COMPUTERS, PROGRAMMING LANGUAGES AND HOW WE GOT HERE
Hi, I’m Shaun

- Adjunct Clinical Professor
- Graduated from ITP in spring 2016
- Taught JavaScript, HTML/CSS, Processing, Arduino last semester at FIT and CUNY
  - And television and audio production in a past life
- Work focus is on immersive/interactive storytelling
How about you?

- Do you *need* to take this course?
- Do you *want* to take this course?
- How can computer programming improve your career going forward?
How about you?

- We have students from:
  - Liberal studies
  - Dance, film and television, studio art
  - Music technology and Recorded music
  - Theatre
  - Business
  - Food and Nutrition
  - Business
  - Politics
  - …Undecided
  - And more!
Syllabus Overview
What is a Computer?

- A machine that processes information based on a program
- Virtually all electronic devices are computers
Computers are Everywhere!
What makes a computer?
Hardware

“The Guts”
The Processor

- Every computer has at least one Processor
- The processor acts as the computer’s “brain” and coordinates all activity
- As far as a brain goes, the processor isn’t very bright – it can only perform four distinct actions:
  - Receive a new instruction (Fetch)
  - Make sense of this instruction (Decode)
  - Perform the action defined by this instruction (Execute)
  - Store the result of the action (Store)
- These processors, sometimes called a CPU or “Central Processing Unit,” are made up of miniaturized transistors and circuits on a semiconductor.
The ENIAC (1945)
= 100,000 x
= 1,000 x
Moore’s Law

- Advances in chip design have facilitated these technologies thanks to Moore’s Law.
- The empirical observation that at our rate of technological development, the complexity of an integrated circuit, with respect to minimum component cost will double in about 24 months.
- Expected to continue until around 2020.
Moore's Law

Microprocessor Transistor Counts 1971-2011 & Moore's Law

- The graph shows transistor count doubling every two years.
- The x-axis represents the date of introduction, ranging from 1971 to 2011.
- The y-axis represents the transistor count, ranging from 2,300 to 2,600,000,000.
- The graph includes various microprocessors and their corresponding introduction dates.
Memory

- Computers, like us, need short term memory in order to function properly
- We call this RAM or “Random Access Memory”
- It’s very fast!
- We call it “Random” since the computer can selectively read and write to it's RAM at will – (think of accessing any song on a CD versus accessing that same song on a cassette tape)
- When a computer turns off, it loses its short term memory (we call this volatility)
- In general, the more RAM a system has, the better it performs.
Storage

- Temporary storage is accomplished in memory (RAM)
- We use RAM for short term memory because it is very fast, but it is also volatile and loses information if it is denied electricity
- Long term storage is, by definition, non-volatile – it has the ability to persist even if the power is off
Input Devices

- Input devices allow us to communicate “real world” information into a format that the computer can understand.
Output Devices

- Output devices translate the internal workings of your computer into “real world” stimulus
Software

“The Ideas”
What is Software?

- Software is, in essence, a set of instructions that tell our computers how to behave.
- Software is fluid, and can easily be changed up updated.
- Hardware is rigid, and cannot perform that were not originally planned for during the device’s design.
The Operating System

- Specialized software that coordinate all activities among hardware
- Contains instructions for running application software
- Also know as a “platform” or “software platform”
- Programs that run on different operating systems are know as “cross platform” applications.
- The OS is the internal “Traffic Cop” of your computer
The User Interface

- Portion of System Software that allows you to interact with data

- Two types
  - Graphical (GUI)
  - Command Line

- GUI is more user-friendly, but command line is faster.
Application Software

- Software that serves to help “users more productive and/or assist them with personal tasks” (Shelly, Cashman & Vermaat)
- Application Software is described as a set of programs that are designed to perform specific tasks for the user.
- Categories (very loose grouping)
  - Productivity
  - Graphic Design / Multimedia
  - Home / Personal / Educational
  - Communications
  - … + many others
Data Storage
It’s all Zeros and Ones …

- Everything that communicates with a computer “speaks” the same language (binary)
- Only two “letters” in this language – “0” and “1” which really correspond to electrical impulses (+5v / -5v)
Binary

- Only 2 "letters" in the entire language (0 and 1)
- A single 0 or 1 is referred to as a "bit"
  - bit: 1
- Groupings of 8 bits are referred to as a "byte"
  - byte: 01001011
- 1 byte has the possibility of having 256 unique "states"
Counting in Binary (0 to 255)

<table>
<thead>
<tr>
<th>Decimal</th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Now you can understand this shirt

There are only 10 types of people in the world: Those who understand binary and those who don't.
Encoding Characters

ASCII Code: Character to Binary

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Code</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0011 0000</td>
<td>0100 1111</td>
</tr>
<tr>
<td>1</td>
<td>0011 0001</td>
<td>0101 0000</td>
</tr>
<tr>
<td>2</td>
<td>0011 0010</td>
<td>0101 0001</td>
</tr>
<tr>
<td>3</td>
<td>0011 0011</td>
<td>0101 0010</td>
</tr>
<tr>
<td>4</td>
<td>0011 0100</td>
<td>0101 0011</td>
</tr>
<tr>
<td>5</td>
<td>0011 0101</td>
<td>0101 0100</td>
</tr>
<tr>
<td>6</td>
<td>0011 0110</td>
<td>0101 0101</td>
</tr>
<tr>
<td>7</td>
<td>0011 0111</td>
<td>0101 0110</td>
</tr>
<tr>
<td>8</td>
<td>0011 1000</td>
<td>0101 0111</td>
</tr>
<tr>
<td>9</td>
<td>0011 1001</td>
<td>0110 0000</td>
</tr>
<tr>
<td>A</td>
<td>0100 0000</td>
<td>0110 0001</td>
</tr>
<tr>
<td>B</td>
<td>0100 0001</td>
<td>0110 0010</td>
</tr>
<tr>
<td>C</td>
<td>0100 0010</td>
<td>0110 0011</td>
</tr>
<tr>
<td>D</td>
<td>0100 0100</td>
<td>0110 0100</td>
</tr>
<tr>
<td>E</td>
<td>0100 0101</td>
<td>0110 0101</td>
</tr>
<tr>
<td>F</td>
<td>0100 0110</td>
<td>0110 0110</td>
</tr>
<tr>
<td>G</td>
<td>0100 0111</td>
<td>0110 0111</td>
</tr>
<tr>
<td>H</td>
<td>0100 1000</td>
<td>0111 0000</td>
</tr>
<tr>
<td>I</td>
<td>0100 1001</td>
<td>0111 0001</td>
</tr>
<tr>
<td>J</td>
<td>0100 1010</td>
<td>0111 0010</td>
</tr>
<tr>
<td>K</td>
<td>0100 1011</td>
<td>0111 0011</td>
</tr>
<tr>
<td>L</td>
<td>0100 1100</td>
<td>0111 0100</td>
</tr>
<tr>
<td>M</td>
<td>0100 1101</td>
<td>0111 0101</td>
</tr>
<tr>
<td>N</td>
<td>0100 1110</td>
<td>0111 0110</td>
</tr>
<tr>
<td>space</td>
<td>0010 0000</td>
<td>0110 0000</td>
</tr>
</tbody>
</table>
Encoding Images

Red

\[
\begin{align*}
\text{Red} & \{ 11111111 \\
           & 00000000 \\
           & 00000000 
\end{align*}
\]

Green

\[
\begin{align*}
\text{Green} & \{ 00000000 \\
             & 11111111 \\
             & 00000000 
\end{align*}
\]

Blue

\[
\begin{align*}
\text{Blue} & \{ 00000000 \\
           & 00000000 \\
           & 11111111 
\end{align*}
\]

Yellow

\[
\begin{align*}
\text{Yellow} & \{ 11111111 \\
           & 11001100 \\
           & 00000000 
\end{align*}
\]
Encoding Audio

![Diagram](Image)

The top diagram represents a continuous waveform, with amplitude (in volts) on the y-axis and time (in milliseconds) on the x-axis. The bottom diagram shows a quantized waveform, with amplitude (in volts) and time (in milliseconds) as the axes.
How a Program Works
Computers aren’t smart!

(they’re just really, really, really, really, really, really, really fast!)
Most computers can only do a handful of things

- Read information from memory
- Add, subtract, multiply, divide numbers
- Move data to memory or to permanent storage
- Compare values
Machine Language

10110000
ADD

000000001
1

000000010
2
So what can a computer do?

- Processors can only perform a few very simple operations.
- Each processor has a fixed number of capabilities, called its “instruction set”.
- Each manufacturer maintains its own instruction set.

134. SUB - Subtract
135. TEST - Test For Bit Pattern
136. VERR - Verify Read (286+ protected)
137. VERW - Verify Write (286+ protected)
138. WAIT/FWAIT - Event Wait
139. WBINVD - Write-Back and Invalidate Cache (486+)
140. XCHG - Exchange
141. XLAT/XLATB - Translate
142. XOR - Exclusive OR
For a program to be meaningful we need lots of instructions!

- Usually on the order of millions or billions
- Programs are generally stored on external devices, but they must be copied into memory as needed
- Once in memory the CPU can begin to work its magic
  - Fetch
  - Decode
  - Execute
  - Store
High Level Languages

- Program was solved in the 1950’s when Grace Hopper, a captain in the US Navy, invented COBOL.
- The big idea: Take English words and translate them into machine language in a way that was “device independent”.
- Allowed programmers to concentrate on the tasks the needed doing, not on the mechanics of how a machine worked.
High Level Languages

COBOL

DISPLAY "Hello, World!"

Python

print ('Hello, World!')
Display "Hello, World!"

COBOL Compiler

Machine Language (for Intel Processor) 10101010101110101

COBOL Compiler

Machine Language (for Motorola Processor) 0001010101010110101
Many, many high level languages

http://en.wikipedia.org/wiki/List_of_programming_languages

- COBOL
- Java
- Visual Basic
- PHP
- C
- C++
- Python
- JavaScript
- ActionScript
High-Level Programming Language Structure
## Key Words

- Defined list of words that make up the language
- Sometimes called “Reserved Words”

<table>
<thead>
<tr>
<th>and</th>
<th>del</th>
<th>from</th>
<th>not</th>
<th>while</th>
</tr>
</thead>
<tbody>
<tr>
<td>as</td>
<td>elif</td>
<td>global</td>
<td>or</td>
<td>with</td>
</tr>
<tr>
<td>assert</td>
<td>else</td>
<td>if</td>
<td>pass</td>
<td>yield</td>
</tr>
<tr>
<td>break</td>
<td>except</td>
<td>import</td>
<td>print</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>exec</td>
<td>in</td>
<td>raise</td>
<td></td>
</tr>
<tr>
<td>continue</td>
<td>finally</td>
<td>is</td>
<td>return</td>
<td></td>
</tr>
<tr>
<td>def</td>
<td>for</td>
<td>lambda</td>
<td>try</td>
<td></td>
</tr>
</tbody>
</table>
Operators

- Special symbols that perform certain actions on pieces of data
  - \( \text{answer} = 5 + 2 \)
  - \( \text{name} = 'Harry' + 'Potter' \)
  - \( \text{average} = 250 / 300 \)
Syntax

- Set of rules that must be followed when writing a program
- It’s not smart enough to make up for mistakes that we as humans can account for

```python
if name == 'craig':
    print('Hi there!')
else:
    print('Who are you?')
```
Instructions that you write, consisting of keywords, operators, punctuation, etc

Like a like in a recipe or instruction manual

average = average * 2
High Level Languages: Code

- All statements you write while programming is referred to as “code” or “source code”

(we don’t say “source codes”)
Python

- This semester we will be working with Python
- High level interpreted language
- Used extensively as both a production language as well as a teaching language
- Two modes
  - Interactive
  - Script
- IDLE
  - Integrated Development Environment
“Hello, World!”
Homework

☐ Obtain required materials for the class
☐ Begin "Self-Paced Learning Module #1" on the class website
☐ Do the quiz on
☐ Bring a laptop with you to class (if you have one)