*International Journal of Engineering Applied Sciences and Technology, 2016*
**Vol. 1, Issue 8, ISSN No. 2455-2143, Pages 299-302**
Published Online June - July 2016 in IJEAST (http://www.ijeast.com)

# DESIGNING ISSUES IN PYTHON LANGUAGE - A REVIEW

Shally Agrawal
Department of IT
IET-DAVV, Indore (M.P)

***Abstract*-A data scientist needs data which is to be manipulated for visualizing results .For manipulation there already exists numerous methods which can be used, there is no need to reproduce those methods again. Python is able to draw a curve, smooth a signal, do a Fourier transform in a few minutes. For the easy communication of results, the language should contain as few syntax symbols as possible that would divert the reader from the mathematical or scientific understanding of the code. Also for efficient usage, a single environment is required as downloading different software make the job tedious. Computer data scientist can either use compiled language or scripting language. Compiled languages though fast have disadvantages of being less interactive, verbose syntax and manual memory management. On the other hand Python works on the phrase "we code what we think" making it more interactive and user friendly. It has rich computing libraries with widely spread open source environment. It is a very readable language with clear non-verbose syntax. It is easy to combine Python with compiled languages, like Fortran, C, and C++, which are widely used languages for scientific computations. This study emphasises on the Python's role in scientific computing. .In this paper, it is demonstrated how Python can be used in scientific computing.**

*Keywords*-**Scientific Computing, Python, Compiled languages, Scripted languages**

## I. INTRODUCTION

Scientific Computing is the burgeoning field that involves statistics, computer science and numerous applied scientific domains. Python is programming language developed in the late 1980s, by Guido van Rossum. It is used by thousands of people to do things from testing microchips at Intel, to powering Instagram, to building video games with the PyGame library. Python very closely resembles the English language .Being an open source language, a lot of this has been released for others to use. Dealing with large amounts of data, processing and visualising it can be a challenge. Python with libraries like NumPy, Scipy and matplotlib has become powerful scientific computing language.

## II. SCIENTIFIC COMPUTING MODULES

### A. **NumPy (Numerical Python)-**

NumPy(Numerical Python) is the fundamental package for scientific computing with Python which can apply linear algebra, fourier transforms, high performance vector and make a N-dimensional powerful array object. Python with NumPy module provides a best alternative to FORTRAN and C++.To use NumPy module following syntax is used:

```
>>>Import numpy as np
```

NumPy provides:

- Extension package to Python for multi-dimensional array
- Higher efficiency
- Array oriented programming

Ways to initialize NumPy array:

- Lists
- Functions
- Reading data from files

**Lists**-To create new vector and matrix arrays from Python lists we can use the numpy.array function.

```
>>>a=np.array([2,4,6,8])
```

Getting the other attributes of created array-

```
>>>a.ndim
>>>1
>>>a.shape
>>>(4,)
>>>len(a)
>>>4
```

Fig 1.1

Functions : To create large arrays it is impractical to enter data manually, so in this case arrays are generated by functions like arange, linespace and logspace,mgrid, random data, diag, zeros and ones etc.

Using arange function:-

```
>>>a=arange(2,10,2)#start,stop,step
>>>a
>>>([2,4,6,8,10,12,14,16,18,20])
```

Fig. 1.2

Using mgrid:-

```
>>>a,b=mgrid[0:5, 0:5]
>>>a
>>>array([[0, 0, 0, 0, 0],
              [1, 1, 1, 1, 1],
              [2, 2, 2, 2, 2],
              [3, 3, 3, 3, 3],
               [4, 4, 4, 4, 4]]
```

Fig 1.3

Using diagonal matrix:-

```
>>>diag([1,2,3])
>>>([[1, 0, 0],
      [0, 2, 0],
      [0, 0, 3]])
```

Fig 1.4

## B. Scipy (Scientific Python)

Build on the top of the low-level framework of NumPy framework for multi-dimensional frameworks provide many scientific algorithms. The SciPy extends the functionality of NumPy with a collection of useful algorithms. It is the core package for scientific routine in python. It contains many toolboxes useful for solving common issues in scientific computing. Its different sub-modules correspond to different applications. Some are as follows-

- Special functions
- Integration
- Fourier transformations
- Linear algebra
- Interpolation
- Statistics
- File IO
- Optimization
- Image and Signal Processing
- Sparse Eigen value Problems

Each of these sub-modules provides a number of functions and classes that can be used to solve problems in their respective topics.

To access the SciPy package following syntax is used:

```
>>>Import SciPy as sp
```

```
>>>from scipy import integrate
>>>result , error = integrate.quad(np.sin , 0,np.pi)
>>>result , error
>>>(2.0, 2.220446049250313e-14)
```

Fig.1.5

## C. .Matplotlib:

Matplotlib is an exclusive library for generating 2D scientific figures which can be controlled programmatically. It is used for visualizing results. It provides both a very quick way to visualize data from Python and publication-quality figures in many formats. Matplotlib tries to make easy things easy and hard things possible. It can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc, with just a few lines of code. The key advantages are-

- Support LATEX formatted labels and texts
- Good control of every element in figure
- High quality outputs in different formats

Matplotlib comes with a set of default settings that allow customizing all kinds of properties like figure size and dpi, line width, color and style, axes, axis and grid properties, text and font properties etc.

To access Matplotlib:-

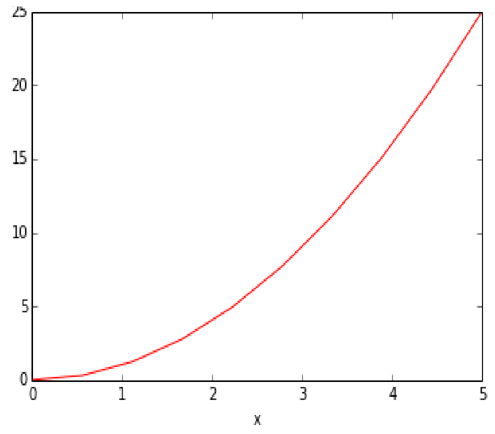
```
>>>import matplotlib.pyplot as plt
```

Input:

```
x = linspace(0, 5, 10)
y = x ** 2
In[6]: figure()
plot(x, y, 'r')
xlabel('x')
ylabel('y')
title('title')
show()
```

**Fig. 1.6**

Output:



## III. REVIEW OF LITERATURE

Guido van Rossum, Python’s original author, explains “This emphasis on readability is no accident. As an object-oriented language, Python aims to encourage the creation of reusable code. Even if we all wrote perfect documentation all of the time, code can hardly be considered reusable if it’s not readable. Many of Python’s features, in addition to its use of indentation, conspire to make Python code highly readable.”

Long time pythoneer tim peters writes about the Pyton design, The Zen of Python: Beautiful is better than ugly; Explicit is better than implicit; Simple is better than complex; Complex is better than complicated; Flat is better than nested; Sparse is better than dense; Readability counts; Special cases aren't special enough to break the rules; Although practicality beats purity; Errors should never pass silently; Unless explicitly silenced; In the face of ambiguity, refuse the temptation to guess; There should be one— and preferably only one —obvious way to do it; Although that way may not be obvious at first unless you're Dutch; Now is better than never; Although never is often better than right now; If the implementation is hard to explain, it's a bad idea; If the implementation is easy to explain, it may be a good idea; Namespaces are one honking great idea—let's do more of those!

## IV. FINDINGS

- Python is a highly interactive programming language that gives students a chance to learn by interactive experiments and exploration as it is object oriented, imperative and functional.
- NumPy and SciPy are the bread and butter extensions for numerical analysis and scientific computing. They provide almost all the functionalities required for data science.
- Python being an open source offers wide range of libraries useful for so many applications in many areas.
- Readability has a number of beneficial effects.
- Many common mathematical and numerical routines have been pre-compiled to run very fast and grouped into packages that can be added to Python in an entirely transparent manner.
- Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems
- Python has an extensive eco-system for scientific libraries and environments.

  - numpy: http://numpy.scipy.org - Numerical Python
  -scipy:http://www.scipy.org
  -matplotlib: http://www.matplotlib.org

- Existing numerical C/Fortran libraries can be interfaced to be usable from within Python

## V. CONCLUSION

Scientific computing in python has expanded many folds. Python has become the language of choice for many people in scientific computing. Python has maintained the philosophy of “batteries included" .portability, flexibility, extendibility, syntax and style of python overcome the disadvantage of slower than other languages like JAVA,C++ etc. The use of python in the areas like scientific computing, web and internet development, developing games, Desktop GUIs ,software development etc is making it more popular and widely used language of all the times, even it is the official language at GOOGLE. Python has been instrumental as it put the advanced software techniques within our reach. Programming languages have their own ecosystems, cultures and philosophies distributing their roles, for each role the most suitable language are chosen. Python being a dynamic programming language is perfect for major applications. Companies worldwide are using Python to harvest insights from their data and get a competitive edge.

## VI. REFERENCES

[1] Robert Johansson. Introduction to Scientific Computing in Python.

[2] Hans Petter Langtangen. A Primer on Scientific Programming with Python.

[3] Zed A. Shaw. Learn Python the hard way.
[4] Atanas Radenski. "Python First": A Lab-Based Digital Introduction to Computer Science
[5] Hans Fangohr. Introduction to Python for Computational Science and Engineering.
[6] Jacco Hoekstra. AE Tutorial Programming Python.
[7] Randy Paffenroth, Xiangnan Kong. Python in Data Science Research and Education. Proc. of the 14th python in science conf. (scipy 2015).
[8] M. Scott Shell. An introduction to Python for scientific computing.
[9] Christian Seberino. Python: faster and easier software development