DSC 510

WEEK 1

Computer

A computer is an electronic device which takes input from the user, processes the input, and outputs the results for the user.

Computers include; cell phones, iPods, and Tablets.

Input consists of any form of instruction or data the user provides the computer.

Process consists of processing instructions and data and storing results

Output consists of displaying the stored results or printing the output.

Computer Hardware

A computer system consists of multiple pieces of hardware that allow them to function

Central Processing Unit (CPU) - The brain of the machine. This is where all the basic operations of the computer are carried out. The CPU tells the rest of the computer what to do.

Memory - The memory of a machine stores programs and data.   Computers have two different types of memory: main and secondary memory.

Input/Output devices - Input and output devices all users to interact with the computer. Examples of Input devices are the computer mouse and keyboard. Examples of output devices are the computer's monitor. Information from input devices is process by the CPU and may be stored in main memory or secondary memory such as RAM.

Main Memory

Considered the computer’s work area

Computer stores the program that is running as well as the data

Commonly known as the ***random-access memory (RAM)***

Data is quickly accessed

RAM is a volatile type of memory

Used for temporary storage

RAM is erased when computer is turned off

Secondary Memory

Type of memory that can hold data for long periods of time.

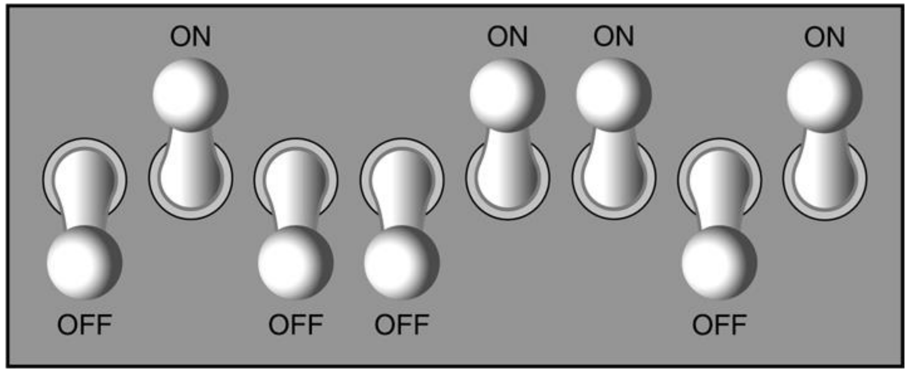
Programs and important data are stored in secondary storage

Disk drive is a common type of secondary storage

- Data is stored by magnetically encoding it onto a circular disk
- Most computers have an internal disk drive
- Some have external disk drives; they are used to create backup copies

Storing Information

- A computer’s memory is divided into tiny storage locations known as ***bytes***
- One byte represents one number
- A byte is divided into eight smaller storage locations known as ***bits (binary digits)***
- Bits are tiny electrical components that can hold either a positive or a negative charge.
- A positive charge is similar to a switch in the *on* position
- A negative charge is similar to a switch in the *off* position



Storing Numbers

- The positive charge or the *on* position is represented by the digit 1
- The negative charge or the *off* position is represented by the digit 0
- This corresponds to the binary numbering system where all numeric values are written as a sequence of 0s and 1s
- Each digit in a binary number has a value assigned to it

For more information on binary visit: https://www.codeproject.com/Articles/4069/Learning-Binary-and-Hexadecimal

Computer Program/Software

A computer program is a sequence of instructions that are processed by a computer’s CPU and tell the computer to perform a specific set of tasks. These instructions must be in a language that the computer can understand (machine code). In order to get these instructions into machine code a software developer writes a source code in the form of a program using one of many programming languages such as Python. When source code is written it must follow rules specific to the language called syntax.

Two categories of software

- System software
- Application software

Compiled and Interpreted Programs

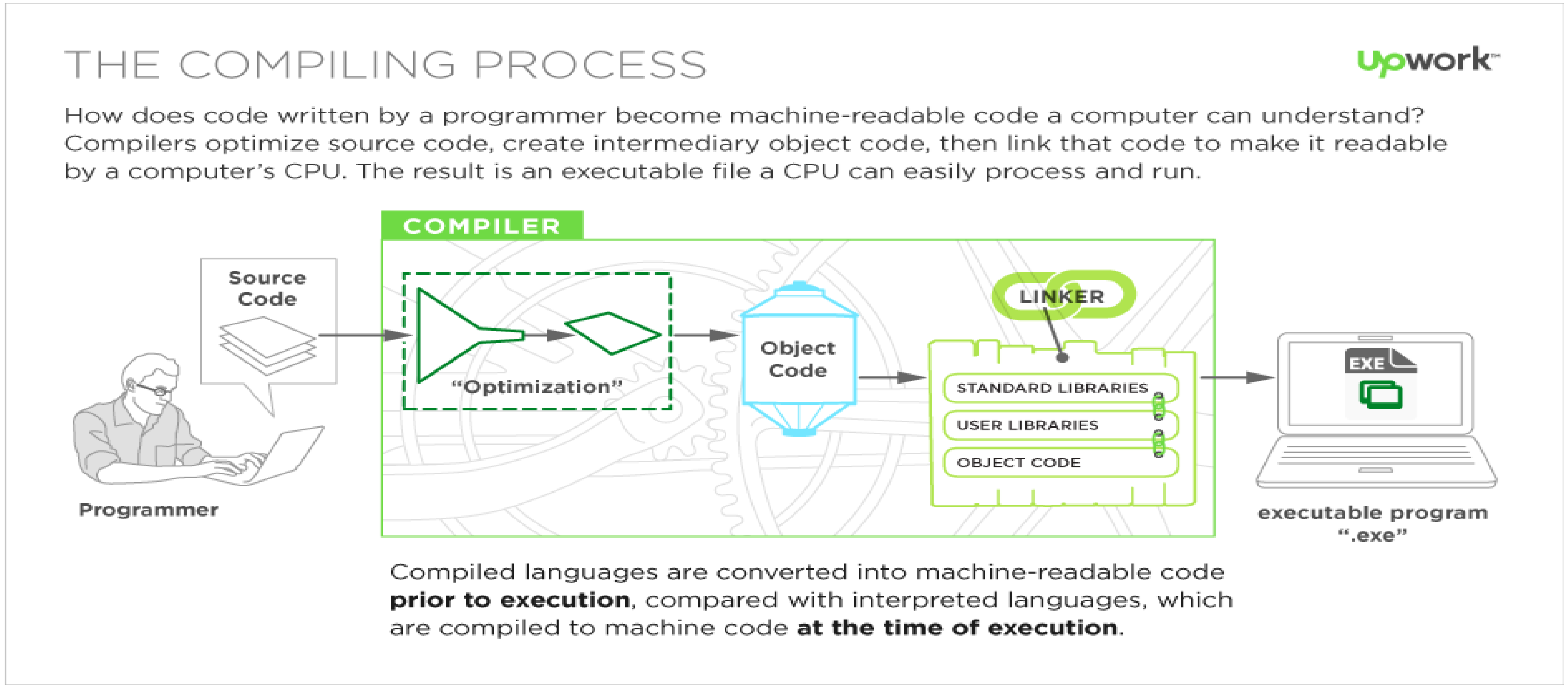
Compiled Languages - Compiled languages require a compiler in order to convert source code into a computer’s native language (machine code). Once the code is compiled the application can be executed on the chosen platform. With compiled languages the source code only needs to be compiled once and then the application can be ran multiple times. In most cases compiled languages execute faster than interpreted languages.

C, C++, Java, ASP.NET

Interpreted Languages - Interpreted languages use an interpreter to convert source code into machine code one line at a time when the application is executed. Each time the program is run the source code is interpreted.

Python, Perl, Javascript

Compiling Process



THE COMPILING PROCESS

How does code written by a programmer become machine-readable code a computer can understand? Compilers optimize source code, create intermediary object code, then link that code to make it readable by a computer's CPU. The result is an executable file a CPU can easily process and run.

Compiled languages are converted into machine-readable code **prior to execution**, compared with interpreted languages, which are compiled to machine code **at the time of execution**.

Python 2 vs Python 3

What are the differences?

Short version: Python 2.x is legacy, Python 3.x is the present and future of the language
Python 3.0 was released in 2008. The final 2.x version 2.7 release came out in mid-2010, with a statement of extended support for this end-of-life release. The 2.x branch will see no new major releases after that. 3.x is under active development and has already seen over five years of stable releases, including version 3.3 in 2012, 3.4 in 2014, 3.5 in 2015, and 3.6 in 2016. This means that all recent standard library improvements, for example, are only available by default in Python 3.x.

Python 2 vs Python 3 continued...

Guido van Rossum (the original creator of the Python language) decided to clean up Python 2.x properly, with less regard for backwards compatibility than is the case for new releases in the 2.x range. The most drastic improvement is the better Unicode support (with all text strings being Unicode by default) as well as saner bytes/Unicode separation.

Besides, several aspects of the core language (such as print and exec being statements, integers using floor division) have been adjusted to be easier for newcomers to learn and to be more consistent with the rest of the language, and old cruft has been removed (for example, all classes are now new-style, "range()" returns a memory efficient iterable, not a list as in 2.x).

The What's New in Python 3.0 document provides a good overview of the major language changes and likely sources of incompatibility with existing Python 2.x code. Nick Coghlan (one of the CPython core developers) has also created a relatively extensive FAQ regarding the transition.

However, the broader Python ecosystem has amassed a significant amount of quality software over the years. The downside of breaking backwards compatibility in 3.x is that some of that software (especially in-house software in companies) still doesn't work on 3.x yet.

Deciding Which Python Version to Use

Deciding which version of Python to use will depend on your specific circumstances. Mac and Linux/Unix operating systems general come pre installed with Python. For ease of use you may wish to use the preinstalled version. The Python community however, recommends that new developers start with Python 3.

Python 2 has better library support than Python 3 so depending on which libraries you need to use for your program may also influence the version of Python you use.

Additional Documentation regarding Python 2-3 differences

http://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html

Checking for Python Installation

Before Installing Python it is advisable to determine if Python has been installed on your PC. From the command prompt simply type “Python”. You should see results similar to below:

```
C:\Users\mikee>python
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:25:58) [MSC v.1500 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Installing Python

Installing Python is a relatively simple activity. In General these are the high level instructions for installing Python from a WINDOWS PC.

1. Download the Python installer
2. Run Python Installer
3. Add the Python Home directory to your Path
4. Test Python installation by running the “Python” command from the command line.

Python IDEs

An IDE is an Integrated Development Environment and make writing programs more effective by using certain language specific features such as syntax highlighting and completion.

Python comes with a built in IDE called IDLE. IDLE is great for short programs and for testing purposes however other IDEs such as PyCharm, ATOM, and NOTEPAD++ are often more efficient for larger programs. Throughout this class we will use PyCharm to write our applications.

https://atom.io/
https://notepad-plus-plus.org/
https://www.jetbrains.com/pycharm/

IDLE

IDLE is Python’s Integrated Development and Learning Environment.

IDLE has the following features:

- coded in 100% pure Python, using the **tkinter** GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and Mac OS X
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

NOTE: Additional Information on IDLE usage can be obtained here: https://docs.python.org/2/tutorial/interpreter.html

PyCharm

- PyCharm is a Python IDE for professional programmers. We will be using PyCharm in this class to complete assignments.
- JET Brains has a video series regarding PyCharm that everyone should review before using PyCharm.
  - https://www.jetbrains.com/pycharm/documentation

What is Github?

Have you heard of Dropbox? Google Drive? The Cloud? Github is *sort of* these things.

From Github itself: https://www.youtube.com/watch?v=w3jLJU7DT5E

In plain english: it is a way to make sure the coding stuff you're working on is backed up and accessible for people who might also be working with you. It has version control and very structured collaborative features. For example, if a ton of people make changes to something inside your folder then you or those you designate as capable of making changes can go through each suggested change and merge them all.

GitHub

- GitHub is an web based open source version control solution which we will use for this course.
- GitHub allows developers to share code with other individuals while keeping track of the changes to a specific code base.
- There are multiple ways that you can interact with GitHub. These include cloning from the GitHub URL, using GIT (i.e. commandline), using a tool such as Sourcetree (GUI). Each of these tools offer the capabilitiy to interact with GIT repositories.
- Essentially GIT is the foundation and is the underlying technology which allows you to create code repositories. GITHub is an online tool which allows users to create and interact with GIT repositories. Sourcetree is a GUI which allows you to interact with remote or local GIT repositories.
- This video is a fantastic intro into Git/GitHub
  - https://www.youtube.com/watch?v=SWYqp7iY_Tc

Getting started with Github

First things first, let's take a bit of a tour of everything related to Github:

https://guides.github.com/activities/hello-world/

This tutorial is also interesting. It will help to introduce you to something else we'll be talking about *introducing you to other ways to do python.*

https://try.github.io/ is good to see *how* things work.

This tutorial will usually have the most introductory aspects to Github. It is a useful refresher if it's been a while. It's also a useful space to learn those things that mark every aspect of learning how to *be* in a technical space:

vocabulary

concepts

Learning how to speak like someone who does technical stuff is just as important (for your career) as learning how things work.

PyCharm and GitHub

Now that we've set up PyCharm and GitHub we can integrate PyCharm and Github.

This tutorial will help you get started with integration: https://www.youtube.com/watch?v=2856pbucPAE

Now, there's something to think about with this. You don't *have* to understand this product from beginning to end. All you need to do are two specific things right now:

- Get files into the repository you want.
- Download files from that repository.
- And you can do that using something like
  - Github Desktop: https://desktop.github.com/
  - SourceTree: https://www.sourcetreeapp.com/
- These two products work a lot like Dropbox or Google Drive. They are ways for you to grab (pull) or send (push) to a folder that exists on a computer hard drive that is replicated and redundant in many different ways. So it's replicated, redundant, and this means your data won't disappear. However, this also doesn't mean you can't make mistakes and do something to your data. You can always overwrite, not save, or screw up your data. And you will. Don't worry, you'll make a ton of mistakes!