BASICS

- Course Mechanics
- Expectations
- Why are we here?
- Getting going
What is computer science?
What is computer science?

- The study of process or computation expressed in the widest sense as **algorithms**.
- Implementation of said algorithms and processes

- Computers are to computer science what instruments are to music
  - Playing music vs. computing
Algorithms

- **Algorithms** are the main focus
  - Algorithms formally describe computational processes
  - Programs embody algorithms

Note: *An algorithm is independent of how a program implements it*
Algorithm Example

1. Remove book from bag
2. Place book on desk
3. Open book to first unread page

Step 4 contains a few complexities. *Until* suggests that there is some repetition and *Read* could represent an algorithm also.
- The algorithm describes the process that we want the computer to perform

- The trick is telling the computer how to perform that process
One of the oldest algorithms

- Euclid’s algorithm for finding the greatest common divisor (GCD) of two integers:
  1. Input: numbers \( m \) and \( n \)
  2. While \( m \neq n \) do step 3
  3. if \( m > n \), replace \( m \) by \( m - n \)
     otherwise replace \( n \) by \( n - m \)
  4. Output: \( m \)

- Try it with 18 and 12, or with 60 and 18.
Algorithms

- Algorithms can specify how we access data
Data structures

- Data structures specify how data is organized
  - What if the phone book was not alphabetical?
So what is a program?

- A collection of algorithms and data structures
- Both operations and data must be “fair” – *buy low then sell high* ??
More Technical

How to sort a list of numbers:

1. Scan the list and find a smallest number X
2. Output X and delete it from the list
3. Repeat steps 1 and 2 until the list is empty

How do we find X?
Algorithm steps must be effectively computable

Effectively? Includes finitary

Is $x$ a prime number?

- Unfair: first, make a list of all prime numbers
- Fair: generate a list of all prime numbers $\leq x$
Up Next

- Binary
- Programming Languages
- Brief Introduction to Python
  - Statements
  - Python Math Library
Homework

- Read Chapter 1
- If you use your own computer, install Python. Chapter 2 has the necessary instructions.
- Read Course Policies, links on the course home page
- Register your iclicker with Blackboard Learn
- Signup for Piazza
- Check out codelab.com and codingbat.com/python
Quick Review

- What is Computer Science?
- What is an Algorithm?

- We know that a program is a collection of implemented algorithms and data structures
  - How do we develop a program?
The Software Development Process

- The process of creating a program is often broken down into stages according to the information that is produced in each phase.
The Software Development Process 1

- **Analyze the Problem**
  Figure out exactly the problem to be solved. Try to understand it as much as possible.
Determine Specifications
Describe exactly what your program will do.

- Don’t worry about *how* the program will work, but *what* it will do.
- Includes describing the inputs, outputs, and how they relate to one another.
- You can think of your assignments as providing a specification
The Software Development Process 3

- **Create a Design**
  - Formulate the overall structure of the program.
  - This is where the *how* of the program gets worked out.
  - You choose or develop your own algorithm that meets the specifications.
The Software Development Process 4

- **Implement the Design**
  - Translate the design into a computer language.
  - In this course we use Python.
Test/ Debug the Program

- Try out your program to see if it worked.
- If there are any errors (bugs), they need to be located and fixed. This process is called debugging.
- Your goal is to find errors, so try everything that might “break” your program!
The Software Development Process 6

• **Maintain the Program**
  • Continue developing the program in response to the needs of your users.
  • In the real world, most programs are never completely finished – they evolve over time.
What is a computer?

- Processor (CPU)
- Internal memory (RAM), fast, volatile
- System memory, persistent, slower
- Various interface components (keyboard, mouse, USB, blue tooth, ...)

Processor
What Computers Understand

- Modern (digital) computers understand two basic states:
  - On
  - Off
- This is represented by high and low voltage on a wire.
• A single On/Off wire represents 1 bit at a time.
• 8 bits are combined together to form a byte.
• Representing On/Off as 1/0 is called Binary.
  • Binary is a base 2 number system
  • Our every-day numbers are base 10
Binary

- In base 10, each *place holder* or *digit position* represents a power of 10.
- In binary, each *place holder* represents a power of 2.
- The number 1305 is broken down as
  \[ 1 \times 10^3 + 3 \times 10^2 + 0 \times 10^1 + 5 \times 10^0 \]
in base 10
Binary

- The base determines how many digits are available.
  - In base 10 there are ten digits (0 … 9)
  - In base 2 there are two digits (0 or 1)

- The binary number 101 is broken down as
  \[ 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]
  \[ 4 + 0 + 1 = 5 \] (when converted to base 10)
Binary

- Despite having a different base, arithmetic is the same.
- Add each column and carry digits.
  - $1 + 1 = 10$ where the 1 is carried.
  - $1 + 1 + 1 = 11$, again with a carry of 1.
- $101 + 101 = 1010$. 
Longer Example

\[
\begin{array}{c}
11 \\
001101 \\
+ 101011 \\
\hline \\
11100 \\
\end{array}
\]
Longer Example

\[
\begin{array}{c}
\phantom{001101} \\
001101 \\
+ 101011 \\
\hline
111000\ 0
\end{array}
\]
Longer Example

```
001101
+ 101011
```

```
111000
```

```
111000
```
Longer Example

\[
\begin{array}{c}
111 \\
001101 \\
+ 101011 \\
\hline
111000
\end{array}
\]
Longer Example

\[
\begin{array}{c}
\text{1111} \\
\text{001101} \\
+ \text{101011} \\
\hline \\
\text{111000}
\end{array}
\]
Longer Example

```
  1111
  001101
+ 101011
```

```
  11000
```
Longer Example

Verification: $13 + 43 = 56$

\[
\begin{array}{c}
1111 \\
001101 \\
+101011 \\
\hline
111000
\end{array}
\]

\[
\begin{align*}
1101 &= 1 + 4 + 8 = 13 \\
101011 &= 1 + 2 + 8 + 32 = 43 \\
111000 &= 8 + 16 + 32 = 56
\end{align*}
\]
Arithmetic Hardware

- All modern computers have various hardware tasks built into them.
- For addition, each column of the computation corresponds to a single wire and the 0/1 states are toggled by basic logic gates.
- For complex tasks this is much too tedious.
Punched card input
Hardware Instructions

- One step up from the hardware itself is a set of hardware instructions.
- These are the basic elements of programming that the hardware is capable of supporting.
- These are typically still very tedious as each instruction directly corresponds to a hardware element.
x86 assembly

section .text
global _start, write
write:
    mov al,1
    syscall
    ret
_start:
    mov rax,0x0a68732f6e69622f
    push rax
    xor rax,rax
    mov rsi,rsp
    mov rdi,1
    mov rdx,8
    call write
exit:
    xor rax,rax
    mov rax,60
    syscall
Programming Languages

- Programming languages are a compromise between spoken language and formal math.
- They allow humans to communicate with computers at a higher level than machine instructions.

Note: Software can usually be created using different programming languages, such as Java, Python, C++, etc.
Languages Cont.

- The easier the language is for humans to use, the harder it is for a computer to interpret.

- Natural language is ambiguous:
  - Fruit flies like bananas
  - I never said she took the money

- There are many programming languages at many different ‘levels’ of complexity and convenience.
Grammar and Syntax

- Languages have a set grammar and syntax
  - Defines a “valid” program
- Punctuation: “, . ! ; : ?” always follows a word in English.
- Words are the basic building block in English
Building Blocks of Python

- Numbers, booleans, strings, floating point numbers, and variables are basic building blocks of Python
  - True for most programming languages
  - Keywords

- Python’s grammar defines how these building blocks fit together
Grammar Check

- When executing a python program IDLE will tell you when your grammar is incorrect
  - Given as error messages
  - One error at a time
Numbers in Python

- Python allows us to work with familiar base 10 numbers.
  - This is an example of the utility of programming languages.

- Numbers in python have various types.
  - 1.0 vs 1

See Chapter 3, Section 1, for more details!

Note: *Python needs to be told what type of math you want to do*
Booleans

- Booleans are truth values
  - True or False (both are keywords in Python)
  - Booleans also have a type: `bool`
Python Grammar

- Code in python consists of one of several things
  - Statement
  - Function Declaration
  - Function Call
  - Loop
  - Conditional
Statements

- These are the most basic elements in python code
- Statements fit on a single line and typically
  - Assign a value to a variable
  - Perform arithmetic
  - Print something
  - Call functions or procedures
Examples

- print("Hello")
- print("Hello" + "World")
- print(abs(-1))
- print(1.0/2.0)
- print(1/2)
Using Python as a Calculator

- In the interactive mode you can type statements for python to evaluate

```python
>>> 4+5
9

>>> 10/2
5

>>> 10%2
0
```
Using the Math Library

- Besides (+, -, *, /, //, **, %, abs), we have lots of other math functions available in a math library.

- A library is a module with some useful definitions/functions.
Using the Math Library

- Let’s write a program to compute the roots of a quadratic equation!

\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

- The only part of this we don’t know how to do is find a square root… but it’s in the math library!
Using the Math Library

- To use a library, we need to make sure this line is in our program:

  ```python
  import math
  ```

- Importing a library makes whatever functions are defined within it available to the program.
Using the Math Library

- To access the sqrt library routine, we need to access it as `math.sqrt(x)`.

- Using this dot notation tells Python to use the sqrt function found in the math library module.

- To calculate the root, you can do `discRoot = math.sqrt(b*b - 4*a*c)`
More Homework

- Read Chapter 3, Section 1
- Practice navigating the ebook
- Register your iClickers
- Install IDLE and Python 3.2
- Play with Python as desk calculator