CITS 1401
Problem Solving & Programming

Ajmal Mian

Lecture 01 – Introduction

(These slides are based on John Zelle’s powerpoint slides for lectures accompanied with the text book)
Organization

- Unit coordinator
  - Assoc. Prof Ajmal Mian
  - Room 1.12
  - Consultation time: 1:00 – 2:00PM Thursdays
  - Website: http://www.csse.uwa.edu.au/~ajmal/

- Unit website: http://undergraduate.csse.uwa.edu.au/units/CITS1401/

- Two Lectures per week (1 hour each)
- Three Labs per week (3 hours each)
- One Workshop per week (1 hour)
Lectures

- Engineering Lecture Theatre 2 (ELT2)
- Mondays 1:00 – 1:50PM
- Wednesdays 1:00 – 1:50PM

- All lectures are recorded and will be available from LMS
- All lecture slides will be available from the unit website
Laboratory sessions

- Starting Week 2 in CSSE 2.03
- Mondays 2:00 – 4:45PM
- Tuesdays 9:00 – 11:45AM
- Wednesdays 2:00 – 4:45PM

- You are expected to attend at least one lab session
- You are welcome to attend as many lab sessions as you want
- Only the first two hours of each session are supervised
Workshop

- Thursday 4:00 - 5:00PM in ELT2 (starting week 2)
- Attendance is purely optional and on a need basis
- Friendly embarrassment free environment for asking questions
- Aimed at students who are having difficulty
- Basic questions will be given priority over advanced questions
- Additional examples complementing lectures
Online material

- Everything distributed in CIT1401 will be available on the unit website.
- Lectures will be available on LMS.
- For initiating online questions and discussions, use cshelp i.e. go to

https://secure.csse.uwa.edu.au/run/help1401
help1401

- READ FIRST: if the answer is not there, THEN POST

- Asking is useful - Sometimes formulating a question properly help you realize the answer yourself

- Answering is useful – explaining to others helps you understand better

- DO NOT post assessment code on help1401
Announcements

- Announcements will be made in any of the following three places
  - Lectures
  - Unit website
  - Help1401

- Information distributed by email will be sent to your UWA account

- Please check CITS1401 webpage + help1401 + your UWA email regularly

When an announcement has been made in these places, it will be assumed that you are aware of it
Text and other resources

- PYTHON PROGRAMMING: AN INTRODUCTION TO COMPUTER SCIENCE
  by JOHN ZELLE

  Publisher: Franklin Beedle & Associates
  ISBN-10: 1887902996

- Other resources
Development Environment

- In the lab you will use Python 3.3 IDLE
  - An integrated software development environment where you can write, edit, execute and debug programs

- IDLE is a free software available for all major operating systems such as Windows, OS, Linux

- You can download it from python.org and install it on your home computer

- Make sure that your submitted code (lab assignments etc) runs on the lab computers because that is where it will be tested
Assessment

- Assessment is based on both
  - Understanding of fundamental concepts
  - Practical problem-solving and programming skills

- Mid-semester test in Week 8 (worth 15%)

- Two programming projects
  - Project 1 due in Week 8 (worth 10%)
  - Project 2 due in Week 13 (worth 15%)

- Final exam in June (worth 60%)
Passing and satisfactory progress

- At least 50% overall, AND
- At least 40% in Project 1 and mid semester test combined, AND
- At least 40% in the final exam

CAUTION: 15-20% STUDENTS fail in CITS1401
Seeking help – options are many

- help1401
- Supervised labs
- Workshops
- Text and other resources
- Consultation times

Seek help **early** when there is a problem, you are not alone
All students are required to complete three online learning modules

- Academic Conduct Essentials
- Communication and Research Skills
- Indigenous Study Essentials

The above site also contains many other helpful resources
Make your Week 1 in the labs useful

- Get your computer account name and password
- Organize your UWA email account
- Obtain your timetable from OLCR
- Get familiar with the CSSE website
- Get familiar with the CITS1401 website
- Install Python 3 on your home computer/ laptop
LET US BEGIN
CITS1401

- CITS1401 is about problem solving and programming in general
  - Python is used as a platform

- CITS1401 covers
  - Important problem solving techniques used widely in computer science and programming
  - Writing basic program in Python (a modern high level language)
  - An introduction to software engineering

- Lectures, labs and the projects will complement each other.
Objectives of today's lecture

- To understand the roles of hardware and software in a computing system.
- To learn what computer scientists study and the techniques they use.
- To understand the basic design of a modern computer.
- To understand the form and function of programming languages.
The Universal Machine

- A computer is “a machine that stores and manipulates information under the control of a changeable program.”

- Two key elements
  - Computers are devices for manipulating information
  - Computers operate under the control of a changeable program
What is a computer program?

- A detailed, step-by-step set of instructions telling a computer what to do.

- If we change the program, the computer performs a different set of actions or a different task.

- The machine stays the same, but the program changes!
  - Hence what the machine does, also changes.
Computer programs

- Programs are executed, or carried out.

- Most computers have similar capabilities.

- With a suitable program, each computer can do the things other computers can do.

- In other words, the same program will produce similar results on different computers.
Program Power

- Software (programs) rule the hardware (the physical machine).

- The process of creating this software is called programming.

- Why learn to program?
  - It is a fundamental part of computer science.
  - Having an understanding of programming helps understand the strengths and limitations of computers.
What is Computer Science?

- It is NOT the study of computers!
  
  “Computers are to computer science what telescopes are to astronomy.”
  
  Edsger Dijkstra

- Since computer can carry out any process, the question really is “what process we can describe?”

- The fundamental question is “What can be computed”? 
What is Computer Science?

Computer scientists find the answers to questions through

1. Design
2. Analysis
3. Experimentation
Design

- One way to show a particular problem can be solved is to actually design a solution.

- This is done by developing an algorithm

- An algorithm is simply a “recipe”

- Algorithm: A step-by-step process for achieving the desired result

- CITS1401 teaches you how to
  - Design an algorithm
  - Write a program for it
Analysis

- “Design” can only answer the question “What is computable?” in the positive.
  - Not being able to design an algorithm does not mean it is unsolvable.

- Analysis is the process of examining algorithms and problems mathematically.

- Some seemingly simple problems are unsolvable by any algorithm.

- Problems can be intractable if they would take too long or take too much memory to be of practical use. [More on this in Week 12, Ch-13]
Experimentation

- Some problems are too complex for analysis.

- Implement a system and then study its behaviour.

- Experimentation is sometimes still needed after theoretical analysis
  - To verify the analysis
  - To refine the analysis

Thus, computer science is about designing, analyzing and evaluating algorithms
Hardware Basics

- The central processing unit (CPU) is the brain of the computer.

- The CPU carries out all basic operations on the data e.g. simple arithmetic.
Basic Hardware

- Memory stores programs and data.

- CPU can only directly access information from the main memory (RAM). Random Access Memory.

- RAM is fast but volatile i.e. all information is lost when power is lost.

- Secondary memory provides more permanent storage (non-volatile).
  - Magnetic (hard drive, USB drives)
  - Optical (CD, DVD, Blue Ray Disc)
Hardware Basics

- Input devices – pass information to the computer
  - Keyboards
  - Mice
  - Camera
  - Microphone

- Output devices – pass information back to the user
  - Monitor
  - Printer
  - Speaker
The Fetch Execute Cycle

- Load program into the main memory (RAM)
- Fetch the first (next) instruction from memory
- Decode the instruction to see what it represents
- Carry out the appropriate instruction.
- Repeat
Programming Languages

- Natural languages cannot precisely describe an algorithm.

- Programming languages can express algorithms in a precise way.

- Every structure in a programming language has a precise form called its syntax.

- Every structure in a programming language has a precise meaning called its semantics.
Programming Languages

- Programming language is like a code for writing instructions that the computer can understand and follow.

- Programs are sometimes called *computer code*.

- The process of writing an algorithm in a programming language is often called *coding*.
Programming Languages

- High-level programming languages
  - Designed to be understood and written by humans

- Low-level language
  - Computer hardware can only understand a very low level language known as machine language
High-level Programming Language

- In a high-level language, the addition of two numbers may be written as
  \[ c = a + b \]

- This needs to be translated to machine language so the computer can execute it

- **Compilers** convert programs written in a high-level language into machine language
Low-level Language

- The corresponding low-level language may look something like this

  - Load the number from memory location 5001 into the CPU
  - Load the number from memory location 5002 into the CPU
  - Add the two numbers in the CPU
  - Store the result into location 5003

- In reality, these low-level instructions are represented in binary (1’s and 0’s).
Interpreters

- Interpreters simulate a computer that understands a high-level language.

- The source program is not translated (compiled) into machine language all at once.

- An interpreter analyzes and executes the source code instruction by instruction.
Compiling vs. Interpreting

- Once program is compiled, it can be executed over and over without the source code or compiler

- Compiled programs generally run faster since the translation of the program happens only once

- Program needs to be compiled after every minor change in it

- A program compiled for Windows will not run on OS (Mac) or Linux

- The source code and interpreter are needed each time the program is executed

- Interpreted programs run slower due to the interpretation each time it is executed

- More flexible programming environment since programs can be developed and run interactively

- Interpreted programs are more portable across different platforms e.g. Macs, Windows, Linux
Python

- Python is an interpreted language.

- If a suitable interpreter exists for two different machines with very different CPUs, the same program will run on both.
Python

- When you start Python, you will see something like:

```
Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```

- The ">>>" is a Python prompt indicating that Python is ready for us to give it a command. These commands are called statements.

```
>>> print("Hello, world")
Hello, world
>>> print(2+3)
5
>>> print("2+3=", 2+3)
2+3= 5
>>> 
```
Summary

- Understanding the roles of hardware and software in a computing system.

- Learning what computer scientists study and the techniques they use.

- Understanding the basic design of a modern computer.

- Understanding the form and function of programming languages.