Choosing a Programming Platform

Diane Hobenshield Tepylo, Lisa Floyd, and Steve Floyd
(Computer Science and Mathematics teachers)

The Tasks Working Group had many questions and concerns about choosing a programming language including:

- What factors influence our choices of language and why?
- What should be the role of block programming?
- For whom are certain languages best suited and why?
- For what are certain languages best suited and how do we best take advantage of these for teaching / learning purposes?
- How to help prepare newbies to vet coding languages for their future students (at all levels, K- to university

As computer science teachers, we have had to choose programming languages periodically over our careers. Here we will discuss some factors we consider when making decisions, and discuss some possible options: Scratch, VB, Python and Java.

Factors to Consider in Choosing a Language

Teacher's Comfort Level
In order for the teacher to make appropriate decisions in their planning based on pedagogically sound principles, they will need to have a solid understanding of the potential benefits/drawbacks and what the language has to offer.
(putting a math and coding slant on our comments to the best of our ability). While we each teach several programs, we suggest starting with one language, developing competence and confidence before adding another language.

Availability of the Program
A free version is available of each (see individual programs for links).

Note: To get a program installed on a school network often takes 4-6 months, after an application and approval by a central IT department. Also many school networks have older machines and operate on older operating systems, so the programming environment must work well on that technology.

Most schools now have reliable Internet service so online languages such as Scratch should be available when you need them. However with Scratch's growing popularity the concern was raised as to how many users it can support.

A basic version of Python is available online- http://cscircles.cemc.uwaterlorandom and matho.ca/console/ which allows one to get started right away and even allows one to import modules such as math and random. However, there is no option for interaction with the user, file input/output, graphics, or sound in the online version. To take the full advantages of Python and its wide variety of modules, Python should be installed.

Python is now on more and more school networks making this easier. Ideally additional modules should be installed with the program in its libs, but these module can be saved in the same folder as the current program and can be imported.

Visual Basic Express only works when installed on a Windows computer.

Availability of Language Out of School
Scratch, Python and Visual Basic Express are freely available to students at home?

Availability of Resources for Level of Students
It can take substantial time to create all the resources for teaching, and if you can find some appropriate teaching resources- even if you need to change them for your purposes- this can save substantial time. Scratch have great resources for teaching younger children and Python has great resources for teaching secondary CS. See below for more details.

Purposes in Teaching Coding
Language has to be consistent with the end game of the teacher… see descriptions below

The teacher will need to condider what the end goal is….
Is it to develop– thinking skills, abstract concepts?
-to prepare them for postsecondary education?
-to prepare them for potential careers?
-to use as another tool to be creative?
-to use as a tool to communicate for other classes, a new creative outlet to produce something?

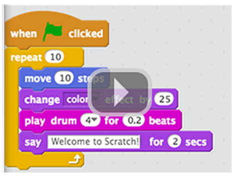
Individual Languages

Block Programming: for example Scratch

Scratch is designed for ages 8-16 but educators have used it with younger children. It is easy to access, easy to use and has lots of resources created for educators.

From our experience, block-programming environments are less intimidating for many students than line programming. Block programs with their drag and drop blocks avoid many syntax errors such as spelling and punctuation mistakes. We have found Scratch to be a wonderful learning tool for elementary school level students. We have watched as students have learned complicated concepts such as lists and counted loops with ease.

The design of block programming environments also allows students to quickly create event-driven programs where a click on a button performs a designated task. This is motivating for students but hides many important programming concepts which could raise concerns for more advanced students.


```
when clicked
repeat 10
    move 10 steps
    change color effect by 25
    play drum 4 for 0.2 beats
    say Welcome to Scratch! for 2 secs
```

However, Lisa uses Scratch as an introduction to programming. She finds that it helps students learn complicated programming concepts in another language later in the course: the students can visualize structures such as loop having seen physical, colour-coded versions of these concepts in Scratch.

A list of other Block-based programming languages is found at http://wiki.scratch.mit.edu/wiki/Alternatives_to_Scratch

Visual languages such as Visual Basic (VB)

Visual Basic Express is free as part of Visual Community 2013
https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx
From the site "Visual Studio Community is a free and full-featured IDE for building apps for Web, Windows Store, Windows Desktop, and even Android and iOS using programming languages including C#, C++, HTML/JavaScript, and Visual Basic." It is also used in many businesses.

From our experience, visual programming languages less intimidating for many students than pure line programming. The drag and drop design of visual programming environments allow students to quickly create event-driven programs where a click on a button performs a designated task. One advantage for using VB is that it allows for students to make professional looking forms right away without much coding experience. Often, the more visually artistic learners appreciate that they can adjust colour, backgrounds, images quite readily in the IDE and then when it is time to code, they can focus on the real programming concepts.

If the teacher wants students to be excited about graphics, VB gives them the hook that they need right away….Something for them to see, visually… may seem gimmicky, but important. Buttons seem powerful and students are amazed. Because students attach code to buttons and text boxes, students are writing short bits of code, which is easier and less intimidating than the long text of a similar program in a pure line based coding environment such as Python or Java. This gives students exposure to modular programming right away, and can see that the sections makes up a whole….. this supports decomposition, and breaking problems down into smaller, more manageable parts (an important programming concept) .

VB does allow students to learn variables, strings, decision structures, operators but it hides many important programming concepts such as loops (their usefulness is not obvious in VB). Additionally, it is somewhat confusing to obtain information from the user in VB. They have to assign information typed into textboxes to a variable so students do not learn the variable concept as readily as in a line-based program such as Python. The students eventually have no problem with variables in VB, but it takes a few lessons before they start assigning values rather than just using the textboxes for calculations and such. Lisa thinks Python might be better for math learning if students don't have much of a background in programming but Brock University uses VB in their math program.

While there are a fair number of resources out there for learning VB, I have not found many aimed at K-12 teachers.

Python
Python has the advantage of being easy to learn, used authentically in many businesses and research and very powerful given the libraries that are written for it. Python is open-source (free). Simple programs can be run on line (http://cscircles.cemc.uwaterloo.ca/console/) but for more complex programs with interaction and graphics, Python needs to be installed.

Python has versions that are available for Windows, Mac and Linux operating systems. It has numerous libraries (free) that perform specific functions including Numpy/Scipy for numerical operations (parallels to Maple and MatLab), IPython for interactive work, and MatPlotLib for graphing. iPython notebook is a free web application that allows the user to run python programs, keep texts and multimedia notes and rich media as well as keeping the results of program runs.

There are two major versions of Python to consider: 2.7(2.7.10 is the latest version) and 3.5. As an open-source environment, Python is developed in pieces by individuals and/or teams and not all libraries are available for each version. The decision about which of the major versions that you want to is dependent on the libraries you wish to use. Both versions can be on one computer simultaneously without causing problems

In my limited experience, I have found that 2.7 works best on Windows XP machines and with the Pygame library (which renders graphics for smoother running of games- but not for beginning programmers). 3.5 cleans up a few issues from 2.7 (see https://wiki.python.org/moin/Python2orPython3)

A great way of getting numpy and scipy without having to install each library is to install Anaconda- free (https://store.continuum.io/cshop/anaconda/). They have versions for Mac, Windows and linux. The Anaconda on my Mac OSX laptop is using Python 2.7.

Spyder has also been recommended to me (https://pypi.python.org/pypi/spyder) . It is an integrated development environment that they compare to MATLAB. Apparently it can be tricky to install on a MAC and I haven't attempted this yet. However as with most open-source projects, solutions appear quickly and I just found one (https://pythonhosted.org/spyder/installation.html)

There are lots of resources for Python and students, most with a CS focus but great places to get started.
University of Waterloo's CS Circles http://cscircles.cemc.uwaterloo.ca/0-introduction/
Harrington Python Tutorial http://anh.cs.luc.edu/python/hands-on/

> It has both 2.6 and 3.+ versions
> Download the zip file with sample code for the version you are using and save the graphics library to Python's lib directory or the same folder as your current program. It is a great way to strengthen understanding and skills with x-y plane.

Online Books with programs for games

http://inventwithpython.com/chapters/

In the resources section, Diane will also include a series of Python lessons with lots of exercises (field tested once with a ICS 3C class)

At the symposium, we learned that Noss and Hoyles team is working on a visual version of Python. When this happens, it will likely gain many of the advantages of visual programing discussed above.

Java

Java can do complex programming, authentically, while some languages have been 'bent' to do certain things. Java will also work on any system. However the syntax is complicated and not considered a good first language by many. Having said that, a lot of universities use it in their first course for CS majors.

If the teacher is trying to learn computational skills, then there is no point in learning a 'slow to start - but awesome to end language'. In this case, the teacher will want a 'quick to start language' such as Python.
Java may be slower to start, but in the end, much more powerful.

For a teacher who is just starting out with programming, unless they have an extensive coding background, they might not want to start with Java… too overwhelming. It is one of the most widely used languages, but it can be difficult to learn, mostly because the syntax can be intimidating.

Language has to be consistent with the end game of the teacher… if in-depth, object oriented, modular, then one will need to use Java, VB.

For perspective, these are the languages we teach

Diane	Lisa	Steve
will start with Scratch mini-unit ***Grade 11 &12 U*** – started teaching Python 2015 - for 2015-2016 will add a tangible TBD (arduino, raspberry Pi or Lego robot) programed in one of the above languages. ***11 & 12C*** - Python, VB, Javascript, and a TBD tangible (used to teach Turing and Java)	Starts with 2 weeks Scratch ***Grade 10*** – VB ***Grade 11 C and U*** – Turing (moving to Python) as well as adding tangibles this semester – Arduino, Lego EV3 **Grade 12** – Java (might use Python as well)	***Grade 10*** – VB ***Grade 11 C and U**** – Python **Grade 12 C and U*#** – Java, Some Javascript, Python with Raspberry Pi *Course includes a Discovery Week, students take a week to try to teach themselves a new language #Course includes presentations where students present to the class a variety of languages