pandas - Environment Setup
- Pandas – Data Structure with example

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.
NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy

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.

Why use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.
NumPy aims to provide an array object that is up to 50 time faster that traditional Python lists.
The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
Arrays are very frequently used in data science, where speed and resources are very important.

Installation of NumPy

Standard Python distribution doesn't come bundled with numpy module. A lightweight alternative is to install numpy using popular Python package installer, **pip.**

pip3 install numpy OR
python -m pip install numpy OR
pip install --user numpy

Example of NumPy

NumPy is used to work with arrays. The array object in NumPy is called ndarray.

We can create a NumPy ndarray object by using the array() function.

To create an ndarray, we can pass a list, tuple or any array-like object into the array() method, and it will be converted into an ndarray.

Example of NumPy

Example:

```
import numpy
arr = numpy.array([1, 2, 3, 4, 5])
print(arr)
```

NumPy is usually imported under the np alias.

Example:

```
import numpy as np
arr1 = np.array((10, 20, 30, 40, 50))
print(arr1)
```

Example of NumPy

Example: **Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6:**

```
import numpy as np
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
```

Example: Create a 3-D array with two 2-D arrays, both containing two arrays with the values 1,2,3 and 4,5,6:

```
import numpy as np
arr3 = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3],
[4, 5, 6]]])
print(arr3)
```

Example of NumPy

Example: **Check Number of Dimensions:**

NumPy Arrays provides the ndim attribute that returns an integer that tells us how many dimensions the array have.

```
import numpy as np
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Pandas

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data. Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyse.

Key features of pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.

Key features of pandas

- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Pandas - Environment Setup

Standard Python distribution doesn't come bundled with Pandas module. A lightweight alternative is to install pandas using popular Python package installer, **pip.**

pip3 install pandas OR
python -m pip install pandas OR
pip install --user pandas

Pandas – Data Structure

Pandas deals with the following three data structures .

1. **Series**
2. **DataFrame**
3. **Panel**

These data structures are built on top of Numpy array, which means they are fast.

The best way to think of these data structures is that the higher dimensional data structure is a container of its lower dimensional data structure. For example, DataFrame is a container of Series, Panel is a container of DataFrame.

Pandas – Data Structure

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, 17, 52

10	23	56	17	52

Key Points:

Homogeneous data

Size Immutable

Values of Data Mutable

Pandas – Data Structure

Series:

A pandas Series can be created as follows:

pandas.Series(data, index, dtype, copy)

data: data takes various forms like ndarray, list, constants

index: Index values must be unique and hashable, same length as data. Default **np.arrange(n)** if no index is passed.

dtype: dtype is for data type.

copy: Copy data. Default False

Pandas – Data Structure

Series:

Example:

```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data,index=[100,101,102,103])
print (s)
```

Pandas – Data Structure

DataFrame:

As a matter of first importance, we are going to discuss from where the idea of a data frame came. The cause of data frame came comes from serious experimental research in the realm of statistical software. The tabular data is referred by the data frames. Specifically, data frame is a data structure that speaks to cases in which there are various observations(rows) or measurements (columns).

Pandas – Data Structure

DataFrame:

DataFrame is a two-dimensional array with heterogeneous data.

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9

Each column represents an attribute and each row represents a person.

Pandas – Data Structure

DataFrame:

Key Points

- Heterogeneous data
- Size Mutable
- Data Mutable
- Potentially columns are of different types
- Size – Mutable
- Labelled axes (rows and columns)
- Can Perform Arithmetic operations on rows and columns

Pandas – Data Structure

DataFrame:

A pandas DataFrame can be created as

pandas.DataFrame(data, index, columns, dtype, copy)

data: data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.

index: For the row labels, the Index to be used for the resulting frame is Optional Default np.arange(n) if no index is passed.

Pandas – Data Structure

DataFrame:

A pandas DataFrame can be created as pandas.DataFrame(data, index, columns, dtype, copy)

columns: For column labels, the optional default syntax is - np.arange(n). This is only true if no index is passed.

dtype: Data type of each column.

copy: This command (or whatever it is) is used for copying of data, if the default is False.

Pandas – Data Structure

DataFrame:

Example_1:

```
import pandas as pd
data = [1,2,3,4,5]
df = pd.DataFrame(data)
print (df)
```

Example_2:

```
import pandas as pd
data2 = [['Alex',10],['Bob',12],['Clarke',13]]
df1 = pd.DataFrame(data2,columns=['Name','Age'])
print (df1)
```

Pandas – Data Structure

DataFrame: Create a DataFrame from Dictionary

Example_1:

```
import pandas as pd
data3 = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],
'Age':[28,34,29,42]}
df2 = pd.DataFrame(data3)
print (df2)
```

Pandas – Data Structure

Panel:

Panel is a three-dimensional data structure with heterogeneous data. It is hard to represent the panel in graphical representation. But a panel can be illustrated as a container of DataFrame.

Key Points

Heterogeneous data

Size Mutable

Data Mutable

The Panel class is removed from pandas

Reading CSV Files With pandas

To show some of the power of pandas CSV capabilities, I've created a slightly more complicated file to read, called hrdata.csv.

Name,Hire Date,Salary,Sick Days remaining
Graham Chapman,03/15/14,50000.00,10 John Cleese,06/01/15,65000.00,8 Eric Idle,05/12/14,45000.00,10 Terry Jones,11/01/13,70000.00,3 Terry Gilliam,08/12/14,48000.00,7 Michael Palin,05/23/13,66000.00,8

Copy this data in notepad and save it as hrdata.csv into current working directory of python.

Reading CSV Files With pandas

Reading the CSV into a pandas DataFrame is quick and straightforward:

```
import pandas as pd
df5 = pd.read_csv('hrdata.csv')
print(df5)
print(df5['Name']) #Reading the date from column "Name"
```

Reading CSV Files With pandas

Here are a few points worth noting:

- First, pandas recognized that the first line of the CSV contained column names, and used them automatically. I call this Goodness.
- However, pandas is also using zero-based integer indices in the DataFrame. That's because we didn't tell it what our index should be.
- Further, if you look at the data types of our columns , you'll see pandas has properly converted the Salary and Sick Days remaining columns to numbers, but the Hire Date column is still a String. This is easily confirmed in interactive mode.
- print(type(df5['Hire Date'][0]))