Introduction to Computer Programming

Computers, Programming and how we got here

CSCI-UA.0002-010
Hi!, I am Michell aka Mi Cha El La

- I’m an architect… the kind that designs buildings/spaces, places, and experiences...

- I also studied interactive design at Tisch School of the Arts in the Interactive Telecommunications Program, 4th Floor

- I’m an adjunct and teach here at NYU, at City University of New York and sometimes all over the city...

- I teach electronics, programming, web design and architecture

- I also work as a designer at an exhibit design firm where we plan and attempt to mastermind the experience of a place… we tell stories through design...

- … i love teaching, biking, doughnuts, fabrication, computation as it relates to design, sustainable approaches to design… it got serious no?
ok ... let’s talk business...

everything you need to know about this class will be on the class website...

I will likely post class notes and or code in the evenings after class or morning after

we have a class website:

  cs.nyu.edu/courses/spring17/CSCI-UA.0002-010/

and a common website for all sections in this course:

  http://cs.nyu.edu/courses/spring17/CSCI-UA.0002-003/common_syllabus/
ok ... let’s talk business...

class modules:

http://cs.nyu.edu/elearning/CSCI_UA_0002/index.php
ahem... what’s a computer?

- a machine that processes information based on a program
- virtually all electronic devices are computers
computers are everywhere... and they will take over the world...?
computers are everywhere... and they will take over the world...?
computers are everywhere... and they will take over the world...?
a computer is made of...

**hardware - the guts**

**software - the ideas**

```java
particle [] p = new particle [10];
void setup(){
  for(int i = 0; i < p.length; ++i){
    p[i] = new particle(random(400),random(400),random(10));
  }
  size (400,400);
  smooth(0);
}
void draw(){
  background (255);
  for(int i = 0; i < p.length; ++i){
    p[i].grow();
    p[i].draw();
  }
  line(mouseX,0,mouseX,height);
}
```
A computer is made of...

- **hardware** - the guts

Every computer has at least one processor. The processor is like the brain. It coordinates all activities. The most important component in a computer is the processor. Without it, the computer couldn’t possibly run software.

But it can only do 4 distinct actions:
- receive a new instruction (we refer to this as **fetching**)
- make sense of this instruction (we refer to this as **decoding**)
- perform the action defined by the instructions it was given (this is known as **executing**)
- saving the result of the action (referred to as **storing**)

A computer is made of...

- **hardware** - the guts

The processors are sometimes called a **Central Processing Unit (CPU)** and these are made of miniaturized transistors and circuits on a semiconductor. Today’s CPUs are small chips known as **microprocessors**...

**Transistors** can amplify a current or act as a switch... there are 2 types... but I digress...

**Semiconductors** are materials that have an electrical conductivity value that is between that of copper and that of glass... glass is an insulator while copper is a type of conductor... again I digress
we started with this...

- the eniac

the first electronic general purpose computer that was capable of being reprogrammed to solve numerical problems.
today our tablets... our phones...

have more computing power than the eniac... like 100,000 times more

and these are smaller and smaller and... well...

all this is thanks to Moore’s Law. this isn’t a natural law or physical law its an observation made by Gordon E Moore that states that the rate of technological advancements in the production of transistors over a dense integrated circuit (IC) has continued to double about every two years... and so we get smaller and smaller devices
memory – what was that?

computers are like us… they need short term memory in order to function properly

this short term memory is referred to as **RAM** aka Random Access Memory aka Main Memory aka computers work area

**RAM** is very fast but can only store it while the computer is on… as soon as you turn it off… it looses its short term memory… we refer to this as **volatility**

**RAM** is random cuz… the computer decides where it wants to selectively read and write to its ram

more **RAM** means better performance…
memory - what was that?

then there is long term storage, which is non-volatile, and will store information even after we have turned off our devices...

disk drives, solid state drives, floppy disc, universal serial ports (usbs) aka flash memory

DVDs and CDs are also used for storing... check out this 3d printed record which kinda works like the encoded pits on the surface of a disc
input devices... 

input devices allow us to communicate real world information into a format that the computer can understand.
output devices translate the internal workings of your computer into real world stimulus.
a computer is made of...

- software - the idea

in essence software is a set of instructions that tell our computer how to behave, what to do...

software is fluid and can easily be updated

hardware however is rigid and can not perform in any other way than what it was originally designed for

there are two categories of software: system software and application software
a computer is made of...

- software - the idea

**system software** generally controls and manages the basic operations of a computer.

**OS** aka **Operating System** controls the internal operations of the computer’s hardware, manages all the devices that are plugged in or connected, allows us to save/retrieve and or run other programs.

**Utility Programs** perform specialized task and enhances operations or safeguards data...

**Software Development Tools** thats what we use to develop, create, modify and test software
the user interface

is the portion of the system software that allows you to interact with data... we typically use a **graphical user interface (GUI)** but we can also interact with the computer via a **command line interface** which is faster...
and then there is application software

are a set of programs designed to perform a specific task for the user...

examples you might have used thus far: Microsoft Word, Adobe Photoshop, Photo Booth
its all about zeros and ones...
its all about zeros and ones...

a computer's memory is divided into tiny storage locations known as **bytes**. one byte can store up to a letter but only a letter... or a small number.

to do something more all encompassing or cool the computer needs to work with and have lots of bytes.

these bytes are divided into eight smaller storage locations known as **bits**.

**bit** stands for binary digit and these can be either off or on (0 / 1)... think of these as switches... bits can be negative or positive... think of 0 as -5v and 1 as +5v. its all about those electrons...

computers speak in **binary**. they use the **binary numeric system** to yap around...
its all about zeros and ones...
its all about zeros and ones…

1 byte has the possibility of having 256 unique “states”

<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>0</td>
<td>1</td>
</tr>
<tr>
<td>3</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>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>
**Encoding Characters**

**American Standard Code for Information Interchange** aka **ASCII**

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hex</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000000</td>
<td>0</td>
<td>0</td>
<td>NUL</td>
</tr>
<tr>
<td>1</td>
<td>00000001</td>
<td>01</td>
<td>1</td>
<td>SOH</td>
</tr>
<tr>
<td>2</td>
<td>00000010</td>
<td>02</td>
<td>2</td>
<td>STX</td>
</tr>
<tr>
<td>3</td>
<td>00000011</td>
<td>03</td>
<td>3</td>
<td>ETX</td>
</tr>
<tr>
<td>4</td>
<td>00000100</td>
<td>04</td>
<td>4</td>
<td>ETB</td>
</tr>
<tr>
<td>5</td>
<td>00000101</td>
<td>05</td>
<td>5</td>
<td>ENQ</td>
</tr>
<tr>
<td>6</td>
<td>00000110</td>
<td>06</td>
<td>6</td>
<td>ACK</td>
</tr>
<tr>
<td>7</td>
<td>00000111</td>
<td>07</td>
<td>7</td>
<td>BEL</td>
</tr>
<tr>
<td>8</td>
<td>00001000</td>
<td>10</td>
<td>8</td>
<td>BS</td>
</tr>
<tr>
<td>9</td>
<td>00001001</td>
<td>11</td>
<td>9</td>
<td>HT</td>
</tr>
<tr>
<td>10</td>
<td>00001010</td>
<td>12</td>
<td>A</td>
<td>LF</td>
</tr>
<tr>
<td>11</td>
<td>00001011</td>
<td>13</td>
<td>B</td>
<td>VT</td>
</tr>
<tr>
<td>12</td>
<td>00001100</td>
<td>14</td>
<td>C</td>
<td>FF</td>
</tr>
<tr>
<td>13</td>
<td>00001101</td>
<td>15</td>
<td>D</td>
<td>CR</td>
</tr>
<tr>
<td>14</td>
<td>00001110</td>
<td>16</td>
<td>E</td>
<td>SO</td>
</tr>
<tr>
<td>15</td>
<td>00001111</td>
<td>17</td>
<td>F</td>
<td>SI</td>
</tr>
<tr>
<td>16</td>
<td>00010000</td>
<td>20</td>
<td>10</td>
<td>DLE</td>
</tr>
<tr>
<td>17</td>
<td>00010001</td>
<td>21</td>
<td>11</td>
<td>DC1</td>
</tr>
<tr>
<td>18</td>
<td>00010010</td>
<td>22</td>
<td>12</td>
<td>DC2</td>
</tr>
<tr>
<td>19</td>
<td>00010011</td>
<td>23</td>
<td>13</td>
<td>DC3</td>
</tr>
<tr>
<td>20</td>
<td>00010100</td>
<td>24</td>
<td>14</td>
<td>DC4</td>
</tr>
<tr>
<td>21</td>
<td>00010101</td>
<td>25</td>
<td>15</td>
<td>NAK</td>
</tr>
<tr>
<td>22</td>
<td>00010110</td>
<td>26</td>
<td>16</td>
<td>SYN</td>
</tr>
<tr>
<td>23</td>
<td>00010111</td>
<td>27</td>
<td>17</td>
<td>ETB</td>
</tr>
<tr>
<td>24</td>
<td>00011000</td>
<td>28</td>
<td>18</td>
<td>CAN</td>
</tr>
<tr>
<td>25</td>
<td>00011001</td>
<td>29</td>
<td>19</td>
<td>EM</td>
</tr>
<tr>
<td>26</td>
<td>00011010</td>
<td>30</td>
<td>20</td>
<td>SUB</td>
</tr>
<tr>
<td>27</td>
<td>00011011</td>
<td>31</td>
<td>21</td>
<td>ESC</td>
</tr>
<tr>
<td>28</td>
<td>00011100</td>
<td>32</td>
<td>22</td>
<td>FS</td>
</tr>
<tr>
<td>29</td>
<td>00011101</td>
<td>33</td>
<td>23</td>
<td>GS</td>
</tr>
<tr>
<td>30</td>
<td>00011110</td>
<td>34</td>
<td>24</td>
<td>RS</td>
</tr>
<tr>
<td>31</td>
<td>00011111</td>
<td>35</td>
<td>25</td>
<td>US</td>
</tr>
</tbody>
</table>

This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0
**encoding characters**

*American Standard Code for Information Interchange* aka *ASCII* represents 128 numeric codes that represent the English letters, various punctuation marks and other characters.

*Unicode* is another character set developed after ASCII that tries to be more extensive than ASCII. It is compatible with ASCII and represents many characters for many of the world's languages.
encoding images

Red
\{11111111
  00000000
  00000000\}

Green
\{00000000
  11111111
  00000000\}

Blue
\{00000000
  00000000
  11111111\}

Yellow
\{11111111
  11001100
  00000000\}
small pieces known as samples are what make up a song... each sample is converted into a binary number. the more samples you have the closer it sounds to the original source of sound
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

and can only read instructions in **machine language** and this language has an underlying binary structure...

each operations that a computer performs comes with a machine language instruction that is maintained by each manufacturer. this means that a computer can only execute instructions that are in its CPU instruction set.
let’s go back to talking about computers
let’s go back to talking about computers

each processor has a fixed number of capabilities called instruction set

for a program to be meaningful we need lots of instructions.

were are talking about millions/billions

programs are generally stored on external devices but they must be copied into RAM/memory as needed

Once in RAM we can fetch, decode, execute, and store

when we are fetching, decoding and executing this is known as the fetch-decode-execute cycle
if you don’t like 0s and 1s

don’t fret... we have something other than machine language... we have **assembly language** which is a low level language (we won’t use this ‘cuz it requires way too much computer knowledge) and we have something niftier called **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”

this 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!')
high level languages
which high level language?

- COBOL
- Java
- Visual Basic
- PHP
- C
- C++
- Python
- JavaScript
- ActionScript
what is a program?

A **program** is a list of instructions that causes the CPU to perform operations.

Each instruction in a program is a command that tells the CPU to perform specific operations.

Remember those 0s and 1s? And all this ASCII Unicode talk? Well, we will program using Python as the language we will speak in, and in turn, that language is mapped/translated to machine language and executed to do our whim…

A **compiler** is a program that translates a high-level language program into separate machine language programs that later have to be executed, while an **interpreter** is a program that translates and executes.
Programming language structure: keywords

defined list of words that make up the language sometimes called “Reserved Words”
programming language structure: operators

special symbols that perform certain actions on pieces of data

answer = 5 + 2

name = ‘Harry’ + ‘Potter’

average = 250 / 300
programming language structure: syntax

set of rules that must be followed when writing a program

```python
if name == 'craig':
    print('Hi there!')
else:
    print('Who are you?')
```
programming language structure: statement

are set of instructions that you write consisting of keywords, operators, punctuations etc.

\[ \text{average} = \text{average} \times 2 \]
we will write **code** or **source code** (never to be confused with source codes) every time we write statements.

we will use python which is a high level interpreted language that is used extensively as both a production language as well as a teaching language

we have two modes within python that we will mostly work with:

**interactive**

**script**

we will also use **IDLE** which is an **integrated development environment**
our first code:

print('Hello World')
next steps:

download python 3.0 (it comes with IDLE)

begin “Self Paced Learning Module # 1

bring a laptop to class if you have one or want to