Introduction and Course Overview

WTP 2010
Computer Science
Day 1: Monday, June 29

The computer science staff


Oshani Seneviratne, Instructor
oshani@mit.edu, office: 32-G515


Jovana Knezevic, Tutor
knezevic@mit.edu


Emily Pitts, Tutor
efpitts@mit.edu


Hilary Monaco, Tutor
htm@mit.edu

Course logistics

Daily class (required)
- Lecture
- Problem sets, mostly in class
- Reading assignments, before class

Study sessions (optional)
- Late afternoons or evenings
- Complete unfinished in-class exercises
- Review and discuss material

Problem sets

Do them!
- Solutions are due before the next class
- Doing is essential for understanding CS
- Discussing concepts in a group is ok
- Copying your friend's program is not
- You must do the written work yourself

No formal grades, but:
- Per-problem grading - feedback for you and us!
- If you don't understand, ask!
- Your participation is how we get to know you.
- Don't worry about grades! You are here to learn.

Lab Rules

- Your time in the lab should only be used for:
  - Going over the course material
  - Solving the problem sets
  - Going over the assigned readings for the next day
- Things prohibited in the lab:
  - Food and drinks
  - Checking email, Facebooking, online chatting
- If you finish early:
  - Do a challenging problem
  - Help a friend
  - Read material for the next day

Course resources

- Course handouts
- Course web page:
  http://people.csail.mit.edu/oshani/wtp
- Readings from:
  How to Think Like a Computer Scientist
  (Downey)
- Instructor and tutors

Group Activity

- Get into four groups and discuss the following:
  - Things that use computers in your day-to-day activities
  - Things that would be really hard to do without a computer
  - Things that are very easy for you to do, but is very hard for a computer to do

What is computer science?

It is NOT:

- using Microsoft Word
- updating your Facebook page
- installing your little brother's printer
- writing a webpage
- even programming!

What is computer science?

It IS:

- problem solving, using the computer as a tool
- finding general solutions to problems
- finding efficient solutions to problems
- thinking about what kinds of problems are hard or slow for computers to solve, and why

"Computer science is no more about computers than astronomy is about telescopes."
- E.W. Dijkstra (famous computer scientist)

Programming?

If computer science is not about programming, why are we learning to program in this class?

Programming is a necessary tool.

Areas of Computer Science

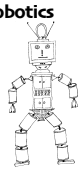
- Artificial Intelligence (AI)
- Systems
- Theory

Areas of Computer Science – AI

- **Machine Learning**

- **Computer Vision**

- **Robotics**



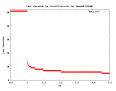
- **Information Retrieval**

- **Cognitive Science**

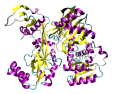
- **Human Computer Interaction**

- **Natural Language Processing**

- **Data Mining**



- **Computational Biology**



Areas of Computer Science – Systems

- Operating Systems

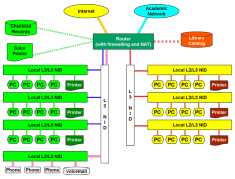
- Databases

- Programming Languages

```
def add3(x):
    return x+3

def dotwrite(ast):
    nodename = getnodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '"%s" [label="%s, %s"];' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= "%s"];' % ast[1]
        else:
            print ';'
    else:
        print ';'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print '    %s -> {' % nodename
        for  in  namedchildren:
            print '%s' % name,
```

- Computer Networks



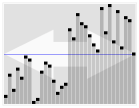
- Computer Security

- Compiler Design

- Ubiquitous Computing

Areas of Computer Science – Theory

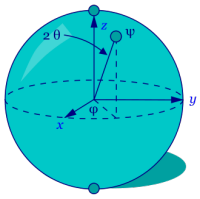
- Algorithms



- Cryptography

%^RCHVGAS&QF

- Quantum Computing



- Complexity Theory

P = NP?

- Computational Geometry

Things we hope you will learn

- What is computer science?
- Computational problem solving skills
  - Write small pieces of code
  - Understand code written by others
  - Understand the capabilities and limitations of computer science
- The basics of programming in the Python language

General CS Vocabulary

- Operating system
- File system
- Computer program
- Programming language

What is an operating system?

What are some examples of operating systems?

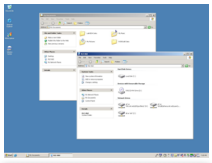
- Windows
- Mac OS
- Unix
- GNU/Linux

What does an operating system do?

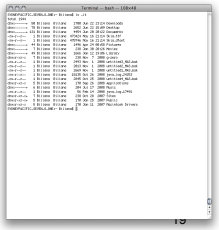
- Controls computer hardware
- Runs other programs
- Handles input/output
- Decides how to allocate resources

Talking to the operating system

- Graphical user interface (GUI)
  - Windows
  - Double-click on icons



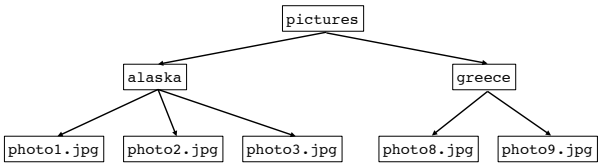
- Text-based user interface
  - Shell
  - Type commands at prompt



What is a file system?

Consists of:
- files that contain data
- directories that contain files

Directories are organized in a tree-structure.



Group Activity

- How would you make a peanut butter and jelly sandwich?

What is a computer program?

A sequence of instructions telling the computer how to perform a computation.

- mathematical: solve a system of equations

$$\begin{aligned} 3x + 2y - z &= 1 \\ 2x - 2y + 4z &= -2 \\ -x + {}^1/_2 y - z &= 0 \end{aligned} \qquad \begin{aligned} x &= 1 \\ y &= -2 \\ z &= -2 \end{aligned}$$

- symbolic: sort a list of names alphabetically

```
iris
lindsay
clare
jessica
amelia
kelly
```

```
amelia
clare
iris
jessica
kelly
lindsay
```


What is a computer program?

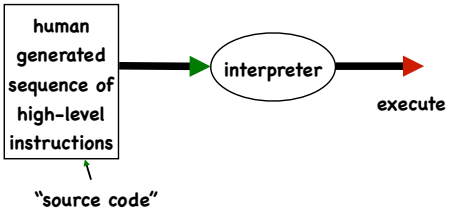
Components of a Computer Program:

- input: get data from keyboard, mouse, file, etc.
- output: display data on screen or write to file.
- math: basic arithmetic operations (+, -, *, /).
- testing: check whether some condition is true to decide which sequence of instructions to perform.
- repetition: do something multiple times, probably with some variation.

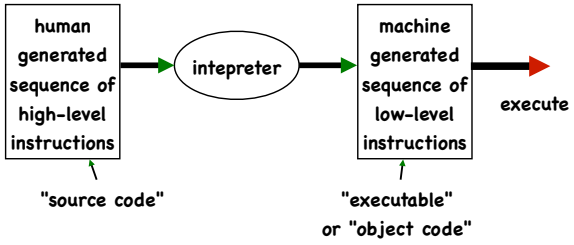
Writing a program

1. Identify the problem.
2. Design a solution strategy.
3. Break up the solution into small enough subtasks that each can be performed by a basic operation.
4. Write down the sequence of basic operations using the syntax of the programming language.

Running a program



Running a program



What is a programming language?

A formal language to express these instructions.

Natural languages:
- syntax less critical
- ambiguous
- redundant
- idioms and metaphors

Formal languages:
- strict syntax
- no ambiguity
- concise
- literal

Characteristics of a programming language

- Syntax - what are legal expressions of a language?
  "cat dog girl"
- Semantics - what is meaningful in a language?
  "My chair is Jenny"

Types of Programming Languages

- Low-level languages, also known as "machine language" or "assembly language"
- High-level languages
  - Interpreted: Python, Scheme, BASIC
  - Compiled: C, C++, Java
  - General Purpose: Python, C, Java
  - Specialized: Matlab

Assembly Language

```
;
; Hello World in nasm/NetBSD(aout)
;

section .text

_start:
    push    dword len    ; Length
    push    dword msg    ; Address
    push    dword 1      ; Stdout
    mov     eax, 0x4     ; write
    call    _syscall
    add     esp, 12      ; Stack pointer

    push    dword 0
    mov     eax, 0x1     ; exit
    call    _syscall

_syscall:
    int     0x80
    ret

msg db      "Hello World",0xa
len equ     $ - msg
```

Java

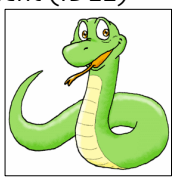
```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World");
   }
}
```


Python

```
print 'Hello World'
```


IDLE

- Integrated DeveLopment Environment (IDLE) for Python
  - Edit your program source code
  - Run your program
  - Debug your program
- Has graphical user interface (GUI)



The next three weeks

Week 1: Programming Fundamentals

Week 2: Advanced Topics

Week 3: Final Project

Weekly Review on every Monday

Expectations

- Work hard
- Ask questions!
- Do your problem sets
- Challenge yourself
- Help one another
- Have fun!

Today's exercises

- Log into Athena, open IDLE
- First program
- Edit the program
- Readings for tomorrow: Chapters 1 & 2, Sections 5.1-3, 5.11, 8.1, 8.4