Computers, Programming Languages and How We Got Here

CSCI-UA.0002
By the end of this course, you will:

- Understand fundamental concepts behind programming
- Be able to write programs using a language called Python
- Be ready to learn other languages and concepts!
Logistics

- We meet Mondays and Wednesdays, 3:30 – 4:45pm. Here.

- For future classes:
  - Bring your laptop!
  - Or, at the very least, pen/pencil and paper

- Bookmark and look at the class website:

- And also the common syllabus site for CS-UA.2:
Course Work And Grading

- Lectures
- Workshops
- Quizzes
- Homework assignments
- 2 “Midterm” Exams
- 1 Final Exam

Your final grade will be based on the following:

- 35% - Final Exam
- 5% - Quizzes
- 20% - Assignments
- 20% - Midterm #1
- 20% - Midterm #2

The course material is cumulative; if you skip anything or fall behind it will be hard to catch up. Don’t miss classes, do the homework, and if you feel you’re falling behind, come to me or a tutor for help.
Let’s Start From the Beginning
Let’s Start From the Beginning

What is a computer?
What is a Computer?

- A machine that processes information according to a set of instructions (a program)
- A person who computes or calculates.
What is a Computer?

A Computer is a tool.

Tools magnify or focus effort, allowing us to do things we can’t do unaided.

So how is a computer different from all these?
What is a Computer?

A computer is like a multi-tool. It can be programmed to do many different things.

And a computer operates on information.
What Makes a Computer Useful?

Computers are very fast and flexible at manipulating data.

Data is information.

Information is everywhere and involved in everything we do.

- Language
- Culture
  - Pictures
  - Poetry
  - Music
- Engineering
- Law
- Economics
- Agriculture
- etc. etc. etc.

Computers can be applied to every field of human endeavor.
Computers are Everywhere!

Nowadays, virtually all electronic devices are computers!
What is a Program?

... and what is programming?
A computer is “A machine that processes information according to a set of instructions.”

A program is a sequence of instructions that specifies to a computer actions and computations to be performed.
A Program is a Set of Instructions

1. Turn to your neighbor
2. Introduce yourself
3. Write down the number of the month you were born.
4. Multiply the month by 4
5. Add 13
6. Multiply by 25
7. Subtract 200
8. Add the day of the month you were born
9. Multiply by 2
10. Subtract 40
11. Multiply by 50
12. Add the last two digits of your year of birth
13. Subtract 10500 from their result
14. Hand this to your neighbor. They will now be able to tell your birthday.
Programs Are Built From Algorithms

An algorithm is a set of steps to solve a specific problem.

Let’s try writing a program together:

How to get a book from the library.
Getting A Book From The Library

1. Know which book you want (input)
2. Go to [https://library.nyu.edu/](https://library.nyu.edu/)
3. Enter the title of the book
4. If any books are returned, look for your book
5. If there are multiple versions of your book, find the one you want
6. If it is available on campus:
   1. Write down the call number
   2. Go to Bobst, and find out which floor your call number is on.
   3. Find your book in the stacks using the call number.
   4. Go to the checkout desk and check out your book. (output)
There are several different algorithms needed to complete that task.

For example, how do you look at the results to see if your book is there?

1. Look at the first result. Is it your book?

2. If not, look at the next result.

3. Continue until you find your book or you reach the end.

That is a very simple algorithm.
Writing Algorithms

- Design algorithms independently of programs
  - Focus on “best” way to solve a problem
  - Ignore punctuation, syntax, etc. of programming language
- Use Pseudo Code (write it out in your natural language)
- Graph it as a Flow Chart
- Any algorithm can be implemented as a computer program in many different computer languages.
Programming is a Process

1. Determine the requirements: what do you (or your end-users) need your program to do?

2. Design the program: identify the key tasks, break it down into simplest elements, determine the algorithms to use.

3. Test your solution.

4. Fix errors

5. Identify and implement efficiencies

In practice, there are usually many cycles of refining the requirements and modifying the code until the program really does what it is needed to do.
Programming Can Be:

- Engineering (software engineering)
  - Precise and exacting
  - Provable and formally verifiable
- A craft (creative coding)
  - Creative and iterative process
  - Expertise built through experience
- A science (computer science)
  - The study of information processing and computation
  - Related to mathematics and philosophy
But It's Really Just Problem Solving

Taking a step back, we write programs to scratch an itch.

And once you get into it, you will start thinking about how every day problems could be solved by writing computer programs.

This is the best way to learn to program well:

- Find an interesting problem that you care about
- Write a program to solve that problem or part of that problem
What makes a computer?

**Hardware**

**Software**
Hardware

“The Guts”
The ENIAC (1945)
We’ve Come A Long Way

= 100,000 x

= 1,000 x
Vacuum tubes: slow, expensive, fragile

Transistors: much simpler, much smaller, much cheaper, more reliable, no warm up, much faster.

Integrated circuits: miniaturization added to all the existing benefits, enabled unthought-of possibilities
Microprocessor Transistor Counts 1971-2011 & Moore’s Law

Click to go forward, hold to see history

The curve shows transistor count doubling every two years.

Date of introduction

Transistor count
But What Is All This Stuff Doing?

- Von Neumann architecture (1945) a.k.a. Stored Program
The Central Processing Unit

- The Control Unit executes the program, according to this basic cycle:
  - 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)

- The Arithmetic Logic Unit contains circuitry for carrying out basic arithmetic or logical operations on data.
  - Add, Subtract, Multiple, Divide
  - And, Or, Xor, Not
Memory

- Computers, like us, store active information and instructions in short term memory
  
- We call this RAM or “Random Access Memory”
  
- 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)
  
- It’s very fast!
  
- When a computer turns off, it loses its short term memory (we call this volatility)
Long term storage is, by definition, non-volatile – it is used to store data when 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 data produced by the internal workings of your computer into "real world" stimulus you can understand.
Without Input and Output…

… a computer is just a dumb box.
Software

“The Ideas”
What is Software?

Software is the programs that run on the computer hardware.

There are two major types of software on modern computers:

1. Systems Software
2. Applications
The Operating System

- Specialized software that coordinates all activities among hardware
- Contains instructions for running application software
- The OS is the internal “Traffic Cop” of your computer
- Other systems software, such as drivers, interact directly with the OS.
- The OS is sometimes called a “platform” or “software platform”
- Programs that run on different operating systems are known as “cross platform” applications.
Application Software

- Software that serves to help “users be 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.

- Any “App” on your phone is a piece of Application Software

- Some Broad Categories:
  - Productivity
  - Graphic Design / Multimedia
  - Home / Personal / Educational
  - Communications
  - ... + many others
Data

... What Is It?
Data is Information

And information can be represented in different ways.

“A Cat In A Hat”
Data is Information

And information can be represented in different ways.

And with different degrees of precision.

“A Cat In A Hat”

or

An unusually large cat standing upright with white gloves on its front paws, a somewhat dopey grin, a red bow-tie and red-and-white striped hat, and the ability to speak in rhymes.
Breaking It Down

- The smallest useful piece of information is something that can be in one of two states — a binary choice. “On or Off” or “Yes or No”

- Because computers are made of electric circuits, this is a natural unit for them to operate on.

- We call it a “bit” — Binary Digit.
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 / 0v)
Binary is Base - 2

- One bit can represent $2^0$ values (that’s 2)
- Two bits can represent $2^2 = 4$
- Three bits: $2^3 = 8$
- etc.

- How many values can eight bits represent?
- How about 16 bits?
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”, representing 256 unique values.
- If you need to represent more values, string more bits together!
### 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>0</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.
Relax, you (usually) don’t have to do this when programming
**Encoding Characters**

<table>
<thead>
<tr>
<th>ASCII Code: Character to Binary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0011 0000 O</td>
<td>0100 1111 m</td>
</tr>
<tr>
<td>1 0011 0001 P</td>
<td>0101 0000 n</td>
</tr>
<tr>
<td>2 0011 0010 Q</td>
<td>0101 0001 o</td>
</tr>
<tr>
<td>3 0011 0011 R</td>
<td>0101 0010 p</td>
</tr>
<tr>
<td>4 0011 0100 S</td>
<td>0101 0011 q</td>
</tr>
<tr>
<td>5 0011 0101 T</td>
<td>0101 0100 r</td>
</tr>
<tr>
<td>6 0011 0110 U</td>
<td>0101 0101 s</td>
</tr>
<tr>
<td>7 0011 0111 V</td>
<td>0101 0110 t</td>
</tr>
<tr>
<td>8 0011 1000 W</td>
<td>0101 0111 u</td>
</tr>
<tr>
<td>9 0011 1001 X</td>
<td>0101 1000 v</td>
</tr>
<tr>
<td>A 0100 0001 Y</td>
<td>0101 1001 w</td>
</tr>
<tr>
<td>B 0100 0010 Z</td>
<td>0101 1010 x</td>
</tr>
<tr>
<td>C 0100 0111 a</td>
<td>0110 0001 y</td>
</tr>
<tr>
<td>D 0100 0100 b</td>
<td>0110 0010 z</td>
</tr>
<tr>
<td>E 0100 0101 c</td>
<td>0110 0111 .</td>
</tr>
<tr>
<td>F 0100 0110 d</td>
<td>0110 1000 ;</td>
</tr>
<tr>
<td>G 0100 0111 e</td>
<td>0110 1010 :</td>
</tr>
<tr>
<td>H 0100 1000 f</td>
<td>0110 0110 ;</td>
</tr>
<tr>
<td>I 0100 1001 g</td>
<td>0110 0111 ?</td>
</tr>
<tr>
<td>J 0100 1010 h</td>
<td>0110 1000 !</td>
</tr>
<tr>
<td>K 0100 1011 i</td>
<td>0110 1001 '</td>
</tr>
<tr>
<td>L 0100 1100 j</td>
<td>0110 1010 &quot;</td>
</tr>
<tr>
<td>M 0100 1101 k</td>
<td>0110 1011 (</td>
</tr>
<tr>
<td>N 0100 1110 l</td>
<td>0110 1100 )</td>
</tr>
<tr>
<td>space 0010 0000</td>
<td></td>
</tr>
</tbody>
</table>
Encoding Images

Red: \{11111111, 00000000, 00000000\}

Green: \{00000000, 11111111, 00000000\}

Blue: \{00000000, 00000000, 11111111\}

Yellow: \{11111111, 11001100, 00000000\}
Encoding Audio
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”
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
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
Would you want to program like this?
(please say no ... )
High Level Languages

- Programming 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 that 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

Machine Language (for Motorola Processor) 000101010101110101
Many, many high level languages

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

- COBOL
- Java
- Visual Basic
- PHP
- C
- C++
- Python
- JavaScript
- ActionScript
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!”
For next time …

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