From C++ to Python

Lakoa Fitina

Abstract

The Python programming language is set to become one of the most popular languages in the years to come. This paper describes some features of the language that make it a logical and attractive language to use as a programming language at Divine Word University.

Key words: computer programming language, Python, problem solving

Introduction

Problem solving is an essential skill that a computer scientist must acquire. Computer science students must spend the first two years learning how to solve problems, and how to express the solution to a problem in a sequence of unambiguous statements, called an *algorithm.* An algorithm may be written in an abstract language, called *pseudocode,* or in a computer programming language.

> *A programming language is a language designed to describe a set of consecutive actions to be executed by a computer. A programming language is therefore a practical way for us (humans) to give instructions to a computer.*
>
> Kioskea (en.kioskea.net) 2008

Writing a solution in a computer language and then running the solution on a computer is called *implementing* the solution. Often a solution may be 'stored' in pseudocode form and then translated into one or more programming languages for implementation on a computer.

Choosing a computer programming language in which to implement an algorithm is not necessarily a trivial matter; one reason is that there are so many such languages to choose from. As of today there are more than three thousand programming languages. One reason to choose a particular language is that it is easy to use, simple to learn, but at the same time that the language contains enough facilities for the project in mind.

Some programming languages are more popular than others. The popularity of a language may be measured based on (a) the number of *new* applications written in the language, (b) the number of existing applications written in the language, (c) the number of developers that primarily use the language, and (d) the number of developers that use the language. Some popular languages include C, C++, Java, Visual Basic, PHP, Perl, Ruby, Delphi, C#, and Python (Tiobe, 2009).

Choosing a programming language to teach is often just as difficult as choosing a language to use. For one thing, one must consider the students.

Mathematics and Computing Science at Divine Word University

At the beginning of 2009 the Divine Word University (DWU) established a degree program in Mathematics and Computing Science (hereafter called *the program*); and the program took in its first intake of students at this time. The program consists of four years of study in Pure Mathematics and Computer Science. Successful candidates are awarded the Bachelor of Mathematics and Computing Science degree. Students may elect to exit after two years of study, in which case they obtain a Diploma in Mathematics and Computing Science.

The program contains a sequence of programming units including:

- Introduction to computing science
- Computer programming
- Design and analysis of algorithms
- Object oriented programming
- Programming languages
- Graphics programming
- Human computer interaction
- Software engineering.

Programming therefore forms a substantial component of the program.

The unit Introduction to *Computing Science* is meant to be an introduction to computing science in general, programming being one of the many topics taught. Depending on the lecturer and the types of students the programming component of this unit may be increased or decreased.

The unit *Computer Programming* focuses almost entirely on programming concepts, with an emphasis on algorithm creation and translation from algorithm to computer program. Students write programs that are mostly *procedural*; that is, programs are composed of blocks of code, each block performing a single task.

These two units form the basis for all programming and data structure units in the program. Therefore teaching students the science, mathematics and art of programming is of paramount importance. Just as important as the teaching techniques, is the programming language itself (Reynolds, 2008).

Most novelists use a single language, whereas most programmers use many, quite different programming languages. Moreover, almost every programming language attempts to embody a particular style of programming, so that a broad education requires immersion in a variety of programming languages. However, when learning programming, *students should first learn to program in a single well designed programming language* (or perhaps a small number of stylistically varied well-designed languages) that imposes a minimal number of obstacles to the programming task (Reynolds, 2008).

The question therefore is, which, of the many thousands of languages currently in existence, should we choose?

Choosing a programming language

At Divine Word University, students come with different backgrounds. Those who grew up 'in town' may have had some exposure to computers either at home or in the school they attended. Then there are those who had never seen a computer before, let alone touched one.

Therefore, there are issues that come with teaching programming at DWU that are not obvious in a developed world scenario.

The latter type of student needs to be kept in mind at all times, in the first few weeks of their first semester. The pace must be such that such students are given time to fiddle around with the keyboard until they are sufficiently confident enough to write programs. Extra tutorial classes may be scheduled for such students. But this is neither here nor there with respect to the programming language that one must choose to teach introductory programming.

The author of this article came to DWU in June of 2006, as a senior lecturer in the Information Systems program. The program has a procedural programming unit in year two, and an object oriented programming (OOP) unit in year three, both taught using the programming language C++. In addition, there is a year two unit which teaches aspects of Visual Basic, which the author also took.

He was asked to teach the OOP unit in his first semester. To his surprise, the level of programming skill in the class was less than what one would expect from a third year class doing their third programming unit. Students were still struggling to write code for basic structures like loops, decisions and arrays. Writing one's own classes was a difficult exercise. Obviously there was not going to be much OOP learning that semester. The basics had to be revisited all over again.

The students in the Mathematics and Computing Science program were not much better. Much of the unit Introduction to Computing Science was programming using C++. C++ was chosen because it is used widely in many universities throughout the world. As well, there is an abundance of textbooks and teaching resources one can use with the language.

One can argue that there were various factors contributing to the low-level performance of students in programming, and that may be true. There needs to be an extensive study of this. However the author noticed that students were better able to 'write' code in Visual Basic, especially if the code required manipulation of graphical objects like windows, boxes and various other widgets; things that had shorter development time in Visual Basic than in C++. He thought he should start looking for a programming language wherein students could start programming with graphics in the first few weeks and at the same time would be easy to learn, has easy syntax, and is powerful.

What programming language should be taught at DWU? How does one choose a language? What properties/requirements should a (teaching) programming language have? Requirements for a programming language suitable for teaching have been precisely specified by many academics who reached an agreement that an ideal language should have the following properties (Kölling 1999; McIver 1996).

- Readable syntax, using keywords over symbols wherever possible. A readable program is easier to understand for students, and it is more likely to be correct.
- High level, isolating students from concepts irrelevant at an early stage of programming, for example memory management, execution model, etc.
- Small size, so that the teacher does not have to present only a subset of the language, and the students are not confronted with concepts they did not learn about in class.
- Core concepts presented in a clean and consistent way. The language should use these concepts in the same way as they are being presented in class. In no way should the language dictate what is taught and how.
- There should exist a clear connection between the teaching language and languages used in the industry, facilitating an easy transition from classroom learning to the ‘real world’.

The last point here is the one that requires many compromises on other important features. A number of languages, such as Pascal, Eiffel or Blue failed as teaching tools because of their ‘academic’ characteristics, too distant from languages widely used in the industry.

On the other hand, professional languages like C++ and Java proved to be too complex and difficult for teaching. They simply obscure the key concepts that every first year computer science undergraduate should learn. Some argued that because of their complexity, they should start being taught as early as possible for students to be able to master them by the time they leave university. The problem with this reasoning is that languages used by the industry are constantly replaced by new ones, and hardly any computer professional uses for their daily work the language he or she first learned at university.

One of the most recent ideas on how to get out of this paradigm/language deadlock is to use dynamic scripting languages that can potentially provide the best of both worlds – early exposure to object-orientation without overwhelming students with incomprehensible, overly verbose code. *Python fits the bill.*

The Python programming language

The unit Computer Programming is an optional unit that can be taken in semester two of the year one Mathematics and Computing Science program, in place of an accounting unit. The Computer Programming unit is meant to be taught at a fairly advanced level, if the department feels that a particular class of students is capable of and will benefit from an accelerated unit in computer programming or software engineering. However as the first intake class did not do so well, the department decided to run the unit, using a different language to C++, in order for students to go over the basics of programming again, while at the same time learning something new, like a new language. The author seized on the opportunity to try out Python.

Python was released by its designer, Guido Van Rossum, in February 1991 while working for CWI also known as Stichting Mathematisch Centrum. Python actually got its name from a BBC comedy series from the seventies *‘Monty Python's Flying Circus’*. The designer needed a name that was short, unique and slightly mysterious. Since he was a fan of the show he thought this name was great.

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need programming interfaces. Finally, Python is portable across all major hardware and software platforms. Python is ideally suited for rapid prototyping of complex applications. It is also used as a ‘glue language’ for connecting up the obvious pieces of a complex solution, such as Web pages, databases and Internet sockets. Best of all, Python is easy to use.

Following is a comparison of programs written in C++, Java and Python. The programs take two inputs (integers) from the user, add the numbers, and then print out the answers.

C++

```
#include <iostream>
using namespace std;

int main()
{
    int A, B;
    cin >> A >> B;
    cout << A+B;
}
```

Java

```
import java.io.*;
public class Addup
{
    static public void
main(String args[])  {
        InputStreamReader
stdin = new
InputStreamReader(System.in);
        BufferedReader console
= new BufferedReader(stdin);
        int i1 = 0,i2 = 0;
        String s1,s2;
        try {
            s1 =
console.readLine();
```

Python

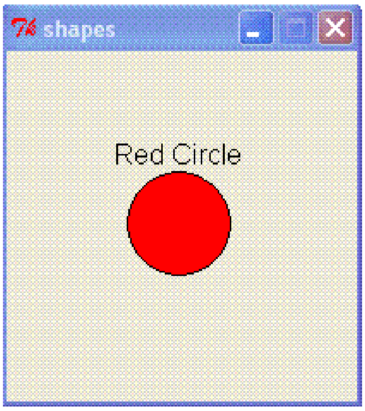
```
a = input()
b = input()
print a+b
```

Clearly the program in Python is the shortest and takes less time to write. The syntax is considerably easier to understand than either C++ or Java. The C++ syntax is intimidating to first year DWU students; but the Java syntax would be simply baffling. In addition to easy syntax, it is very easy to program graphics with Python. In C++, doing graphics in a first year course is ‘uncommon’. One reason is that C++ does not have a default graphics library (Gibbons, 2002). The following Python program draws a circle inside a window:

The program

```
import java.io.*;
public class Addup
{
    static public void
main(String args[])  {
        InputStreamReader
stdin = new
InputStreamReader(System.in);
        BufferedReader console
= new BufferedReader(stdin);
        int i1 = 0,i2 = 0;
        String s1,s2;
        try {
            s1 =
console.readLine();
```

The result



Writing programs to make things ‘move’ around the screen is a good way to learn ‘loopy’ structures like the ‘for’ and ‘while’ loops. Again Python makes this very easy. Students can experiment with 3-D graphics as well as 2-D. Python has modules for incorporating multimedia objects like sound, images and movies. Students can experiments with manipulating the objects and in the process learn about the properties of these objects.

Python is growing in popularity (Goldwasser, 2008) and is likely to become the language of choice for more and more institutions throughout the world. DWU may be the first university in PNG to adopt the language. It may also be the first to benefit from the simple, easy to use but very power language.

References

Gibbons, Thomas E. 2002, *Using Graphics in the First Year of Programming with C++*, *http://www.cs.uni.edu/~fienup/mics_2002/proceedings/papers/gibbons.PDF*

Goldwasser, Letscher 2008, *Using Python To Teach Object-Oriented Programming in CS1*, PyCon http://euler.slu.edu/~goldwasser/publications/pycon2008.pdf

Kioskea (en.kioskea.net) 2008, *Programming Languages*, http://en.kioskea.net/contents/langages/langages.php3

Kölling, Michael 1999, *The design of an object-oriented environment and language for teaching*, Phd thesis, University of Sydney.

McIver, Conway 1996, *Seven Deadly Sins of Introductory Programming Language Design*, Proceedings of the 1996 International Conference on Software Engineering, Education and Practice.

Reynolds, John C. 2008, *Some Thoughts on Teaching Programming and Programming Languages*, Proceedings of the SIGPLAN Programming Language Curriculum Workshop, *SIGPLAN Notices*, 43(11), November 2008, pp. 108-110.

Tiobe 2009, *Most Popular Programming Languages*, Tiobe.com

Author

Lakoa Fitina
Senior Lecturer
Head, Department of Mathematics and Computing Science
Divine Word University
Madang, Papua New Guinea
Email: lfitina@dwu.ac.pg