SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 120: Introduction to Computing Science and Programming 1

Algorithms, Flowcharts and Pseudocodes

One-Stop Access To Course Information

- Course website: One-stop access to all course information.

http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html

- Course Outline
- Exam Schedule
- Python Info
- CourSys/Canvas link
- Learning Outcomes
- Office Hours
- Textbook links
- and more…
- Grading Scheme
- Lab/Tutorial Info
- Assignments

- Canvas: Discussions forum.

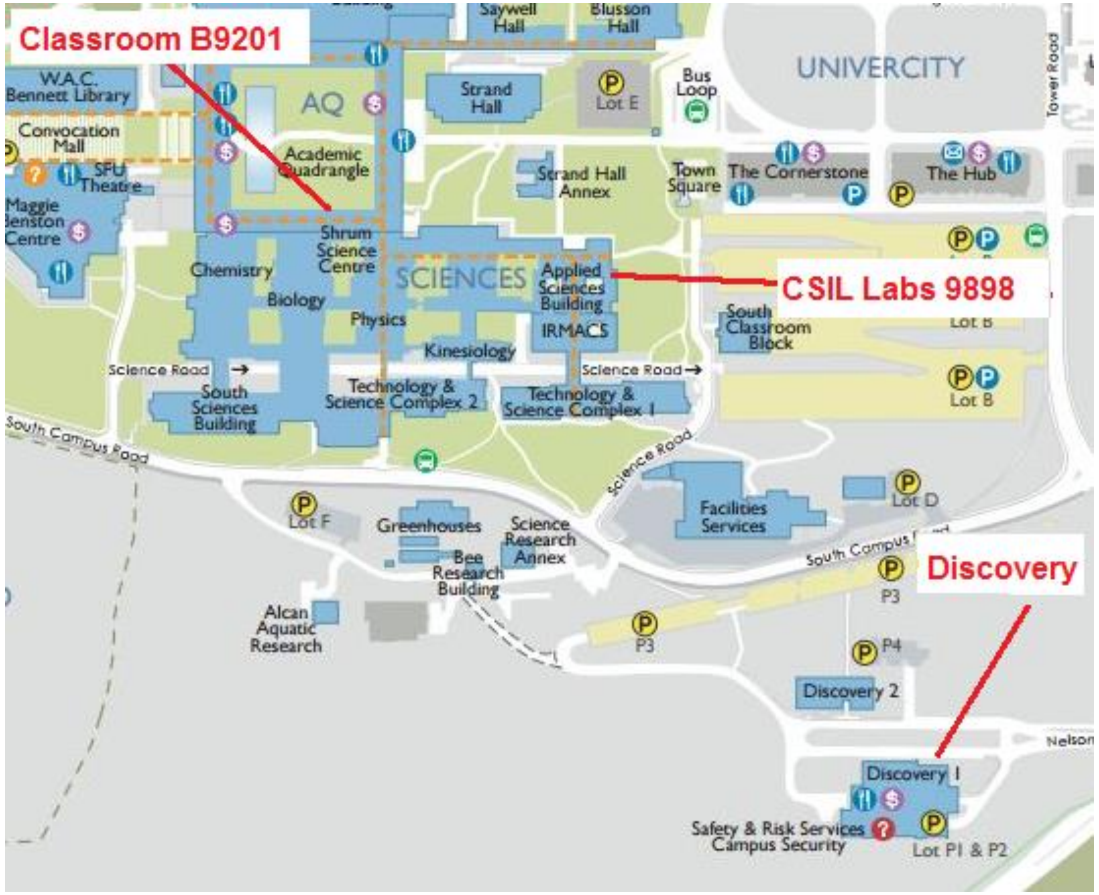
  https://canvas.sfu.ca/courses/39187

- CourSys: For assignments submission, and grades.

  www.coursys.sfu.ca

Some Reminders

- **Get familiar with the course Website.**
  - http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html
  - Minor updates may occur during first week.
- **Get fob to access LABS (start next week!)**
  - If you don’t have it already, get a new fob from Discovery Park 1 .


Classroom B9201

CSIL Labs 9898

Discovery

Additional Resources / Online References

- There are several online references that are **as important as the texts**. (Links provided on the course web site.)
- These resources are **very important to your success** in this course. They aren't meant to be read from beginning to end like the readings in the textbook.
- You should **use them to get an overall picture of the topic** and as references as you do the assignments.

How to Learn in This Course?

A **Attend** Lectures & Labs

R **Read** / review Textbook/Slides/Notes

R **Reflect** and ask Questions

O **Organize** – your learning activities on weekly basis,

and finally…

W **Write** Code, Write Code, and Write Code.

Today’s Topics

1

Algorithms?

Algorithm: Its Definition and Key Properties - 1

- During the last lecture, we talked about algorithms.
- Now, let's have a look at a couple of more definitions.

An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

[Source: CMPT 120 Study Guide; Anany Levitin, Introduction to The Design & Analysis of Algorithms, p. 3]

Algorithm: Its Definition and Key Properties - 2

- An algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.
  - An algorithm is a sequence of computational steps that transform the input into the output.

[Source: Thomas H. Cormen, Chales E. Leiserson (2009), Introduction to Algorithms 3rd edition.]

Algorithm: Key Properties

- **Unambiguous**: Each step of an algorithm has to be precisely defined.
    - **After reading an algorithm, there should be no question about what to do.**
- **Specific problem**: An algorithm should always present a solution to a particular problem, or group of problems.
- **Legitimate input**: An algorithm might need some kind of input to do its job. This input should be relevant.
- **Finite amount of time**: If started, an algorithm must end eventually. If it never ends, it's useless.
- **Clear I/O**: Inputs and outputs should be defined clearly.
- **Effective**: Should be effective among many different ways to solve a problem.

Algorithm: Watch A Video

- Let’s watch this short video about algorithms.
- You will hear two new terms related to algorithms in this video. Let’s see if you can note them down.
  - What’s an algorithm?
  - https://study.com/academy/lesson/what-is-an-algorithm-in-programming-definition-examples-analysis.html#lesson

Algorithm: The Two New Terminologies

1. Pseudocode: A semi-programming language used to describe the steps in an algorithm.
2. Flowchart: A diagram used to represent the steps used in an algorithm.

- We will talk about these terms later. Let's do some examples of algorithms.

Algorithm: Add Two Numbers Entered by a User

```
Step 1: Start
Step 2:   Suppose, N1 is the first number.
Step 3:   Suppose, N2 is the second number.
Step 4:   Suppose, SUM is the sum of two numbers.
Step 5:   Get the value of N1 from the user.
Step 6:   Get the value of N2 from the user.
Step 7:   Add N1 and N2 and assign the result to SUM.
              SUM ← N1 + N2
Step 8:  Display SUM
Step 9: End
```

Algorithm: Verify the Properties

```
Step 1: Start
Step 2:   Suppose, N1 is the first number.
Step 3:   Suppose, N2 is the second number.
Step 4:   Suppose, SUM is the sum of two numbers.
Step 5:   Get the value of N1 from the user.
Step 6:   Get the value of N2 from the user.
Step 7:   Add N1 and N2 and assign the result to SUM.
                SUM ← N1 + N2
Step 8:  Display SUM
Step 9: End
```

1. Is it Unambiguous?
2. Solves specific problem?
3. Legitimate input?
4. Finite time?
5. Clear I/O?
6. Is it effective?

Algorithm: A Few Computing Science Terminologies

- In Computing Science, we usually don't write "**suppose**". Rather, we typically say "**declare**".
- We call N1, N2, and SUM as "**variables**".
  - And, variables typically "**store**" values.

So, We may choose to re-write the step: Suppose, N1 is the first number.

As: Declare a variable N1.

Or, Declare a variable N1 to store the value of first number.

Or, Declare a variable N1 to store the value of the first number entered by the user.

Re-Write the Add Two Numbers Algorithm

Re-write the following “add two numbers algorithm” replacing the words declare, variable and store, as necessary.

Step 1: Start

Step 2: Suppose, N1 is the first number.

Step 3: Suppose, N2 is the second number.

Step 4: Suppose, SUM is the sum of two numbers.

Step 5: Get the value of N1 from the user.

Step 6: Get the value of N2 from the user.

Step 7: Add N1 and N2 and assign the result to SUM. $SUM \leftarrow N1 + N2$

Step 8: Display SUM

Step 9: End

Re-Write The Add Two Numbers Algorithm

```
Step 1: Start
Step 2:     Declare a variable N1 to store the first number.
Step 3:     Declare a variable N2 to store the second number.
Step 4:     Declare a variable SUM to store sum of numbers N1 and N2.
Step 5:     Get the value of N1 from the user.
Step 6:     Get the value of N2 from the user.
Step 7:     Add N1 and N2 and assign the result to SUM.
                $SUM \leftarrow N1 + N2$
Step 8:     Display SUM
Step 9: End
```

Algorithm: Find the Smaller of Two Numbers

Write an algorithm to find the smaller of two numbers entered by a user.

```
Step 1:  Start
Step 2:      Declare a variable num1 to store the first number.
Step 3:      Declare a variable num2 to store the second number.
Step 4:      Get the value of num1 from the user.
Step 5:      Get the value of num2 from the user.
Step 6:      If num1 < num2 then print num1 is smaller.
Step 7:      If num2 < num1 then print num2 is smaller.
Step 8:      If num1 = num2 then print “Both the numbers are equal.”
Step 9:  End
```

Algorithm: Find the Smallest of Three Numbers

- Write an algorithm to find the smallest of three numbers entered by a user.

- Solution in the next class.

Questions?

Course Topics

1. General introduction
2. Algorithms, flow charts and pseudocode
3. Procedural programming in Python
4. Data types and control structures
5. Fundamental algorithms
6. Binary encodings
7. Basics of computability and complexity
8. Basics of Recursion
9. Subject to time availability:
   - Basics of Data File management