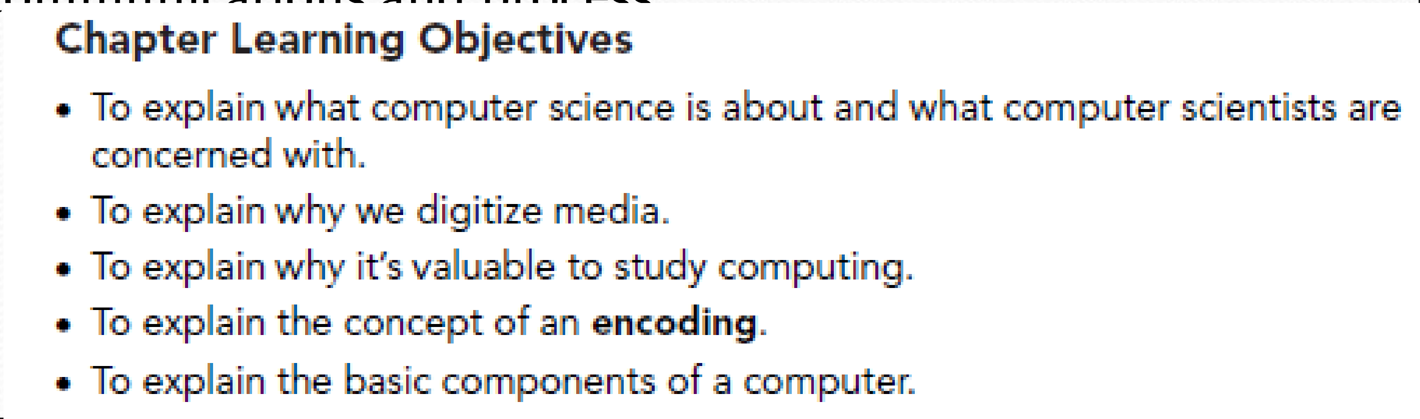
Introduction to Computing and Programming in Python: A Multimedia Approach

Chapter 1: Introduction to Computer Science and Media Computation

Story

- What is computer science about?
- What computers *really* understand, and where Programming Languages fit in
- Media Computation: Why digitize media?
  - How can it possibly work?
- Computer Science for Everyone
  - It's about communications and process



Chapter Learning Objectives

- To explain what computer science is about and what computer scientists are concerned with.
- To explain why we digitize media.
- To explain why it's valuable to study computing.
- To explain the concept of an **encoding**.
- To explain the basic components of a computer.

What's computation good for

- Computer science is the study of recipes
- Computer scientists study…
  - How the recipes are written (algorithms, software engineering)
  - The units used in the recipes (data structures, databases)
  - What can recipes be written for (systems, intelligent systems, theory)
  - How well the recipes work (human-computer interfaces)

Specialized Recipes

- Some people specialize in crepes or barbeque
- Computer scientists can also specialize on special kinds of recipes
  - Recipes that create pictures, sounds, movies, animations (graphics, computer music)
- Still others look at *emergent properties* of computer "recipes"
  - What happens when lots of recipes talk to one another (networking, non-linear systems)

Key concept:
The *COMPUTER* does the recipe!

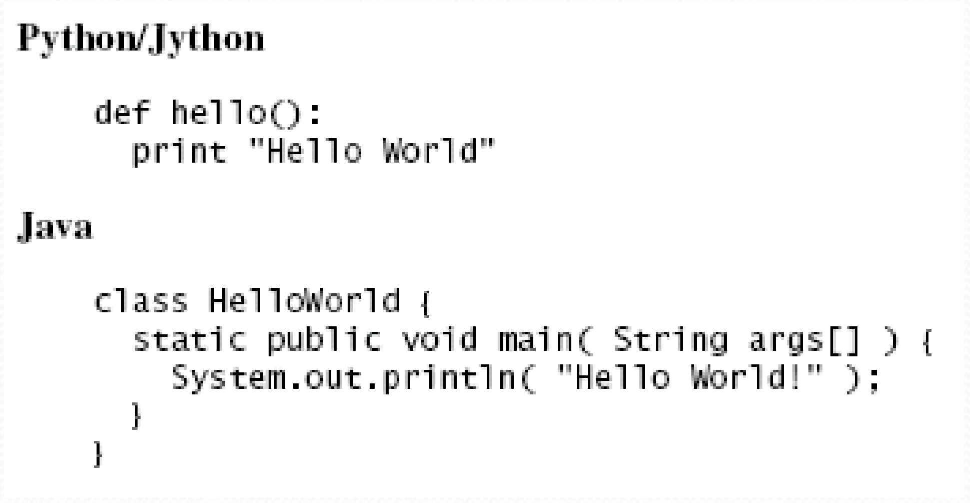
- Make it as hard, tedious, complex as you want!
- Crank through a million genomes? No problem!
- Find one person in a 30,000 campus? Yawn!
- Process a million dots on the screen or a bazillion sound samples?
  - *That*'s media computation

What computers understand

- It’s not really *multimedia* at all.
  - It’s *unimedia* (said Nicholas Negroponte, founder of MIT Media Lab)
  - *Everything* is 0’s and 1’s
- Computers are *exceedingly* stupid
  - The only *data* they understand is 0’s and 1’s
  - They can only do the most simple things with those 0’s and 1’s
    - Move this value here
    - Add, multiply, subtract, divide these values
    - Compare these values, and if one is less than the other, go follow this step rather than that one.
  - Done fast enough, those simple things can be amazing.

Programming Languages

- Different programming languages are different ways (*encodings*) that turn into (same/similar) commands for the computer

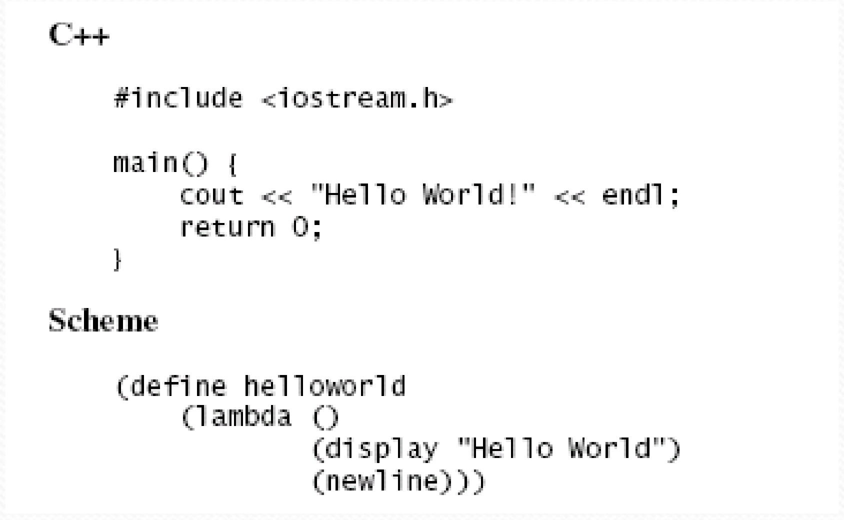


Python/Jython

```
def hello():
  print "Hello World"
```

Java

```
class HelloWorld {
  static public void main( String args[] ) {
    System.out.println( "Hello World!" );
  }
}
```



C++

```
#include <iostream.h>

main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

Scheme

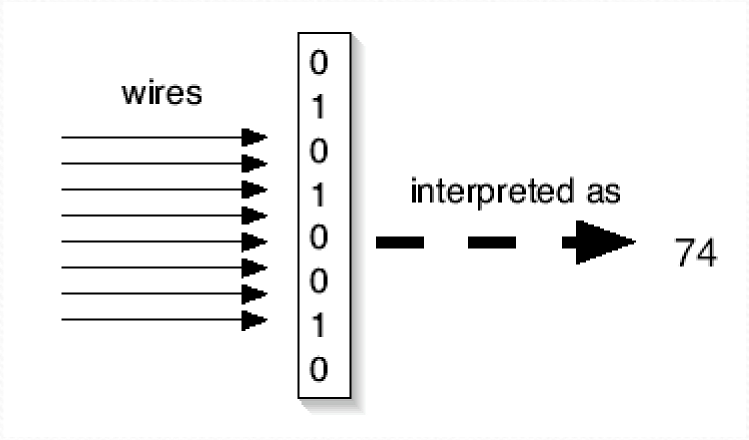
```
(define helloworld
    (lambda ()
            (display "Hello World")
            (newline)))
```

A word about Jython

- Jython ***is*** Python
- Python is a language implemented in C.
- Jython is the *same* language implemented in Java.
  - Is the pizza different if a different company makes the flour? If so, not by much.

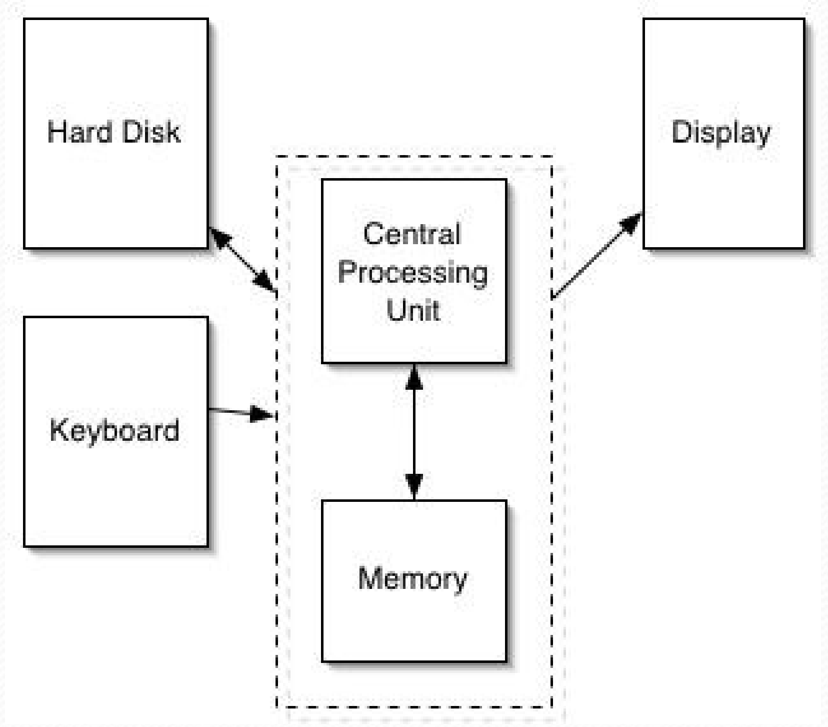
Key Concept: Encodings

- We can *interpret* the 0's and 1's in computer memory any way we want.
  - We can treat them as numbers.
  - We can *encode* information in those numbers
- Even the notion that the computer understands numbers is an interpretation
  - We encode the voltages on wires as 0's and 1's, eight of these defining a *byte*
  - Which we can, in turn, interpret as a decimal number



How a computer works

- The part that does the adding and comparing is the *Central Processing Unit (CPU).*
- The CPU talks to the *memory*
  - Think of it as a sequence millions of mailboxes, each one byte in size, each of which has a numeric *address*
- The *hard disk* provides 10 times or more storage than in memory (20 billion bytes versus 128 million bytes), but is millions of times slower
- The display is the monitor or LCD (or whatever)



Layer the encodings as deep as you want

- One encoding, ASCII, defines an "A" as 65
  - If there's a byte with a 65 in it, and we decide that it's a string, POOF! It's an "A"!
- We can string together lots of these numbers together to make usable text
  - "77, 97, 114, 107" is "Mark"
  - "60, 97, 32, 104, 114, 101, 102, 61" is "<a href=" (HTML)

What do we mean by *layered* encodings?

- A number is just a number is just a number
- If you have to treat it as a letter, there's a piece of software that does it
  - For example, that associates 65 with the graphical representation for "A"
- If you have to treat it as part of an HTML document, there's a piece of software that does it
  - That understands that "<A HREF=" is the beginning of a link
- That part that knows HTML communicates with the part that knows that 65 is an "A"

Multimedia is unimedia

- But that same byte with a 65 in it might be interpreted as…
  - A very small piece of sound (e.g., 1/44100-th of a second)
  - The amount of redness in a single dot in a larger picture
  - The amount of redness in a single dot in a larger picture which is a single frame in a full-length motion picture

Software (recipes) defines and manipulates encodings

- Computer programs manage all these layers
  - How do you decide what a number should mean, and how you should organize your numbers to represent all the data you want?
  - That's data structures
- If that sounds like a lot of data, it is
  - To represent all the dots on your screen probably takes more than 3,145,728 bytes
  - Each second of sound on a CD takes 44,100 bytes

Thank God for Moore’s Law

- Gordon Moore, one of the founders of Intel, made the claim that (essentially) computer power doubles for the same dollar every 18 months.
- This has held true for over 30 years.
- Go ahead! Make your computer do the same thing to everyone of 3 million dots on your screen! It doesn’t care! And it won’t take much time either!

Why digitize media?

- Digitizing media is encoding media into numbers
  - Real media is *analogue* (continuous).
  - To digitize it, we break it into parts where we can’t perceive the parts.
- By converting them, we can more easily manipulate them, store them, transmit them without error, etc.

How can it work to digitize media?

- Why does it work that we can break media into pieces and we don't perceive the breaks?
- We can only do it because human perception is limited.
  - We don't see the dots in the pictures, or the gaps in the sounds.
- We can make this happen because we know about *physics* (science of the physical world) and *psychophysics* (psychology of how we perceive the physical world)

Why should you need to study "recipes"?

- To understand better the recipe-way of thinking
  - It's influencing everything, from computational science to bioinformatics
  - Eventually, it's going to become part of everyone's notion of a liberal education
  - That's the *process* argument
  - BTW, to work with and manage computer scientists
- AND…to communicate!
  - Writers, marketers, producers communicate through computation
- We'll take these in opposite order

Computation for Communication

- All media are going digital
- Digital media are manipulated with software
- You are limited in your communication by what your software allows
  - What if you want to say something that Microsoft or Adobe or Apple doesn't *let* you say?

Programming is a communications skill

- If you want to say something that your tools don't allow, program it yourself
- If you want to understand what your tools can or cannot do, you need to understand what the programs are doing
- If you care about preparing media for the Web, for marketing, for print, for broadcast… then it's worth your while to understand how the media are and can be manipulated.
- Knowledge is Power,
Knowing how media work is powerful and freeing

We’re not going to replace PhotoShop

- Nor ProAudio Tools, ImageMagick and the GIMP, and Java and Visual Basic
- But if you know what these things are doing, you have something that can help you learn new tools

Knowing about programming is knowing about process

- Alan Perlis
  - One of the founders of computer science
  - Argued in 1961 that Computer Science should be part of a liberal education: *Everyone* should learn to program.
    - Perhaps computing is more critical to a liberal education than Calculus
    - Calculus is about rates, and that's important to many.
    - Computer science is about process, and that's important to *everyone*.
  - *Automating* process changes *everything*.

A Recipe is a Statement of Process

- A recipe defines how something is done
  - In a *programming language* that defines how the recipe is written
- When you learn the recipe that implements a Photoshop filter, you learn how Photoshop does what it does.
- And that is powerful.

Finally: Programming is about Communicating Process

- A program is the most concise statement possible to communicate a process
  - That’s why it’s important to scientists and others who want to specify *how* to do something understandably in as few words as possible

Python

- The programming language we will be using is called *Python*
  - http://www.python.org
  - It’s used by companies like Google, Industrial Light & Magic, Pixar, Nextel, and others
- The *kind* of Python we’re using is called Jython
  - It’s Java-based Python
  - http://www.jython.org