Introduction to Python Language

Prof. (Dr.) K.R. Chowdhary
Email: kr.chowdhary@jietjodhpur.ac.in

Campus Director,
JIET College of Engineering, Jodhpur

Tuesday 6th June, 2017

History

- A popular programming language, Python was created by Guido van Rossam in 1989 and released as open source in 1991.
- With applications of data mining and data analysis, Python is a premier language of data analysis and the choice of new programmers.
- The Python community is strong and active – Python now attracts more than 2,500 attendees annually (2015), and has over 40 conferences each year around the world.
- Over the past two decades, Guido has worked for many companies that allowed him to remain involved with Python, including Centrum Wiskunde & Informatica, CNRI, Zope, Google, and now Dropbox.

Motivation

- Python features a mixture of readability and practicality – nice features for an introductory language. It is also an interpreted language that encourages experimentation–a great learning aid.
- It has a number of immediately available data structures (strings, lists, dictionaries a.k.a. associative arrays, and sets) with associated functions and methods to easily manipulate those structures.
- It is object-oriented which helps in preparation for both solving complex problems and other languages.
- Python interacts well with other languages which allows one to apply high-level constructs to them.

Motivation ...

- It is a free language that runs under most environments including, but not limited to, Microsoft Windows, Mac OS-X, and Linux.
- It includes many modern programming language features together with a seemingly limitless set of modules that extend it. Those modules come from the large and supportive Python community that has been rapidly growing.
- In short, Python can be described as a "best-practices" language, providing practical tools to do a job with a minimum of effort.

Motivation ...

- Taken together these features allow a novice to focus more on problem solving and less on language issues.
- In addition, the built-in language features make data manipulation particularly easy allowing students to more easily work on real data.
- As a result, not only do students solve more challenging problems, but they also have a tool that can be used in subsequent courses, research or even personal use.
- In an experiment, students have been able to download real data from the Web and analyse it. Examples include building a simple classifier for medical data, online transaction data, and simulating DNA transcription.
- The idea is to grab some real data, parse it into useful form, and analyze it–ideal problem solving skills for scientists.

Evidence that supports the suitability of Python as a teaching language.

- While Java remains a popular choice because of its association with commercial use, the difficulty in learning programming is aggravated by its complex syntax and semantics.
- As such, Java's notational overhead has been criticized as making it unsuitable for novice programmers.
- Python has a simple and clean syntax and structure and other characteristics that make it appealing for teachers and learners, such as dynamic typing, powerful built-in functions and structures, and a simple development environment.

Evidence that supports the suitability of Python as a teaching language.

- Python was designed to have simple syntax and semantics leading to the elimination of the vast majority of errors commonly made by novice programmers, such as missing semicolons, bracketing problems, and variable type declaration errors.
- Python is also increasingly being used in real-world applications (for instance, in high-profile organizations like Google and Nokia.
- Finally, since Python also supports the object-oriented paradigm, it can be used as a transition language for second-term or second-year courses that are based on C++ and Java.

Empirical studies document outcomes of revising introductory programming courses, switching to Python

- Grandell et al. [2006] found that Python facilitated teaching and learning and increased student satisfaction.
- Conclusions were based on analysis of grade distributions, self-reports, identifying the use of constructs in students' code, and surveys of student attitudes toward programming.
- However, despite the multiple measures used, the authors did not provide evidence of statistical or grounded analysis, which may limit the strength of the conclusions.
- Similarly, Kasurinen and Nikula [2007]: moving from C to Python led to higher grades, improved student satisfaction, and a decline in dropout and failure rates.

Empirical studies document outcomes of revising introductory programming courses, switching to Python

- A discussion and evaluation of the "Python first" approach by Radenski [2006]: led to the development of a full online study pack implementing this approach.
- Empirical studies by Patterson-McNeill [2006], Stamey and Sheel [2010], Miller and Ranum [2005], Oldham [2005], Goldwasser and Letscher [2008], and Shannon [2003]: detail the choices involved in redesigning the curriculum and the shift from one language to another in their colleges and argue for dramatic improvements with the use of Python, but no explicit evaluation results are reported in any of these studies.

Empirical studies document outcomes of revising introductory programming courses, switching to Python

- The study involved the analysis of 30 programs written in Java and 30 programs written in Python produced by high school students in 2 different academic years.
- The results revealed that the Python programs contained fewer logic and syntax errors and more frequently fulfilled the required functionality.
- Interviews with eight of the students who had learned Python after Java revealed positive perceptions of the language.

Python vs Java

- Java was used as the introductory programming language for students in many institutions.
- Java introduces basic aspects of programming. However, it may be problematic, not least because Java is heavily coupled with object-oriented concepts, which may interfere with the basic aim of an objects-later strategy.
- Java "forces" some of the more advanced concepts into the foreground–concepts which teachers do not typically want to introduce at an early stage [Kolling 1999].
- As a consequence, a student's focus may switch from learning the basic programming concepts to learning the language's syntax.

Python vs Java

- Drawing on the evidence that Java may not be well suited for education, especially when introducing programming to novices [Siegfried et al. 2008], the use of an alternative programming language with a lower syntactic burden was considered.
- Several educators have argued that scripting languages could offer a more effective alternative. Python is one example, offering a simple and expressive language with support for procedural programming.
- It can be argued that the lower overhead associated with Python should provide a gentler introduction to the basic concepts of programming.
- By using Python, a greater emphasis on core principles was expected with less of an unwanted focus on syntax.

Python vs Java

- As already noted, Python is also widely used in industry and is therefore considered to be attractive to students.
- In the following, a set of programs are written in Java and Python to exemplify the syntactic and semantic differences between the two languages.
- The juxtaposition of these programs suggests that Python has a simple and intuitive syntax. On the other hand, a basic program in Java may expose students to notation and concepts that cannot be understood until well into their study.
- As such, Python is expected to provide a speedier, less overwhelming, and possibly more effective startup platform for novice learners.

Concepts Taught: Variables, primitive data types, and data structures (arrays and lists)

```
Java:
public class ExampleClass
{
public static void main(String[]args)
 {
 int x = 42;
 System.out.println(x);
 }
}
Python:
x = 42
print x
```

Concepts Taught: Control structures: selection

Java:

```
int x = 42;
if (x%2==0)
{
System.out.println(x +
      " is even");
}
else
{
System.out.println(x +
          " is odd");
}
```

Python:

```
x = 42

if x%2==0:
    print x, "is even"
else:
    print x, "is odd"
```

Concepts Taught: Control structures: iteration

```
Java:
for (int i = 0; i <42; i++)
{
System.out.println(i);
}

Python:
for i in range (0,42):
     print i
```

Sub-programs: procedures/ methods

```
Java:
public static boolean
           evenOdd(int n)
{
 if (n%2==0)
 {
  return true;
 }
else
 {
return false;
 }
}
```

```
Python:
def evenOdd(n):
if n%2==0:
    return True
else:
    return False
```

Importing libraries

```
Java:
import java.util.Random;
Random randy = new Random();
int r= randy.nextInt(10);

Python:
import random
r= random.randrange(10)
```