

CMPT 125

Introduction to Computing Science and Programming II

<https://coursys.sfu.ca/2019sp-cmpt-125-d1/pages/>

Lecture 1 Plan

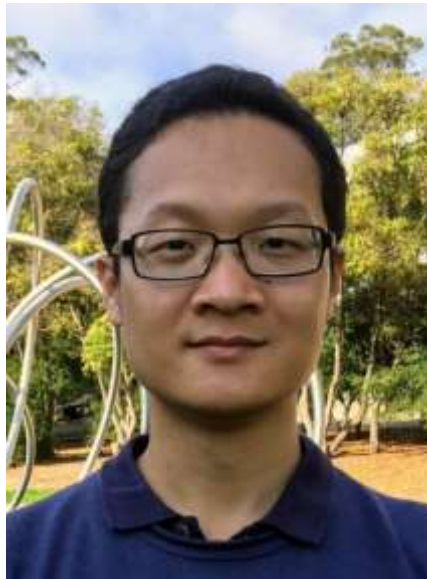
Today:

- Introductions
- CMPT 125 vs CMPT 127
- Grading scheme
- Other expectations
- Computer science review / overview
- Running a program in C

Introductions

Two instructors for two courses!

CMPT 125 and CMPT 127 (co-taught)



Mo Chen



Anne Lavergne

CMPT 125 vs CMPT 127

- Co-requisite courses
 - You must take them as a pair
 - Separates theory from practice
- CMPT 125 will be focused on algorithms, computer science, analysis
 - Assignments help you understand theory
- CMPT 127 will be focused on writing code, debugging, testing
 - Assignments help you understand coding and implementation

CMPT 125

- Lectures MWF 15:30-16:20
 - RCB Images Theatre
- Weekly assignments (40% of grade)
 - Due Wednesdays, no late assignments accepted
 - Omit assignment with lowest grade
- In-class Midterm (20% of grade)
 - Mar. 6
- Final (40% of grade)
 - Date and location TBA

Online Interactions via Piazza

- Piazza sign up
 - piazza.com/sfu.ca/spring2019/cmpt125
- Online question and answer platform
 - Questions can be asked anonymously
 - Students can answer each other's questions
 - Instructors can also approve student answers
- Instructors will answer questions and monitor questions and answers daily
- **Please use this first!**

Simon Fraser University

(change school)

Are you a professor?

[Click here to create & join classes](#)

Selected Term: Spring 2019



Spring 2019

Class 1: CMPT 125: Introduction to Computing Science and Programming II (edit)

Instructors: Mo Chen · 1 Enrolled

✓ **Join as:** ☒ Student ☐ TA ☐ Professor

Class 2:



Class 3:



Class 4:



Class 5:



Office Hours

Prof. Mo Chen: Thursdays at 14:00 - 15:00, TASC 1 8225

Ekramul Hoque: Tuesdays, 3:00 pm - 4:00 pm, ASB 9810

Farzad Sharifbakhtiar: Wednesdays 15:00-16:00, ASB 9808

Muhammad Rafay Aleem: Fridays 9:00-10:00, ASB 9810

Anjian Li: Fridays 10:00 - 11:00, ASB 9810

Xubo Lyu: Fridays 13:30-14:30, ASB 9810

Other Expectations

By the end of these two courses you can expect to be able to:

- write high quality code in C
- use standard command line tools in Linux
- use version control to manage your work
- develop algorithms to solve problems
- predict the behaviour of algorithms

Other Expectations

From CMPT 120, it's assumed that you are proficient at the basic concepts of programming.

- Data types and conversions (integer, float, string)
- Expressions
- Basic terminal input/output (`raw_input()` and `print`)
- Libraries (import from modules)
- Conditionals (if-elif-else)
- Definite loops (for) and indefinite loops (while)
- Functions and parameter passing
- The [develop → test → debug] cycle

Other Expectations

You are not expected to know the C syntax for these concepts, only that you know the concepts

- Over the first few weeks, you will learn how they are expressed in C

Other Expectations

Our expectations of you:

- 10 hours per week per course
 - standard workload for SFU courses
- CMPT 125 = 3 hours lecture + 7 hours reading / studying / solving assignment problems
- CMPT 127 = 3 hours lab + 7 hours of coding / experimenting / benchmarking
- **Reflect on how you learn**

Other Expectations

RESPECT

Theme: Do not interfere with the learning of others.

- show up to class on time
- no talking during class [about non lecture-related material]
- no texting / Facebook / youtube in the e-free zone - sit in the back row of class if you **must** do this
- complete / submit your OWN work == be academically honest

Bottom line: Do not interfere with the learning of others.

Course Objectives / Outline

Summary

- Two courses; two co-instructors
- Lecture course is computer science focused
- Lab course is computer programming based
- Both courses are fundamentals - put in the time and your future work will be easier
- Respect your classmates, both inside and outside of the classroom / lab.

Any questions?

Computer Science Review / Overview

What is Computer Science?

[From CMPT 120]

- The study of algorithms, their formal and mathematical properties, their hardware realizations, their linguistic realizations, and their applications.

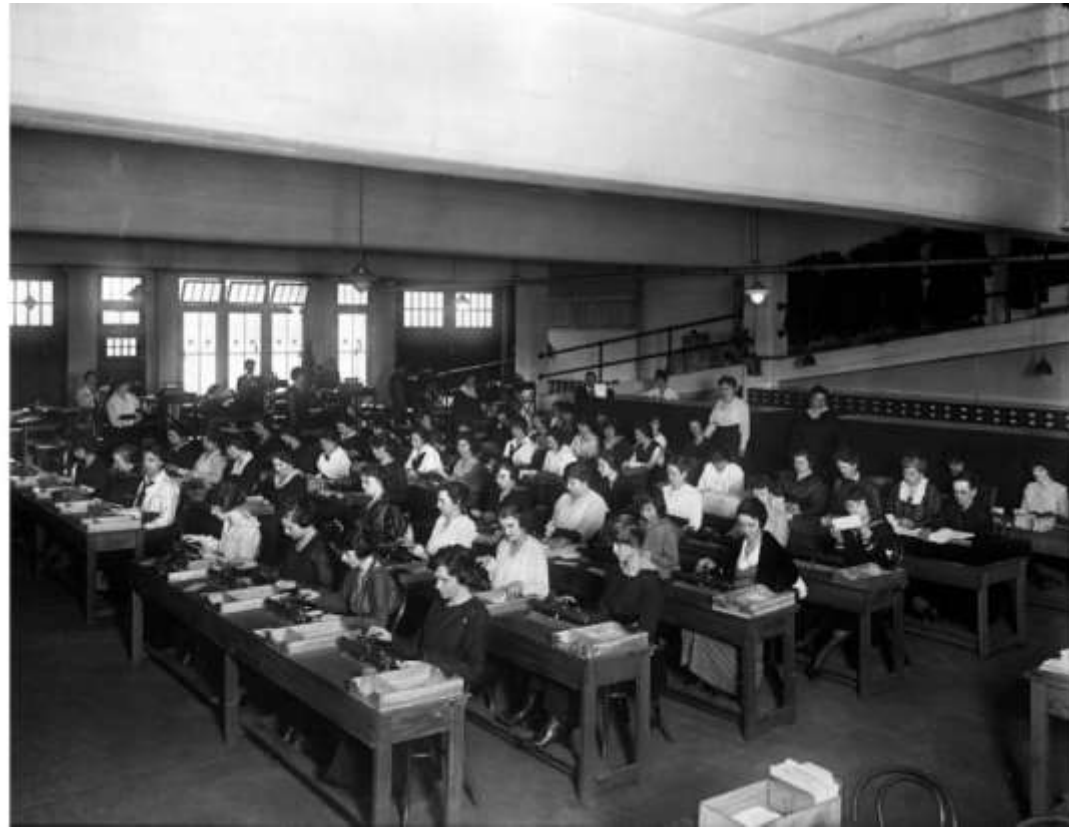
[From real life]

- The study of what computers can and cannot do.

Computer Science Review / Overview

The very first computers were utilized to perform pure calculation: tables for $\sin(x)$, $\cos(x)$, $\log(x)$

- Human calculators replaced by automation!
- “Calculator” and “Computer” used to be job titles!



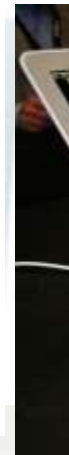
Computer Science Review / Overview

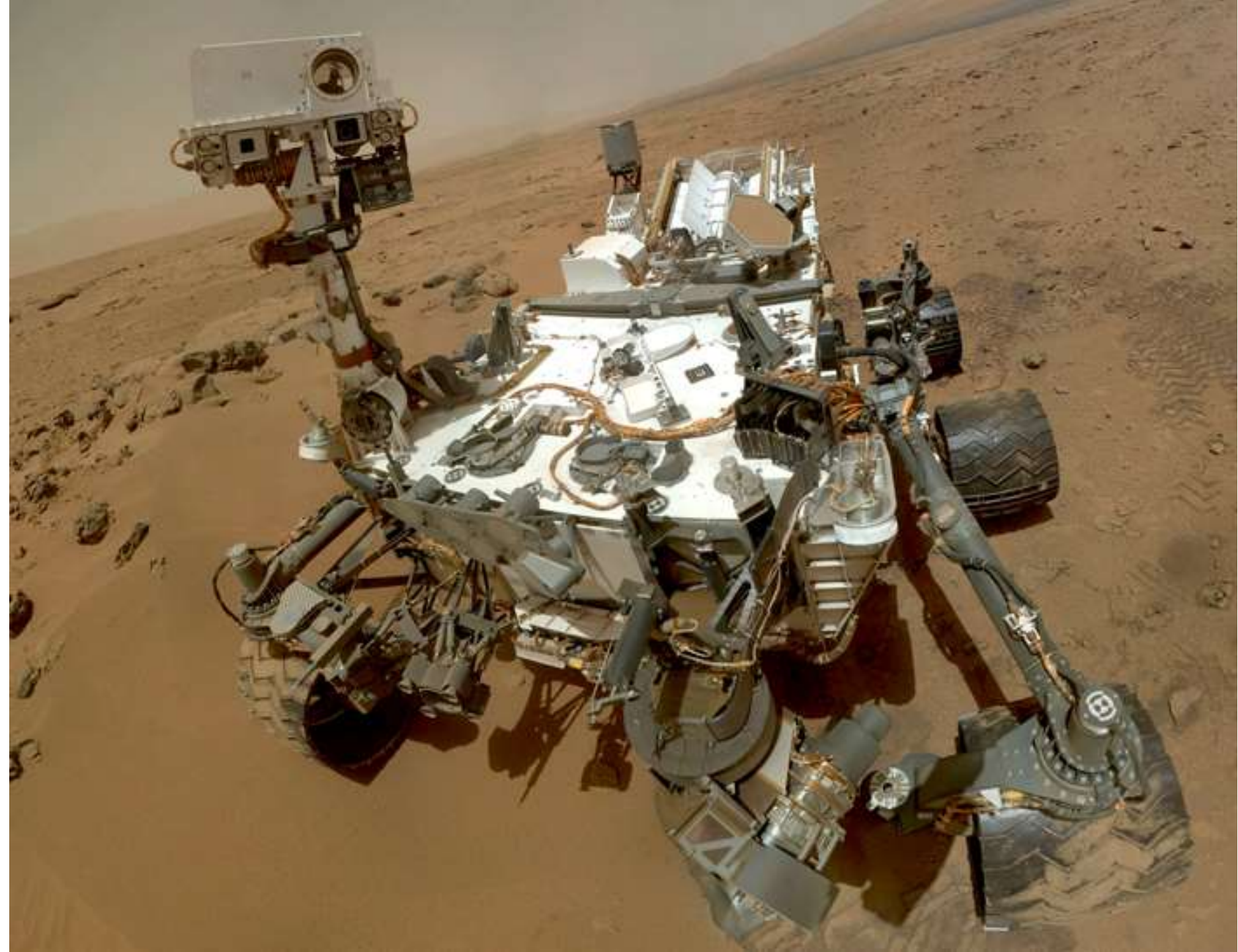
Computing is applied everywhere

- big - mainframes, supercomputers
- medium - desktop, laptop, tablet
- small - smartphones, cars, microwaves

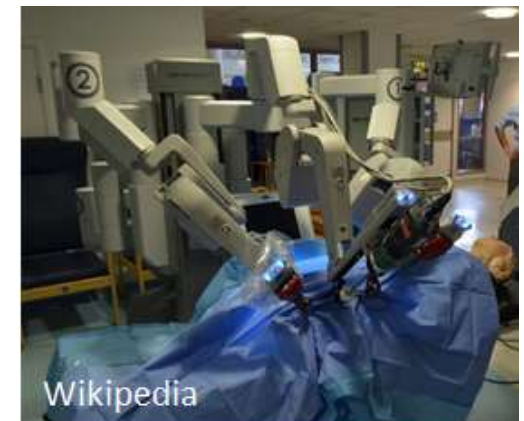
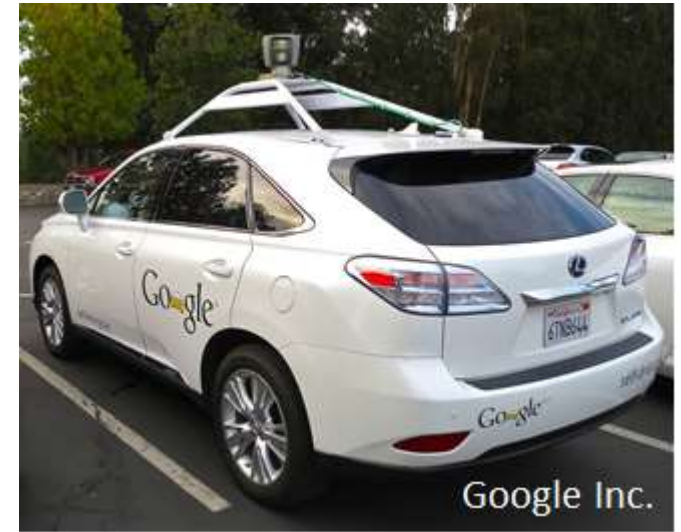
Automating more and more of our society







Computer Science Review / Overview



Computer Science Review / Overview

Algorithms are the core component.

- Definition: An *algorithm* is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining some required output for any valid input in a finite amount of time.
- You communicate algorithms to computers using a programming language.

Computer Science Review / Overview

programmingLanguageForCMPT125And118 = C;

- Well, it will be mostly just C, but seasoned with some of the elements of C++.

Why C?

- Because it is everywhere
- Because it is fast

Running a Program in C

3 Steps:

1. Edit your program.

- Use “`gedit`”.
- Save in a `.c` file.

2. Compile your program.

- Use “`gcc program.c`” ...
- ... to generate “`a.out`”.

3. Run your program.

- Use “`./a.out`”.

Preparation for the lab

Login to CSIL and do some file manipulations in preparation for your first lab next week.

You will:

- Use the terminal
- Create new directories (folders)
- Create new files
- Upload them to your personal repository
- Upload them to the CourSys management system.

Running a Program in C

Step 1: `gedit`

or . . .

Running a Program in C

Step 1: `gedit program.c`

Running a Program in C

`gedit`

- a simple editor, like Notepad (Windows) or TextEdit (Mac)
- does text highlighting for C syntax

Other Linux editors:

- `emacs`
- `vim`
- `sublime`

Running a Program in C

```
#include <stdio.h>
```

“#include” in C is like “import” in Python

Running a Program in C

```
#include <stdio.h>
```

```
int main() {
```

```
}
```

This is your main function - it is always where your program starts its execution.

Running a Program in C

```
#include <stdio.h>
```

```
int main() {
```

```
}
```

Curly braces { } denote a block of code.
(Like block indentation does for Python.)

Running a Program in C

```
#include <stdio.h>
```

```
int main() {  
    printf("Hello world\n");  
}
```

- `printf(...)` is your output function.
- All statements end with a semicolon “;”.
- Newlines are not automatic: use “\n”.

Running a Program in C

Save your program as a `.c` file

Open a console window to get to the command prompt, and run the *C compiler*

```
>$ gcc program.c
```

```
>$
```

If successful, creates an executable program called `"a.out"`.

Running a Program in C

You are finally ready to run your program!

Type “`./a.out`” as your next command

```
>$ gcc program.c
```

```
>$ ./a.out
```

```
Hello world!
```

```
>$
```

<https://coursys.sfu.ca/2019sp-cmpt-125-d1/pages/>