ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

A Brief Study on Python and its use in Data Science

Shubham Bagre

B.E Student, Dept. of CSE., Angadi Inst. of Tech. and Mgmt., Belagavi, India

ABSTRACT: In this paper I demonstrate what actually python is along with its characteristics and the advantages of using python programming. I will also demonstrate how it can be used throughout the Data Science. Python's ease of use, open source license and access to vast array of libraries make it particularly suited for programmers and students. In particular, I will discuss how the packages NumPy, SciPy and Pandas are used in several phases of data science, along with SQLite3 and PyTable packages.

KEYWORDS: Python; Data Science; Data Science packages; NumPy; SciPy; Pandas; SQLite3;

I. INTRODUCTION

The brief study below is been carried out in order to focus on the educational characteristics of python and its effective use in new emerging technologies like Data Science. The various important packages in this field are explained in the study below.

II. WHAT IS PYTHON?

Python is a general purpose programming language created in late 1980s. Python is dynamically typed language, which simply means that you don't have to define the type of the declared variable, as we do in C, Java, etc. Python is an interpreted language. Python programs runs directly from source code. It converts the source code that is written by the programmer into an intermediate language, which is again translated into machine language that has to be executed.

In python, module is the way to structure program. Each python program file is a module, which imports other modules like objects and attributes. And python package is a set of modules. Python is designed to be used interpretively. A python statement may be entered at the interpreter prompt (>>>), and will be executed immediately.

III. CHARACTERISTICS OF PYTHON PROGRAMMING

- **Simple:**

Python is a simple and minimalistic language. Reading a good program feels like almost like reading English language. This pseudo-code nature of python is one of its greatest strengths.

- **Easy of Learn:**

Python is easy to get started with. Python has an extraordinary simple syntax.

- **High-level language:**

When you write programs in python, you need not to be bothered about low-level details such as managing the memory used by program.

- **Portability:**

Due to its open source nature, python has been ported to many platforms. All your python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features. We can use python on Linux, Windows, Solaris, Macintosh, OS/2, Amiga, PlayStation, PocketPC etc.

- **Interpreted:**

A program written in a compiled language like C or C++ is converted from source language into binary coded language using a compiler with various options. When we run the program the linker/loader software copies the program from hard disk to memory and start running.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

Python, on other hand, does not need compilation binary. We just run the program directly from the source code. Python converts the source code into an intermediate form called byte codes and then translated into native language of our computer and then runs it.

- **Object oriented:**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented language, the program is built around procedures also called as functions. In object-oriented language, the program is built around objects which combine data and functionality.

- **Extensible:**

If we need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, we can code that part of our program in C or in C++ and then use these in our python program.

- **Embedded:**

We can embed python within C/C++ programs to give 'scripting' capabilities for our programs users.

IV. ADVANTAGES OF PYTHON PROGRAMMING OVER OTHER PROGRAMMING LANGUAGES

- **Presence of third party modules:**

The python packages index (PyPi) contains numerous third party modules that make python capable of interacting with mist of other languages and platforms.

- **Extensive support libraries:**

Python provides large standard library which include areas like internet protocols (IP), string operations, web service tools and operating system interfaces. Using the high level standard libraries the length of code can be significantly reduced.

- **Open source and community development:**

Python language is developed under an OSI-approved open source license, which makes it free to use and distribute, including for commercial purpose. Its development is driven by the community which collaborates for its code through hosting conferences and mailing lists, and provides for its numerous modules.

- **Learning ease and support available:**

Python offers excellent readability and uncultured simple-to-learn syntax which helps beginners to utilize this programming language.

- **User-friendly data structures:**

Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures. Python also provides the option of dynamic high-level typing which reduces the length of support code that is needed.

- **Productivity and speed:**

Python has clean object-oriented design, provides enhanced process control capabilities and possesses strong integration and text processing capabilities. Python programs are 3-4 times shorter than any other language programs. For example, for a simple program which display HELLO WORLD! We need to declare the header files, define the main function we the brackets etc. in java and C++ programs. But whereas in python we can skip all these above steps of declaring header, main and can directly print HELLO WORLD! using "Print" keyword.

V. WHAT IS DATA SCIENCE?

Data Science is an interdisciplinary burgeoning field of study to extract knowledge from data in either structured or unstructured form which is continuation of some of the data analysis fields such as machine learning, predictive analytics similarly, data science lies at the intersection of statistics, computer science and numerous applied scientific domains.

Merely using data isn't really what we mean by "Data Science". It's not just an application with data; but data product. Data science enables creation of data products. Data science is a holistic approach. It is the study of where information comes from, what it represents and how it can be turned into a valuable resource in the creation of business and IT strategies. The expertise professional in data science are Data Scientists. They use technology and skills to increase clarity, awareness and direction for those working with data.

The three components involved in data science are organizing, packaging and delivering data which is also referred as OPD of data.

- **Organizing** is where the physical location and structure of data is planned and executed.
- **Packaging** is where the prototypes of data arebuild, the statistics is performed and the visualization is created.
- **Delivering** is where the story gets told and the value is obtained as a form of result.

The few important python libraries/packages used in Data science are discussed below:

VI. NUMPY

NumPy, short for “Numerical Python” is the fundamental package for scientific computing with python. The NumPy package provides basic routines for manipulating large arrays and matrices of numeric data. It provides, among other things:

- A fast and efficient multidimensional array objects *ndarray*.
- Sophisticated functions for performing element wise computations
- Linear algebra operations, Fourier transform and random number generation.
- Tools for integrating C, C++ and FORTRAN code to python.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

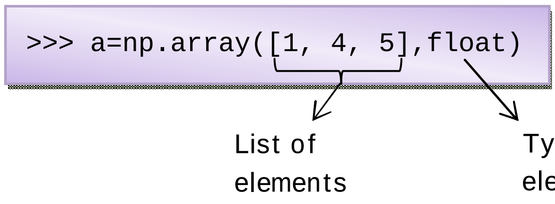
i. Importing the NumPy module:

The most standard approach is to use a simple import statement. For large amount of NumPy function calls we make use of dot notation (.) that is, `numpy.x`

```
>>>import numpy
```

ii. Arrays:

Arrays are similar to lists in python but the only difference is that the elements in array must be of same type; typically *float* or *int.* the function array takes two arguments. The first is the list of elements of array and the second is the type of each element in list.



```
>>> a=np.array([1, 4, 5],float)
```

Arrays can be multi-dimensional, we can represent the multi-dimensional array using the comma (,) inside bracket notation. Below is the example of 2-dimensional array.

```
>>> a=np.array([[1, 4, 5],[3, 7, 9]] , float)
```

We can use shape, dtype, len and in functions to get the size of array dimension, type of value stored in array, length of first axis if array and to test if values are present in an array respectively.

iii. Polynomial mathematics:

NumPy provides methods for working with polynomials. It is possible to show polynomial coefficients when a set of roots is given.

```
>>> np.poly([-1, 1, 1, 10])array([1, -11, 9, 11, -10])
```

The above section of code returns array that gives the coefficients corresponding to quadratic equation: $x^4-11x^3+9x^2+11x-10$.

Similarly when given a set of coefficients, the root function returns all the polynomial roots as mentioned in block of code below.

```
>>>np.roots([1, 4, -2, 3])array([-4.57974010+0.j,
      0.28987005+0.75566815j
         0.28987005-0.75566815j])
```

The functions polyadd, polysub, polymul and polydiv handle proper addition, subtraction, multiplication and division of polynomial coefficients respectively.

iv. **Statistics:**

NumPy provides several other methods for returning statistical features of array along with mean, var, std functions. We can calculate median of given array list.

```
  >>> a=np.array([3, 7, 10, 12,3, 7, 9, 11, 5] , float)
>>> np.median(a)
   9.0
```

Similarly, covariance for data and correlation coefficient for multiple variables at multiple instances be found.

VII. SCIPY

SciPy is built on the NumPy array framework and performs some advanced mathematical functions like integration differential equation solvers, special functions, optimization and more. We can say SciPy is the extension of NumPy functionalities.

All SciPy modules should follow the following conventions.

- Ideally, each scipy module should be self-contained as possible. That is, it should have minimal dependencies on other packages or module.
- Directory scipy/ contains a file `setup.py` that defines `Configuration(parent_package='',top_path=none)`function for numpy.distutils.
- Private module should be prefixed with an underscore(_) for instance`Scipy/_somemodule.py`

i. **Importing the Scipy:**

The SciPy module is imported same as that of NumPy import statement.

```
>>>import scipy
```

ii. **Integration:**

Integration is a crucial tool in math and science as, differentiation and integration are two key components of calculus. The main purpose of integration with scipy is to obtain numerical solutions. Scipy has a range of different functions to integrate equations and data.

For example, working with the integrate function below the block of code to be used is shown.

$$\int_0^1 sin^2(e^x)\, dx$$

```
importnumpyasnp
fromscipy.integrateimport quad
#defining function to integrate
func=lambda x:np.sin(np.exp(x))**2
#integrating function with upper #and lower limits
solution=quad(func,0,1)
print solution
```

iii. **Other available sub-packages in SciPy:**

SciPy imports all the functions from the Numpy namespace and in addition it also provides: many sub-packages, few of them are mentioned below.

- fftpack - Discrete Fourier transformation algorithms.
- io - Data input and output.
- stats - Statistical functions,
- ndimage - n-dimensional image package.
- linalg - linear algebra routines.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

- spatial - spatial data structures and algorithms
- interpolate- interpolation tools.
- optimize- optimization tools.

The above sub-packages are imported explicitly along with the import statement of SciPy, as mentioned in example below.

```
>>>importscipy.interpolate
```

VIII. PANDAS

Pandas is a python package providing fast, flexible and expressive data structures. Pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment. It is one of the most powerful and flexible open source data analysis/manipulation tool available. Pandas is fast and is dependency of *statsmodel*, making it an important part of the statistical computing ecosystem in python. The things that pandas do well are: *handling of missing data, Automatic and explicit data alignment, Intuitive merging and joining data sets, flexible reshaping and pivoting of data sets.*

Pandas is well suited to work with tabular data with heterogeneously typed columns, ordered and unordered time series data, Arbitrary matrix data with row and column labels and many more.

i. Data structures of Pandas:

The two primary data structures of pandas are *Series* and *DataFrame*. The data structure series is one-dimensional and DataFrame is 2-dimensional.

i.i Series:

Series is a one-dimensional labeled array capable of holding any data type (integer, strings, floating point numbers, python objects, etc.). the axis labels are collectively referred to as the ***index***. The basic method to create a series is by calling the in-built function `series` of pandas. Series accept many different kinds of data, they can be either python dict or an ndarray or a scalar value.

```
>>>s=series(data, index=index)
```

i.ii DataFrame:

DataFrame is a 2-dimensional labeled data structure with many columns of different types like as in SQL table. It is generally the most commonly used pandas object. Unlike series, DataFrame also accept different kinds of input.

- A series
- 2-D numpy.ndarray
- Another DataFrame
- Structured ndarray

Along with data, we can optionally pass index and columns arguments. If we pass these arguments we guarantee the type of resulting DataFrame.

ii. Remote data access using Pandas:

Functions from Pandas library that is `Pandas.io.data`extract data from various internet sources into DataFrame. Currently few of the following sources are supported:

- Yahoo! Finance
- Google finance
- World Bank

The various sources support different kinds of data, hence all sources implement different methods and the data elements returned might also differ. The following are the block of section of code used to extract data into DataFrame.

i.i Yahoo! Finance:

```
importpandas.io.dataas web
importdatetime
start=datetime.datetime(2015,01,01)
end=datetime.datetime(2016,01,01)
f=web.DataReader(“f”,’yahoo’,start,end)
```

i.ii Google finance:

```
importpandas.io.dataas web
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

```
importdatetime
start=datetime.datetime(2015,01,01)
end=datetime.datetime(2016,01,01)
f=web.DataReader("f",'google',start,end)
```

IX. SQLITE3

SQLite3 is one of the important packages used in data format storing in Data Science using python. SQLite is a library that provides a light weight disk-based database that doesn't require a separate server process and allow accessing the database using a variant SQL query language. The SQLite3 module in python provides a SQL interface with DB-API 2.0 specification. To use SQLite3 module first we should create a `Connection` object that represents the database along with file name where the data will be stored.

```
>>>import sqlite3
Conn=sqlite3.connect('example.db')
```

We can also create the database in RAM instead of on disk using special name "`:memory:`", once the connection object is returned we can create a `cursor` object and call its `execute()` method to perform SQL commands. The below sections of code represent the creation of table and insertion of data in created database.

i. To create table:

```
import sqlite3
conn=sqlite3.connect('sample.db')
print"Database opened successfully";
conn.execute('''CREATE TABLE COMPANY
        (ID INT PRIMARY KEY NOT NULL,
            NAME TEXT NOT NULL);''')
print"Table created successfully";
conn.close()
```

ii. To insert the data:

```
import sqlite3
conn=sqlite3.connect('sample.db')
print"Database opened successfully";
conn.execute("INSERT INTO COMPANY(ID,NAME)\VALUES(1211,'Sam')");
conn.execute("INSERT INTO COMPANY(ID,NAME)\VALUES(1311,'Joe')");
conn.execute("INSERT INTO COMPANY(ID,NAME)\VALUES(1111,'Amy')");
conn.commit()
print"Records created successfully";
conn.close()
```

As we inserted the data in the database, in similar way we can retrieve the selected information using SELECT clause. We can also delete and update the data from specified table in database using DELETE and UPDATE clause respectively.

iii. Module functions:

SQLite3 in python has many different module functions and constants and few of them are mentioned below.

➢ **`Sqlite3.version`**

The version number of this module is given in form as a string. It is not the SQLite library version. By using ***version_info*** the version number of this module is represented in form of tuple of integer.

➢ **`Sqlite3.sqlite_version`**

The version number of run-time SQLite library is given in form as a string.

➢ **`Sqlite3.PARSE_DECLTYPES`**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

It makes the sqlite3 module parse the declared type for each column it returns. This constant is used with the *detect_types* parameter of `connect()` function.

- **`Sqlite3.register_converter`**

Register a callable to convert a bytestring from database into a customized python type.

- **`Sqlite3.complete_statement`(sql)**

Returns `True` of the string sql contains one or more complete SQL statements terminated by semicolon(;). It will not verify whether SQL is syntactically correct or not.

X. PYTABLES

The goal of PyTables is to enable the end user to manipulate data tables and array objects in a hierarchical structure easily. It provides a *flexible, very pythonic*tool to deal with large amounts of data in tables and arrays organized in a hierarchical disk storage structure.

A table is defined as a collection of data information stored in ***fixed-length*** fields. All the information records have same structure and values in each field have same data type. It is a very strange requirement when I use terms fixed-length and strict data types for an interpreted language like python, but they serve a very useful function if the goal is to save large quantity of data (such as is generated by data acquisition systems, internet services, for example) in an efficient way that reduces demand on I/O and CPU time.

Other important entities in PyTable are *array* objects, which are analogous to tables with the difference that all the components of array are homogeneous. They come in different categories, like

- *Generic*- they provide a quick and fast way to deal with numerical arrays.
- *Enlargeable*- arrays can be extended in a single dimension.
- *Variable length*- each row in array can consist of different number of elements.

i. Main features of PyTable:

PyTable takes advantage of object orientation capabilities offered by python. Few of the features are:

- **Multidimensional and nested table cells:**

We can declare a column consisting of values having any number of dimensions, which is the only dimensionality allowed by majority of databases. We can also declare columns that are made of other columns.

- **Indexing support for columns:**

It is very useful if we have large tables and want a quickly look up at particular column satisfying some criteria.

- **Support for numerical arrays:**

NumPy arrays can be used as a complement of tables to store homogeneous data.

- **Enlargeable arrays:**

We can add new elements to already existing arrays on disk in any dimension. Beside, we can access just a slice of datasets by powerful slicing mechanism without loading all the dataset in memory.

- **Variable length arrays:**

As the array length varies from row to row, it provides a lot of flexibility when dealing with complex data.

- **Supports a hierarchical data model:**

It allows the user to clearly structure data. PyTables build up an *object tree* in memory that replicates the underlying file data structure. Accessing these objects in file is achieved by manipulating this object tree.

- **User defined metadata:**

Besides supporting system metadata the user may specify arbitrary metadata that complement the meaning of actual data.

- **Data compression:**

It is important to compress the data when we have repetitive data patterns in order not to spend much time searching for optimized way to store them.

ii. Object tree:

PyTables manage tables and arrays in a tree-like structure. In order to achieve this, an *Object tree* entity is dynamically created on disk. We can get the information of what kind of data is kept in the examining *metadata* nodes. The different nodes in the object tree are instances of PyTables classes. The most important classes among other several types are the node, Group and leaf classes.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

All nodes in PyTables tree are instances of node class. The Group and leaf classes are descendants of node. Group instances are structure consisting of zero or more instances of groups with metadata. Leaf instances are actual data containers and also contain further groups or leaves. Working with groups and leaf nodes is similar as working with directories and files in UNIX filesystem. A node is always a *child* of one and only one group, its *parent group*. The below is the PyTables script to illustrate the object tree entity.

```
from tables import *
classparticle (IsDescription):
#character string
name=stringCol(itemsize=22, dflt=” “, pos=0)
#short integer
eid=Int16Col(dflt=1, pos=1)
#single-precision
salary=Float32Col(dflt=1, pos=2)
#open a file in “w”rite mode
fileh=open_file(“objecttree.h5”, mode=”w”)
#get the HDF5 root group
root=fileh.root
#create the group
group=fileh.create_group(root, “group”)
#now, create an array in root group
array1=fileh.create_array(root,”array1”,[“string”,”array”],”string array”)
#create table in group
table1=fileh.create_table(group, “table1”, particle)
#create last table in group
array2=fileh.create_array(“/group”,”array2”,[1,2,3,4])
#now, fill the table
for table in (table1):
row=table.row
#fill the table with 5 records
fori in xrange(5):
#first, assign the values to the particle record
row[‘name’]=’this is employee:%d’ %(i)
row[‘eid’]=i
row[‘salary’]=i*10000
#this injects record values
row.append()
#flush the table buffer
table.flush()
#finally close the file
fileh.close()
```

XI. CONCLUSION

We have discussed what python is and how it is used in Data Science. The vast array of libraries and the packages make it easy to learn and also to use them effectively and efficiently. In particular we discussed what Data Science is and what are some important packages and libraries that can be used in Data Science, like NumPy, SciPy, Pandas, SQLite3 and PyTables.

REFERENCES

[1] Richard L. Halterman,“Learning to program with python”, pp.11-76,115-156,181-200,235-244, 2011.
[2] Lutz, Mark.,“Programming python”,5th edition, O’Reilly media publications, pp.12-250, Inc. 2010.
[3] McKinney, Wes.,“Python for Data analysis: Pandas, NumPy”, O’Reilly media publications, pp.3-6,79-150, Inc. 2012.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 11, November 2016

[4] http://blog.teamtreehouse.com/what-is-python.html
[5] http://www.wikipedia.com
[6] https://github.com
[7] http://pandas.pydata.org/pandas-docs/stable/merging.html
[8] http://www.Scipy.org/scipylib/index.html
[9] http://www.Numpy.org/
[10] http://www.pytables.org
[11] https://docs.python.org/3.4/library/Sqlite3.html