


UCT Department of Computer Science
Computer Science 1015F

Introduction to Computing

Aslam Safla<aslam@cs.uct.ac.za>
(thanks to Hussein Suleman <hussein@cs.uct.ac.za>)

February 2019

What is Computer Science

What is Computer Science?

- Computer Science (CS) is the study of:
  - Computer software
  - Algorithms, abstractions and efficiency
  - Theoretical foundation for computation

- What you learn in Computer Science:
  - Principles of computation
  - How to make machines perform complex tasks
  - How to program a computer
  - What current technology exists and how to use it
  - Problem solving

Problem Solving in CS 1/2

1. Understand the problem
    1. What are the knowns and unknowns?
2. Plan how to solve the problem
    1. What algorithm is used to solve the problem?
    2. What assumptions are being made?
    3. Is this similar to other problems?
    4. Can the problem be split into parts?
3. Carry out your plan – write program
    1. Write program(s) to implement algorithm(s).

Problem Solving in CS 2/2

4. Assess the result
1. Does the program conform to the algorithm?
2. Does the program/algorithm solve the problem?
3. Is the program correct for all cases?
5. Describe what you have learnt
1. ... so you do not make the same mistakes again.
6. Document the solution
1. Write a report for users of the program.
2. Write comments within the program.

Reference: Vickers, P. 2008. How to think like a programmer. Cengage.

Algorithms

- An **algorithm** is a set of steps to accomplish a task.

- Everyday tasks require algorithms but we usually do not think about them.
  - E.g., putting on shoes, brushing teeth

- Algorithms must be precise so that they are
  - Repeatable
  - Have a predictable outcome
  - Can be executed by different people

Algorithm: Read a Novel

1. Acquire book
2. Find comfortable spot to sit
3. Open book to set of facing pages
4. If there are no more unread pages, go to step 8
5. Read facing pages
6. Turn page over
7. Go to step 4
8. Close book
9. Be happy

Elements of Algorithms

- Sequence
  - Each step is followed by another step
- Selection
  - A choice may be made among alternatives
- Iteration
  - A set of steps may be repeated

- Any language with these 3 constructs can express any classical algorithm.

Classic Problems / Algorithms

- Boil water in a kettle
- Take the minibus taxi to town
- Put on a pair of shoes
- Bake a cake
- Making a telephone call
- Buying a #1 Original Chicken Burger

Algorithm to Boil Water in Kettle

1. Take the lid off kettle
2. If there is enough water already, go to step 7
3. Put kettle under tap
4. Open tap
5. While kettle is not full,
   - Wait
6. Close tap
7. Replace lid on kettle
8. Plug kettle into power outlet
9. Turn kettle on
10. While water has not boiled,
    - Wait
11. Turn kettle off
12. Remove plug from power outlet

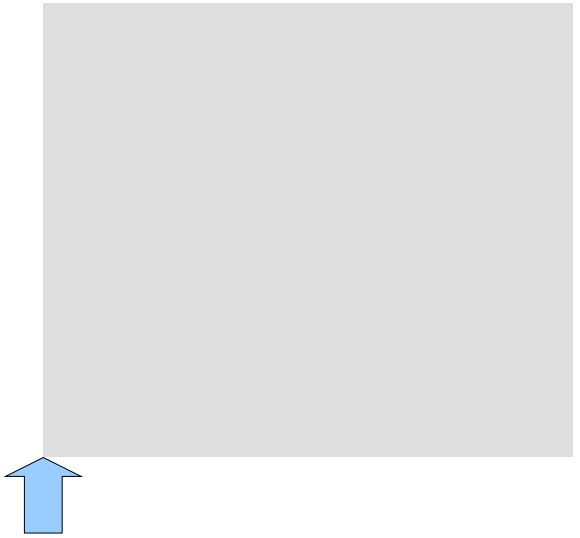
Algorithm: Take Minibus Taxi to Town

1. Make sure you have enough money
2. Wait at bus stop
3. Flag down taxi as it approaches
4. Get into taxi (somehow)
5. Collect fare from behind you, add your money and pass it forward
6. Shout at driver to stop
7. When taxi stops, prod other passengers to make them move out
8. Get out of taxi
9. Give thanks for a safe trip!

Can we be more precise?

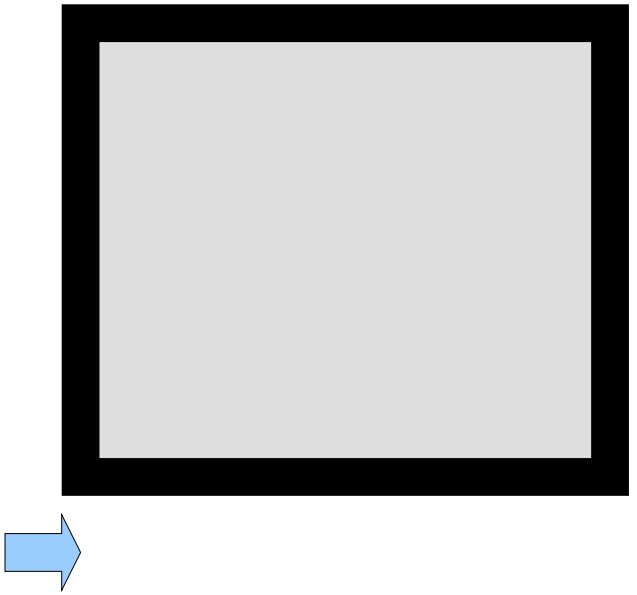
- Let us make up a precise drawing language (inspired by Turtle/Logo).

- Suppose we have an invisible box 10cm square, and we start at the bottom left corner, facing up.
- We have 2 instructions:
  - Draw <centimetres>
  - Draw a line
  - Spin <degrees>
  - Turn to the right



Drawing Example

- Draw 10cm
- Spin 90
- Draw 10cm
- Spin 90
- Draw 10cm
- Spin 90
- Draw 10cm



Drawing Exercise 1

- What does this draw?

- Spin 90
- Draw 10cm
- Spin 180
- Draw 10cm
- Spin 90
- Draw 10cm
- Spin 90
- Draw 10cm

Drawing Exercise 2 (1/3)

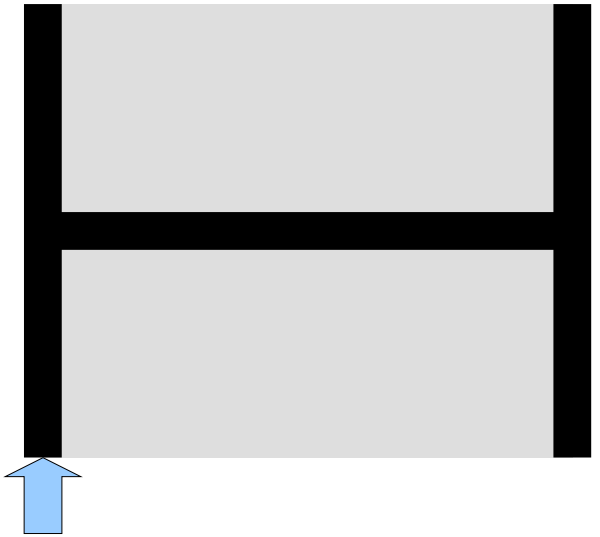
- This exercise is a 2-person task.

- Person A will be the algorithm designer (aka the programmer).
- Person B will be the algorithm implementer (aka the computer).

- At first everyone is Person A - then Person B.
- Some pairs of volunteers will do the task up-front where the roles are distinct.

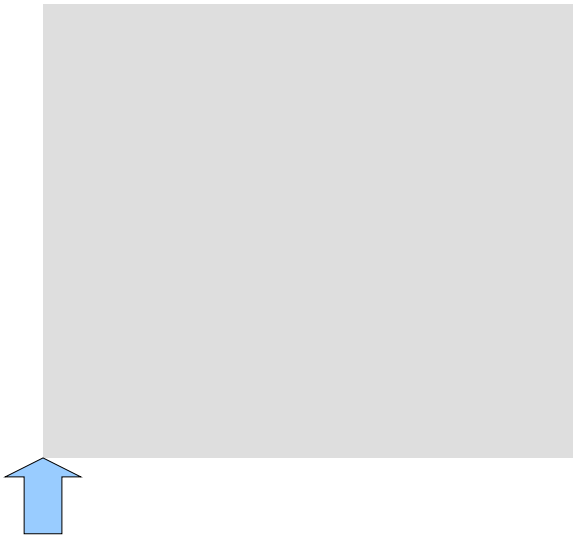
Drawing Exercise 2 (2/3)

- Person A: Write down instructions (in our special language) to draw this shape.
- You have 2 minutes!



Drawing Exercise 2 (3/3)

- Swap your instructions with someone else.
- Person B: Draw this shape using Person A's instructions.
- You have 2 minutes!



Programs

- A **program** is a set of instructions given to a computer, corresponding to an algorithm to solve a problem.
  - The act of writing a program is called **programming**.
- Programs are written in a precise language called a **programming language**.

- Sample Program (in Python):

```
letters = [32, 85, 67, 84]
for a in [7036462799701,4329864565765,4329327694853,
          4329864565765,4329505817940]:
   while a>0:
      print (chr(letters[a & 3]),end="")
      a = a >> 2
   print ()
```


Question

- How is an algorithm different from a program?

Process of Programming

- Programs work as follows:
  - Ingest information from the real world (**input**).
  - Process data internally.
  - Send computed data back to real world (**output**).

- Because of different input, each time a program executes the results can be different.

Python

- There are many different types of computer languages, and many different languages.
- This course is based on Python.
- Python is a general-purpose interpreted programming language invented in the 1980s/1990s by Guido van Rossum at CWI.
- We use version 3 because it is easier to learn.

How We Program in Python

- We write **programs,** stored in text files.
- Each program is a set of instructions that the **Python interpreter** will execute when the program is executed by the user.
- We often do both of these things in an **Integrated Development Environment (IDE)**.

- We can also use the interactive interpreter to run short programs while testing our ideas.
- Later, we will neaten our code into blocks called functions.
- Python is an OOP language but we will not use this.