ADOPTION OF PYTHON IN ARTS FACULTIES OF SRI LANKAN UNIVERSITIES

Sabaretnam Sajiharan
IT Unit
Faculty of Arts and Culture
South Eastern University of Sri Lanka
sajiharans@seu.ac.lk

***Abstract*:**

A variety of programming languages are used to teach fundamentals of programming in Universities in Sri Lanka. Among them Python is a modern language with readable and clean syntax. Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Its design was informed by experiences with other teaching languages so it is considered suitable for such use. However some universities teach PASCAL which is rarely used now. In this research I will discuss the possibility of replacing this Pascal with Python.

Keywords: *programming, modern language, automatic memory management*

Introduction

There are 15 state universities in Sri Lanka. Among them most of the universities have the Faculty of Students of Arts and Culture are following many subjects like Economics, Geography, Sociology, Political Science, History, Tamil….. And also most of the students are studying Information Technology.

One of the main modules of Information Technology is "Introduction to Computer Programming". Most of the universities are teaching this module using PASCAL. The introduction to computer programming forms a compulsory element of the practical course. In the computing course, They have to working through a handbook which aims to provide an introduction to the environment and sufficient exercises such that they are able to write short, procedural programs in Pascal.

Pascal is a commonly used teaching language, it is widely acknowledged as being little more than a toy language. Its deep-rooted problems were described by Brian Kernighan in his paper "Why Pascal is not my favorite programming language.

Pascal's fate as a serious programming language appears to have been sealed. It is now rarely used, even in academia. Having been the standard language for the

"Advanced Placement" college entrance exams in the USA for nearly 20 years, it was replaced by C++ in 1999 (which, incidentally, was almost immediately replaced with Java).

Perhaps Python provides an alternative to Pascal that, as well as being suited to teaching programming fundamentals, is also considered more than a toy by the commercial and academic world.

Research question

1. Whether the use of Python as the principal teaching language of Arts faculties of Srilanka Universities?
2. Python's design and implementation made its use preferable to other languages, in particular Pascal?

Objectives

1. To identify Programming Languages
2. To identify the suitable programming language to Students of Arts faculties of Srilanka Universities?
3. To develop web based, Learning management system for that identified programming Language.

Research methodology

For addressing the research questions and objectives of this study, the approach used is exploratory approach. Literal review and survey is used as a research strategy in this paper. First literature review was conducted to understand the basic concept of programming languages. On the basis of literature review an online survey was conducted from selected students who have the knowledge in PYTHON and PASCAL.

Literature review

A literature review is a critical essay which summarizes the related information about the research question by referring the secondary sources such as analysis, evaluation of the original information etc. The preparation of literature review provides an opportunity to evaluate the knowledge about the research question which helps us to develop a framework to continue our research methodologically.

a. History of programming

This introduction is based on the book "Fundamentals of Programming Languages" by Ellis Horowitz, and is, in an extent, back translated from a Greek translation of the original.

The first known algorithms were discovered by archaeologists to be written on clay tablets. Those tablets are dated between 3000-1500 BC and were found in an area close to Babylon in Mesopotamia, which is located near to present times Baghdad, Iraq.

Babylonians are known to have had a peculiar arithmetic system based in the number 6020. On top of this they had invented floating-point numbers; the latter is the way that numbers with decimals are often called in computer science. To illustrate an example the number 8, 50, 36 can correspond to 31836 or 530.6 or generally to 31836*60^k.

Babylonians did not only write computational tables to facilitate arithmetic operations, they also could solve algebraic expressions using an algorithm that would compute it. The algorithm was a generic method of solving a particular category of problems following various steps and ending with a note like "This is the procedure". In fact, this has a lot of similarities with the way most programs are written nowadays. There is an article by Donald Knuth where some of these algorithms are described. Albeit Babylonians had invented an elaborate, for those times, symbolic system, not much evolution happened in computational methods since then, until the 19th and 20th century.

We may find algorithmic procedures being introduced later, most known being the Euclid's one for computing the Greatest Common Divisor and Eratosthenes' Sieve which is a technique to find the prime numbers up to a certain limit with least computational effort. These cases are dated 1500 years after the Babylonians and the most important fact is that they were isolated examples expressed in natural language, which made no significant advancements towards a formal symbolic representation of algorithms.

The next conscious effort in programming was triggered by Charles Babbage (1792-1871) who designed two mechanical computational machines during the years 1820-1850: the Difference Machine which was based on finite differences theory and the Analytical Machine which had a lot of similarities with a modern digital computer.

Ada Augusta, Countess of Lovelace and daughter of famous poet Lord Byron, was the programmer of Babbage's analytical machine. She is often recognized as the first computer programmer ever.

Babbage and Ada had identified the great capabilities of their inventions -early forms of hardware and software- and in an effort which paved the way of the research field of "algorithmic analysis", commented:

> *In every computation there is a wide spectrum of possibilities for task succession, and multiple investigations must influence the choice of the*

applicable method for a certain computational machine. It is important that the method chosen will tend to minimize the computational time.

b. The revolution of modern computer languages

The first computers could only understand machine language, which is actually a set of 0s and 1s:

```
01000010
11010100
00010001
00010100
10001001
```

Figure 1:First computer language codes

For example, this could be a valid program adding numbers 212 and 20, supplied in binary format. The way this was implemented was by setting a row of switches, set them up and down, press another button to proceed to the next command, repeat for each command, then press the "execute button" to retrieve results on a row of lamps or a printed paper. Albeit tedious, this has been the way first computers worked and the first programs were written.

If someone had to calculate something more complicated than an addition or a multiplication, like a division, he would probably have to break down the work into discrete steps and code these on the computer. This should be done each and every time a new need occurred and, indeed, it happened. Of course, it was necessary to automate this process a little more and be able to generate the necessary code simply by stating a mathematical expression. There was a need for language compilers, in order to have some abstraction over the machine and focus on the problem at hand rather than the peculiarities of the system.

The first language compilers were developed for use in scientific and commercial applications during the 50s and 60s. The vast majority of those were written in FORTRAN and COBOL, the former being the standard in academics and the latter being the standard in business applications. Soon after, there was ALGOL, PL/1 and BASIC. Then, there was ALGOL68, PASCAL, MODULA, C and others. All these were imperative languages, a programming style based on the principle that everything is a sequence of commands, which are executed one after the other:

c. Pascal

Two of the members of the team working on ALGOL were Niklaus Wirth and C.A.R. Hoare, who did not agree on the upcoming ALGOL68 definition and found it too bloated and unsuitable for further development. In 1968, Wirth started designing a variation of ALGOL, named after the French mathematician and philosopher of the 17th century Blaise Pascal, for use in teaching institutions and academic environments. Two years later, PASCAL had combined the best features of the languages in use at the time COBOL, FORTRAN and ALGOL. It was a general purpose language that promoted boldly the structured programming style

and could be used in a range of applications without difficulty, which is why PASCAL is still so popular

d. Python

Research in programming languages didn't stop and in the 90s more languages had appeared, each having a different priority in mind. One of them by that moment was ABC, an evolved and indentation-structured form of BASIC, mainly designed for use in education. Guido van Rossum was working at this time at the Amoeba Distributed Operating Systems group for the Centrum voor Wiskunde en Informatica, and contemplated an extensible scripting language with ABC's easy on the eyes syntax, while having good facilities for calling and cooperating with other programs. He considered a few ideas found in C, C++, Icon and Modula-3, and started writing Python.

Python is a Very High Level Language and an Object Oriented Dynamic Language. It concentrates the experience of multiple years of work in Computer Science and incorporates ideas found in imperative, object-oriented and functional programming paradigms; it does include exceptions, modules and classes. Python's object model is easy to use. It is interpreted and sacrifices some performance in order to maximize development speed. It can call other programs and can be embedded in other programs, which makes it ideal as a scripting language within bigger packages.

The first release of Python in the public domain happened in 1991. The version of the language that is mostly found in various systems is v1.5.2.

Python, contrary to LOGO and ABC which are also educational languages, is not considered at all a toy language: it is not only designed with educational requirements in mind, it also has properties that make it stand high next to professionally widespread languages like C and Perl.

Last but not least, it is available for free. Anyone can download it from the Internet, install it, give to a friend, a student, a teacher and this is perfectly legal. The language is supplied together with source code and documentation, which means that it is possible to see, what are the internals of it?, if you are interested and even contribute in its development or steer its direction. There are also plenty of Python programs on Internet sites, available for download.

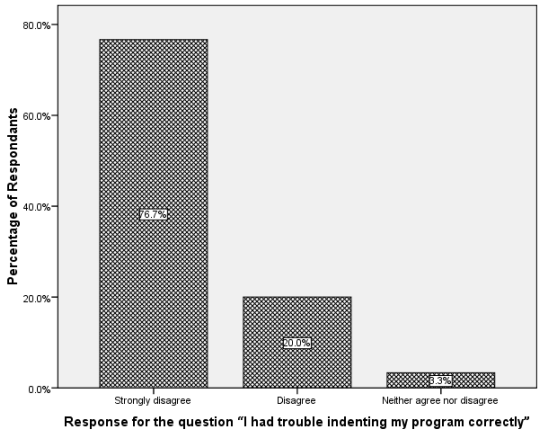
Research Findings

a. Results and Analysis

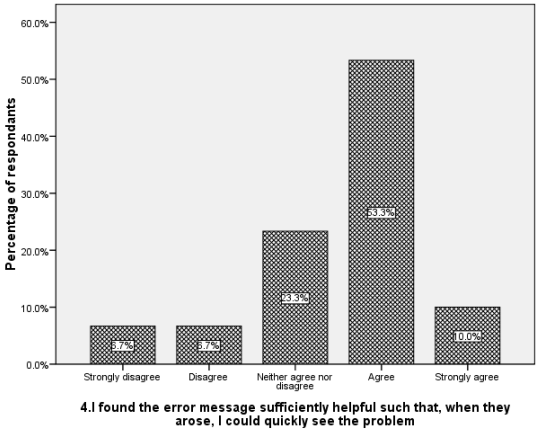
In this phase I have to show the results gain from the above communication methods. So I can deliver it few ways. Here are few responses, which I received from the students.

b. The students' reaction to Python after completing survey

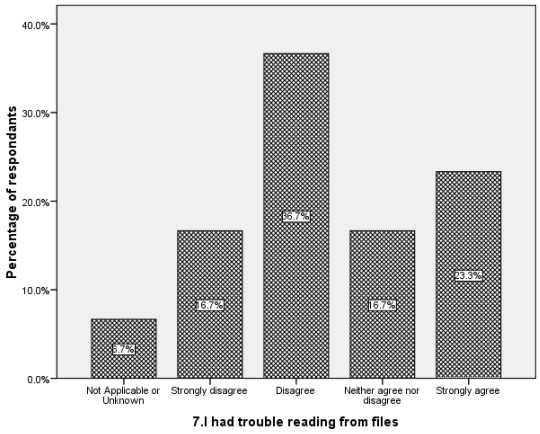
1. I had trouble indenting my program correctly (No students claimed to have trouble indenting and most said they understood its significance)



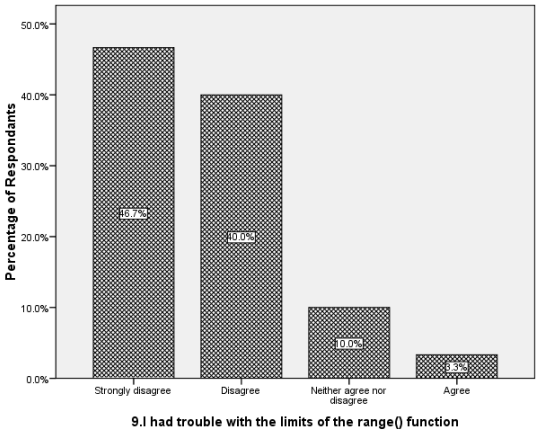
2. I found the error message sufficiently helpful such that, when they arose, I could quickly see the problem (Sixty four percent either agreed or strongly agreed with the statement)



3. Reading and writing files(Only 24% of students found the problems on reading and Writing files)



4. Trouble with the limits of the range() function(problems with the range function itself were surprisingly rare (4%))



5. Python compared to Pascal (When asked to compare Pascal and Python 74% of students expressed a preference for Python)

c. From Literature Review

Python has the following characters

- Portability - There were no problems compiling Python
- Speed with which it can be taught - It is possible for students to quickly write (and understand) basic programs
- Integrated Development Environment IDE – There must be a simple, ideally graphical IDE

- Individualities-It has no any unnecessarily complicated constructs whose use is required to undertake common programming tasks
- It is Free (as in both cost and source code)
- It is trivial to install on a Windows PC allowing students to take their interest further. For many the hurdle of installing a Pascal or C compiler on a Windows machine is either too expensive or too complicated
- code is smaller, less chance for errors
- easier to write, faster development
- resembles a lot the pseudo code, pupils can write code immediately
- It is a flexible tool that allows both the teaching of traditional procedural programming and modern OOP
- It is a real-world programming language that can be and is used in academia and the commercial world
- It appears to be quicker to learn and, in combination with its many libraries, this offers the possibility of more rapid student development allowing the course to be made more challenging and varied
- and most importantly, its clean syntax offers increased understanding and enjoyment for students

Conclusion

From the above features of PYTHON and Results of questionnaires we can come for a conclusion of

"PYTHON is possible and desirable as a programming language to the Arts faculty Students in Srilanka Universities."

References

1) Coombs M.J., Gibson R., Alty J.L, Learning a first computer language: strategies for making sense, International Journal of Man-Machine Studies, 1982, Volume 16, pages 449-486. Academic Press
2) Guido van Rossum, Computer Programming for Everybody: A Scouting Expedition for the programmers of tomorrow, CNRI Proposal #90120-1a,July 1999 http://www.python.org/doc/essays/cp4e.html
3) Dijkstra W. Edsger, Go To Statement Considered Harmful, Eindhoven University of Technology, Communications of the ACM, Vol. 11, No. 3, March 1968, pp. 147-148
4) Clive D Rodgers et al. Handbook of the physics computing course.
5) Brian W. Kernighan. Why Pascal is not my favorite programming language. Computing Science Technical Report 100, AT&T Bell Laboratories, 1981.
6) Python FAQ. http://www.python.org
7) http://pentangle.net/python/
8) The IDLEfork Project. http://idlefork.sourceforge.net/

9) Numerical Python. http://www.pfdubois.com/numpy/
10) Gnuplot.py. http://gnuplot-py.sourceforge.net
11) The great computer language shootout. http://www.bagley.org/ doug/shootout
12) Green T.R.G, The Nature of Programming, Psychology of programming, 1990, Academic Press Ltd., 24-28 Oval Road, London NW1 7DX, ISBN 0-12-350772-3
13) Hoc J. M. Role of mental representation in learning a programming language, International Journal of Man-Machine Studies, 1977, Volume 9, pages 87-105. Academic Press
14) Lutz Prechelt. An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string- processing program. Technical Report 2000-5, Universita¨ Karlsruhe, Fakulta¨t fu¨ r Informatik, Germany, March 2000. http://www.ipd.uka.de/ prechelt/Biblio/.
15) The Astronomical Python home page. http://lheawww.gsfc.nasa.gov/ bridgman/AstroPy/
16) Python for Sciene. Molecular Modelling at the University of Grenoble. http://starship.python.net/crew/hinsen/