CPSC 219
Introduction to Computer Science for Multidisciplinary Studies II

Instructor: Mark Hancock

Lecture 01 Summary

- Administrivia
- Expectations
- Purpose of Programming
- Course Goals & Objectives
- Interpreters vs. Compilers
- Syntax Errors vs. Semantic Errors

Administrivia

Office Hours

- MS 616
- TR 11:00-12:00 (or by appointment)
- Email: msh@cs.ucalgary.ca
- Phone: 210-9499

Textbooks

- Head First Java, 2nd Edition (required)

  Kathy Sierra and Bert Bates (*O'Reilly & Associates*)

- C Programming Language (recommended)

  Brian Kernighan and Dennis Ritchie (*Prentice Hall*)

Grading

- Assignments (50 %)
- Midterm (25 %)
- Final (25 %)

Assignments

Assignment #	Weight	Due Date
1	5%	Friday, Feb 6
2	7.5%	Friday, Feb 20
3	15%	Friday, Mar 13
4	15%	Friday, Apr 3
5	7.5%	Friday, Apr 17

Academic Misconduct

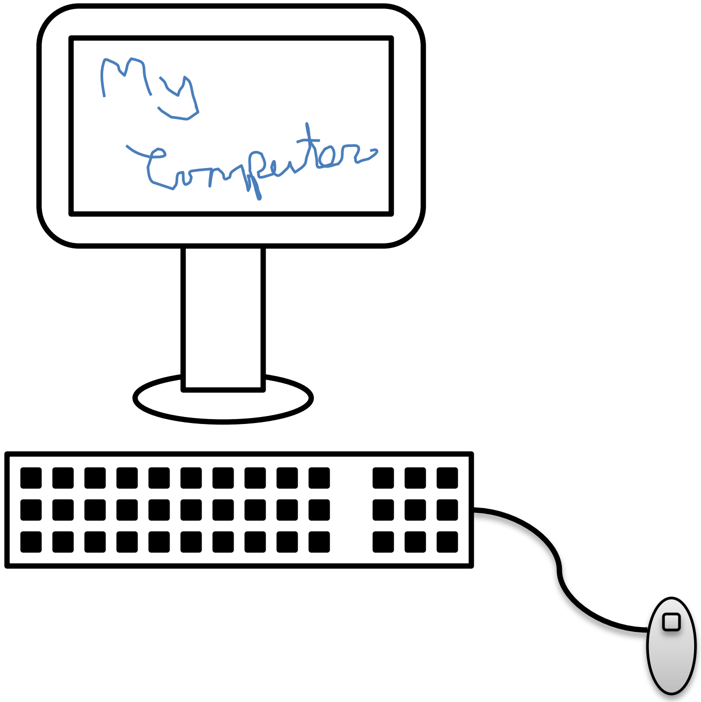
- Working together vs. plagiarism

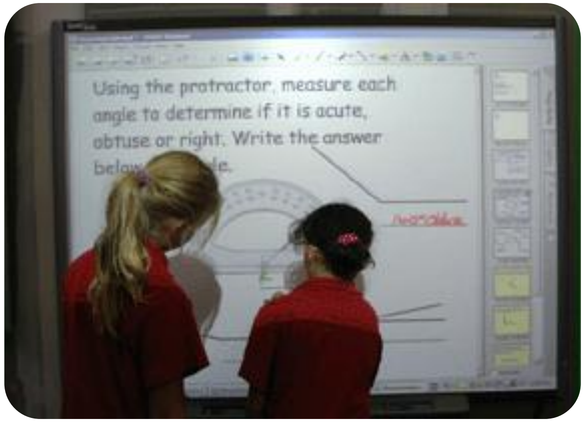
Expectations

Purpose of Programming

By the end of this 30 minute section, you will be able to identify two different uses of a computer program, outside of the field of Computer Science.

Activity: Draw a computer


My
Computer


Using the protractor, measure each
angle to determine if it is acute,
obtuse or right. Write the answer
below

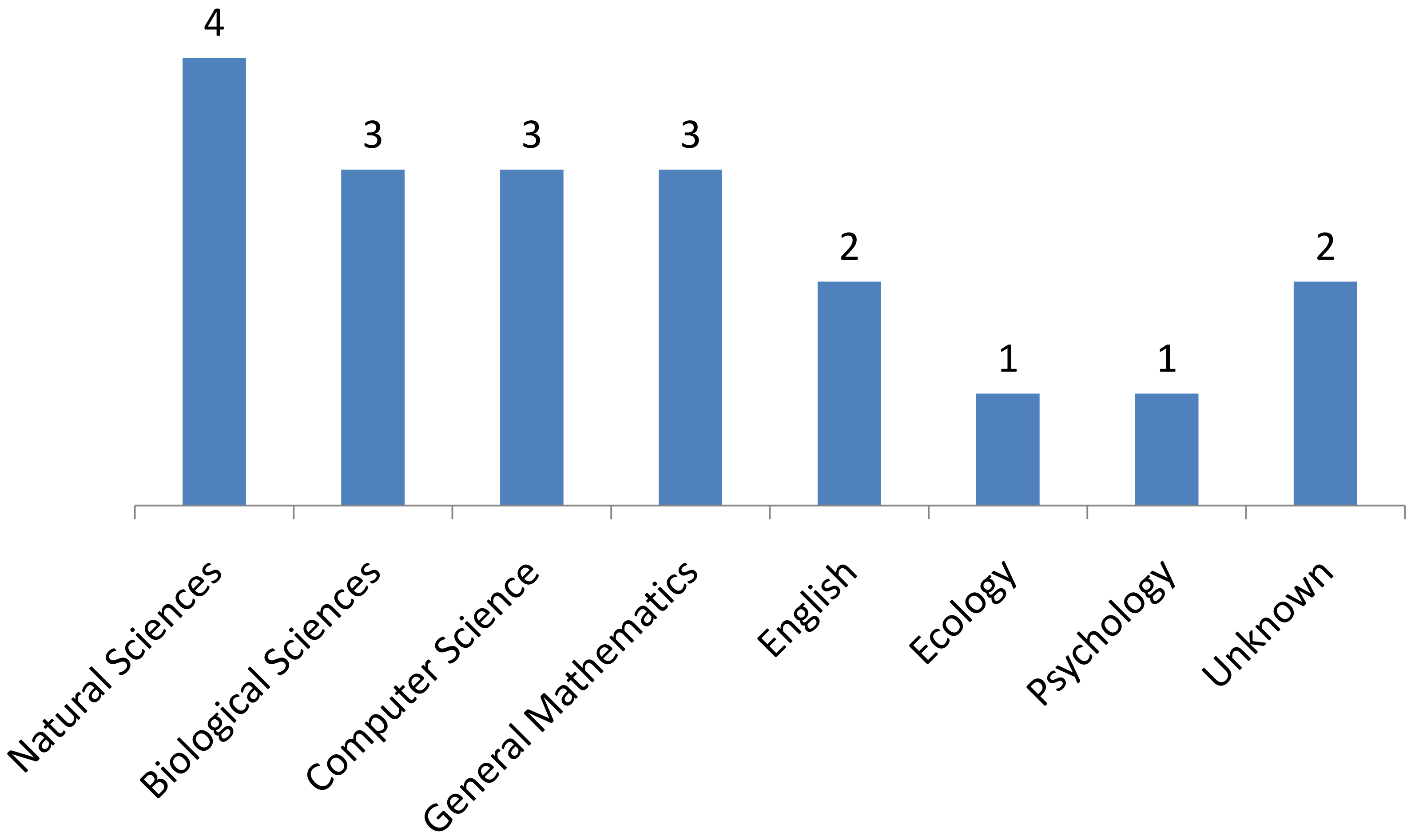
Elements of a Program

- Input
  - Sources: people, internet, weather, cameras, etc.
- Sequence of steps
  - Magic happens
- Output
  - Display the result: on a screen, as sound, etc.

Example Program

Tip Calculator:

- Input
  - Bill and tip amount entered by person
- Sequence of steps
  - Magic happens
- Output
  - Amount of tip displayed on screen


4
3
3
3
2
1
1
2
Natural Sciences
Biological Sciences
Computer Science
General Mathematics
English
Ecology
Psychology
Unknown

In pairs, describe two possible computer programs (one in each person’s discipline).

Example Program

Tip Calculator:

- Input
  - Bill and tip amount entered by person
- Sequence of steps
  - Magic happens
- Output
  - Amount of tip displayed on screen

Course Goals & Objectives

How do we get the computer to perform a sequence of steps on a particular input to produce some sort of useful output?

How do we solve problems with a computer program?

Can we make this sequence of steps be reusable by someone else looking to solve a similar problem?

How do we ensure that our program does what we want it to do?

Change the way we think about this process:
Object-Oriented Programming

Course Goals

- This course aims to help the student develop an awareness of:
    - how *objects* can be used as a basis for solving problems;
    - how to implement solutions using an *object-oriented language*;
    - how to apply *object-oriented* problem-solving techniques to scientific areas of study;
    - the nature of *objects* and their relationship to information and information processing; and
    - how to develop solutions which exhibit elements of good style.

Course Objectives

- By the end of this course students should be able to:
    - analyse problems using an *object-oriented* framework;
    - design and implement solutions using *object-oriented* concepts:
        - – *encapsulation*
        - – *inheritance*
        - – *polymorphism*;
    - create and execute *unit tests* on implemented solutions; and
    - evaluate the *quality* of program designs.

Programming Languages:
C and Java

What “language” does the computer use to execute a sequence of steps?

Machine Code

"Machine code or machine language is a system of instructions and data executed directly by a computer's **central processing unit**."

Source: Wikipedia

Each CPU has its own **instruction set**.

- Arithmetic: add, subtract, multiply, divide
- Move data from place to place
- Control flow: e.g., if, goto, call a function
- Logic: and, or, not, exclusive or (XOR)

Machine Code Example

Instruction: add registers 1 and 2 and place the result in register 6 (MIPS architecture)

0	1	2	6	0	32	Decimal
000000	000001	000010	00110	000000	100000	Binary

Assembly Language

One-to-one mapping from machine code to “human-readable” instruction.

Assembly Language

Motorola 68000 CPU:

- `ADD`: add two operands together and store the result in the destination operand
- `MULU`: multiplies a 16-bit data register by a 16-bit effective address operand leaving the 32-bit result in the data register
- `MOVE`: Copies a byte (8 bits), word (16 bits) or long word (32 bits) from one effective address to another

Assembly Language Example

Evaluate the equation: A2 = A0 * A1 + A3

```
lea $1000, A0
lea $1004, A1
lea $1008, A2
lea $100A, A3
move.l (A0), D0
move.l (A1), D1
mulu D0, D1
move.l (A3), D0
add.l D1, D0
move.l D0, (A2)
```

How would you write a program that evaluates that equation in Python?

Equation: A2 = A0 * A1 + A3

```
A2 = A0 * A1 + A3
```

Summary

- Machine code is a set of binary instructions specific to a CPU
- Assembly language is a one-to-one mapping from machine instructions to “human-readable” instructions
- Reading and writing code in a language like Python is **much** easier

Interpreters vs. Compilers

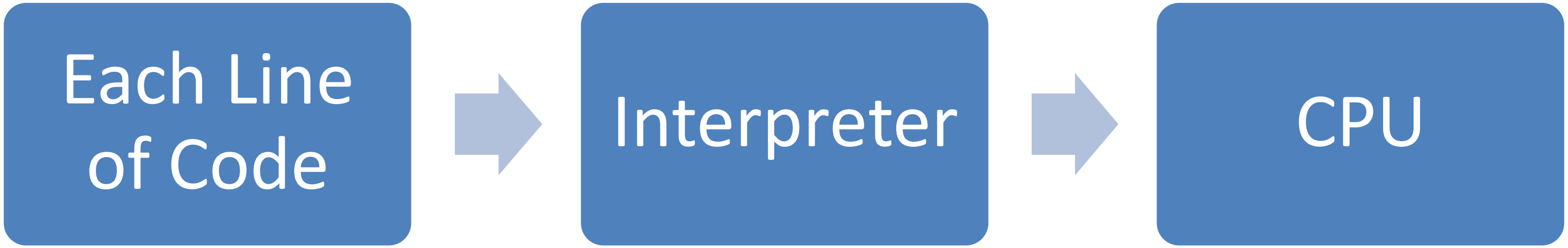
By the end of this 30 minute section, you will be able to describe the steps necessary to run a compiled program.

Can the CPU understand Python (or C/Java)? Why/why not?

Interpreters

Python Source Code

To run:



Each Line of Code

Interpreter

CPU

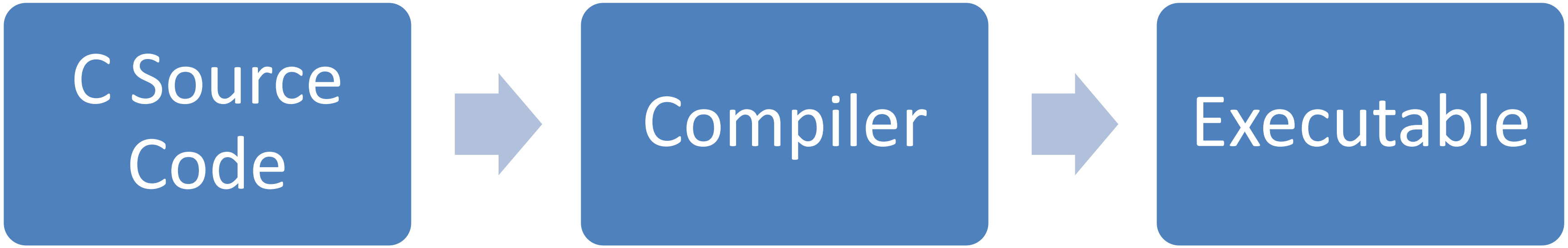
Steps required

- Write the source code in a text file
  - E.g., HelloWorld.py
- Run the program
  - Execute the following command (e.g., in Unix):

```
python HelloWorld.py
```

Compilers

Written by programmer



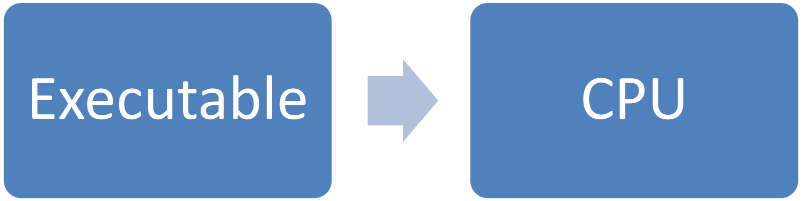
C Source Code

Compiler

Executable

Machine Language
(can be run on CPU)

To run:



Executable

CPU

Steps required

- Write the source code in a text file
  - E.g., HelloWorld.c
- Compile the source code
  - Execute the following command (e.g.):

```
gcc HelloWorld.c
```

- Run the program
  - Execute the following command (e.g., in Unix):

```
./a.out
```

Demo

Java is a compiled language. What steps are necessary to run a program written in Java?

Steps

- Write the source code in a text file
  - E.g., HelloWorld.java
- Compile the source code
  - Execute the following command (e.g.):

```
javac HelloWorld.java
```

- Run the program
  - Execute the following command (e.g.):

```
java HelloWorld
```

Why not learn one language and use it for everything?

In this course

- Pointers:
  - Assembly language
  - **C**
- Abstract Data Types:
  - Python
  - **Java**

Syntax Errors vs. Semantic Errors

By the end of this 15 minute section, you will be able to distinguish between a syntax error and a semantic error.

With a natural language (e.g., English), what is the difference between syntax and semantics?

Syntax error in English:

- “I accidentally the whole class.”

Semantic error in English:

- “I’ve been alive for five light years.”

Syntax Error

- An error caused by incorrect use of the syntax of the programming language
- Result:
  - Compiled language?
  - Interpreted language?

Syntax Error: Example

Semantic Error

- An error caused by code which may be readable by the computer, but has incorrect logic
- Result:
  - Compiled language?
  - Interpreted language?

Semantic Error: Example

Python

```
def addInts(a, b):
  return a + b

x = addInts(10, 20)

print "x = %i\n" % x

y = addInts(200, "Hello")
```

C

```
int addInts(int a, int b)
{
    return a + b;
}

int main()
{
    int x;
    int y;
    x = addInts(10, 20);
    printf("x = %d\n", x);
    y = addInts(200,
  "Hello");
}
```

Find the errors

```
a = [1, 2, 3, 4]

i = 0
while i < 4
  i=i+1
  print a[i]
```

One way to fix

```
a = [1, 2, 3, 4]

i = 0
while i < 4:
  print a[i]
  i=i+1
```

Lecture 01 Summary

- Administrivia
- Expectations
- Purpose of Programming
- Course Goals & Objectives
- Interpreters vs. Compilers
- Syntax Errors vs. Semantic Errors

Next Class

- C/Java Syntax