CMPT 120


Source: https://twitter.com/MEKTORY/status/893408704590204928
Last Lecture

✓ Introducing the course
  ✓ What is this course all about?
  ✓ What kind of students take CMPT 120?
  ✓ What do we need for this course?
  ✓ How does this course work?

✓ Q&A -> Waiting list

✓ Let’s get started!
  ✓ How does a computer work?
    ✓ Data stored in Memory
  ✓ Python demo
How to do well in this course?

• Not about “remembering” but about “doing”
• We can create a cheat sheet for our exams
  • Can start now gathering info for our cheat sheet
Today’s Menu

- Problem Solving
- Software development process
  - Natural and Formal languages -> see our Readings
  - Algorithm and programming language
  - Our first program
    - Comments + header comment block
    - Python
      - `print()` and `input()` function
      - strings
      - variables
      - assignment operator `=`
    - Execution flow
  - Interpreted program versus compiled
What is this course all about?

- Title: “Introduction to Computing Science and Programming 1”

Learn software development using the programming language **Python**

Learn fundamental concepts of **Computing Science** e.g. problem solving, algorithms, and get a sneak preview of some of its most interesting topics such as big data and graphics 😊
Solving a problem in the “real” world

What are the steps we go through when we solve a problem?

Source: https://corporatetrainingmaterials.com/course/Creative_Problem_Solving
https://www.sandler.com/blog/bring-problem-prospect-does
Solving a problem in the “computer” world

Steps of Software Development Process:

1. Problem statement
   - What's the problem?
2. Design
   - How could we go about solving it?
3. Implementation
   - Let write some Python code!
4. Testing
   - Does it work?
5. Solution

Source: https://corporatetrainingmaterials.com/course/Creative_Problem_Solving
https://www.sandler.com/blog/bring-problem-prospect-does
What is an algorithm?

- A **finite sequenced** set of **unambiguous** steps that, once executed, produces a **result**
  - **Finite**: This set of steps executes in a finite amount of time i.e. it should finish at some point
  - **Sequenced**: The steps must be executed in the order in which they are listed
  - **Unambiguous**: Each step is clear
  - **Result**: This result solves the initial problem

- The algorithm also describes
  - The data it needs in order to work -> input
  - The result it produces -> output
More about algorithms

• Video *Algorithms* from Khan Academy

For your viewing pleasure
How do we express an algorithm?

1. Use a natural language like English
   • Example – problem: compute final course grade
2. Use a mix of natural language and computer language -> pseudocode
   • Example – problem: compute final course grade
3. Use a flowchart
   • Example – problem: compute final course grade
4. In a diagram
   • Example – problem: build a Billy shelving unit
5. In a video (verbal instructions and pictures)
   • Example – problem: make tea
Programming language

• This semester, we’ll be learning a new language

• What is a programming language?
  • Definition: is a language that allows us to communicate with the computer, specifying detailed instructions that a computer can understand and execute
  • A programming language, like a natural language, is made of
    1. Vocabulary ("building blocks")
    2. Syntax rules (grammar)

• Python, C++, Javascript are programming languages
The first programmer

- In 1842, Lady Ada Lovelace wrote the first computer program for Charles Babbage's Analytical Engine (1837)

Source:
http://collection.sciencemuseum.org.uk/objects/co62245/babbages-analytical-engine-1834-1871-trial-model-analytical-engines
Let’s give it a go! – Take 1

1. **Problem Statement**
   - Write a Python program using IDLE. Your program must display **Hello, World!** on the computer monitor screen.

2. **Design**

3. **Implementation**

4. **Testing**

Source: https://en.wikipedia.org/wiki/%22Hello,_World!%22_program
Let’s give it a go! – Take 2

1. **Problem Statement:**
   - Create a greeting chatbot that greets the user.

**Requirements:**
- It must include the user’s name as part of the greeting
- We must use Python 3.12 IDLE

2. **Design:**

3. **Implementation:**

4. **Testing:**
Comments in Python

Syntax:

# some comments

or

Python statement  # some comments

The Python interpreter ignores (i.e., does not interpret) anything written to the right of the # character, all the way to the end of a line.

How to create comments in our Python program

• We can use the steps of our algorithm as comments in our Python program

Reasons for using comments in our Python program

1. Explaining what the statements of our program do
2. Temporarily "removing" code from our program without deleting it, as we are developing and debugging our program.
**Header Comment Block**

- **Purpose:** Give information about our program
- **Composed of:**
  - Filename
  - Description of program
  - Author
  - Date of creation or modification
- **Location:** At the very top of our program
- **Execution?**
  - Since we start each line of our header comment block with a # sign, i.e., making each line a comment, this signifies that the Python interpreter skips the entire header comment block and start executing the first non-comment line below it
# FinalCourseGrade.py
#
# Compute course final grade from 3 activities:
# - Assignment 1 -> Assn1
# - Midterm examination -> MT
# - Final examination -> FE
#
# Anne Lavergne
#
# May 2017

... rest of program
Interpreted program versus compiled program

Interpreted program

Source: https://www.blueridge.edu/programs-courses/interpreter-american-sign-language/interpreter-american-sign-language-asl/
Interpreted program versus compiled program

Compiled program

Source: https://www.commoncraft.com/blog-categories/translations
Our first Assignment (0)

• Has now been posted!
• Let’s check it out!
Review Questions

1. What is the single most important skill for a computing scientist?
2. Comments and __________ __________ can make your programs much easier for humans to parse. Use them liberally!
3. What do natural languages and formal languages Have in common?
4. How can I express an algorithm?
5. How would I display \( 4 + 7 = 12 \) on the screen?
Review Questions

6. What are the steps we performed when we solve a problem using software?

7. What do we put in a header comment block and why?

8. What would `print("Above\nBelow")` produce on the screen?

9. Give an example of an algorithm?

10. Why do we need algorithms?
Summary

• Problem Solving
• Software development process
  • Natural and Formal languages
  • Algorithm and programming language
  • Our first program
    • Comments + header comment block
    • Python
      • `print()` and `input()` function
      • strings
      • variables
      • assignment operator `=`
    • Execution flow
  • Interpreted program versus compiled
Next Lecture

• Introducing Automation – Chatbots

• Can we build chatbots using
  • `print( ... )` and `input( ... )`
  • variables
  • strings
  • and more ...

• We shall see ☺