Author & Date:

Matt Schuchhardt 1/7/11

Topic/Concept:

Introduction to programming, programming languages

Type of Activity:

Lecture + small groups + demonstrations

Prerequisite knowledge:

None; an open mind and a willingness to learn

Resources required:

Display computer, computers for students to work on w/ Python installed

Tetris source code

How to Think Like a Computer Scientist: http://openbookproject.net/thinkcs/python/english2e/ch01.html (not used directly in lesson, but helpful for both students and teachers)

Objectives:

Students will understand what a computer program is

Students will understand the role of different kinds of computer languages

Students will understand that no computer language is the best for every situation

Common Misconceptions:

Programming is hard

Only certain people can be computer programmers

Detailed Description of Activity:

Introduce students to programming

First ask the students if they play video games (just to gauge interest, spark excitement), ask if the students if they have ever played Tetris.

Play a 30-second demonstration of Tetris on the display computer (to give a quick refresher on how the game works, different elements that the game has in it). Stress the different aspects that the game has (score keeping, rotating blocks, clearing lines out, etc.) The attached Tetris.zip should run as a Python program (just use another Tetris game if you can't get it to run; as long as you have the source code, you should be well-equipped enough to do the lesson).

Break the students into pairs/groups for two minutes, and have them list as many of the program's attributes/components that they can (for example: the game keeps score; the game has a timer; the game clears a line every time it's full; the game sends a different random block every time a previous one is placed). It is important in programming to think modularly and not monolithically when programming, so make them get as specific as possible with the various functions of the game.

Show them the source code for the Tetris game: start with showing the whole program's code, but then explain that none of this is over their heads: they just haven't learned this specific language yet. Explain that learning the language of computers is just like learning a different language: it seems difficult when you first start, but the more comfortable you are with the new language, the more useful it becomes.

From the lists that they created as groups, start identifying the various components in the source code (e.g., find the code which rotates a block, etc.) (this will require some knowledge of the source code, so be sure to go over this in advance). Show that although they may not understand the code's language, just by reading through the function names can give them a good idea how a program works. Make this as absolutely basic as you can, but keep it interesting; you don't want to scare them (yet!).

<u>What a computer program does</u>

One of the biggest obstacles that I encountered while teaching this was that the students didn't actually know what programming was: they thought it was just like using another computer program (like Word or Excel), and not that they were actually writing their own programs.

(This is a quick, simplified version of HTTLACS 1.1; this is a great reference!).

Describe programs as a black box: inputs, calculations, outputs. All that a program language is a list of simple calculations which take an input, somehow use that input, and produce an output.

Have a discussion on what inputs, outputs, and calculations are. It can be abstract as in data -> calculations -> output, but feel free to get very concrete with what input and outputs are: for example, explain that even a toaster has a very small computer (microcontroller) inside of them: the toaster takes the button press to start toasting as an input, it calculates how long to toast the bread, and then it produces some output (by turning on the coils for a certain amount of time). There are millions of examples like this.

Again, stress that programming *creates* programs like Word and Excel (and Tetris!): you are

specifically telling the computer to do calculations, not simply using it as a tool any more.

Computers are DUMB: be smarter than the computer! It will do exactly what you tell it to (as opposed to simply using a computer program), which can be good, but can also make it hard sometimes. Computers can run millions of calculations every second: they DUMB and FAST: they can (simply put) only do basic arithmetic, but they do these so fast, that you can build very complex calculations by using millions of simpler calculations. You are just telling the computer how to do these simple calculations, and how to store and process that data.

You may spend some time here discussing the wide variety of computers that exist: anything with a blinking light that runs on electricity has some kind of a computer inside of it: alarm clocks, toasters, Mp3 players, etc., etc., etc....

Introduce students to different programming languages

Ask students to name off some sorts of things that had a program written for them at one point in time, identify languages

Main point: No programming language is the 'best': each language is good at different things.

Php: an internet programming language. Google, Facebook, and Amazon all are programmed at least partially in Php. Very good language to start with.

.NET: Microsoft's main programming language. Windows 7 applications and Xbox360 games are programmed in this language.

Python: good at quickly doing calculations on a computer, relatively fast to program in.

Mindstorms: Very good at programming lego robots (my class dealt with these previously)

Again, severely stress that although these languages are good at some things, they may not be good at others. It's not important at all to remember the specifics of these languages, but make them realize that no single language is 'best': just good for different applications. You wouldn't program a computer in mindstorms, and it would be very difficult to program a lego robot in Php.

Rubrick

	Emerging Understanding 1	Working Knowledge 2	Concept Mastery 3
Concept of Programming	Think that programming is just like using any other computer program, and that programming is just another tool	Simply understand that programming is more powerful than using computer tools	Fully grasp the concept that programming is the computer language used to create other computer programs, and can be as powerful as you need it to be

Role of Computers	Think that computers are inherently 'smarter' than humans	Understand simply that computers are very fast	Understand that computers merely run simple calculations very quickly, (not inherently smart), and that a good programmer is able to be smarter than the computer
Variety of Computers	Computers always have a keyboard, monitor, and mouse	Computers are very common, but don't believe that they are as widespread as they are	Computers are EVERYWHERE: in your car, bedroom, classroom, even bathroom.
Source Code Understanding	When analyzing the components of Tetris, identifies very broad, non-specific aspects of the game (a Tetris game lets you play Tetris)	Description of the individual components are somewhat more specific, but the components are still very monolithic (scoreboard, timer, game window).	Descriptions of the components of Tetris are extremely specific, and understand how the components relate to the source code's representation of them (high-level overview)
Programming Languages	There is a 'best' computer language	Some languages are good at certain things, but don't fully grasp the fact that different languages exist because of the wide variety of programming applications	Understand that due to the wide variety of uses that computers have, it is necessary to be able to program in a wide variety of different languages, and each language has its own intended application.