WHAT IS AN ALGORITHM?

Slides by Björgvin B. Björgvinsson

WHAT IS AN ALGORITHM?

- An algorithm is "the process, or a set of rules to be followed in calculations or other problem-solving operations"
- More informally
  - A recipe for solving a problem
- A recipe for baking a cake is an algorithm
  - It is a set of rules you must follow and execute in a specific order to reach a desired result

WHAT IS AN ALGORITHM?

- An example of an algorithm that computes the square root of a numbers is the following:
    1. Guess the square root of the number
    2. Divide the working number by the guess
    3. Average the quotient (from 2) and the guess
    4. Make the new guess the average from step 3
    5. If the new guess is "sufficiently different" from the old guess, go back to step 2, else halt.

WHAT IS AN ALGORITHM?

- What is the difference between a program and an algorithm?
  - An ***algorithm*** is a description of how to solve a problem
  - A ***program*** is an implementation of an algorithm in a particular language to run on a computer (usually a particular kind of computer)
  - The difference lies in ***what we want to do (the algorithm)*** and ***what we actually do(the program)***

WHAT IS AN ALGORITHM?

- We can analyze an algorithm independently from its implementation. This is a big part of the science in Computer Science
- We can examine how easily, or with what difficulty, a language allows us to realize an algorithm
- We can examine how different computers impact the realization of an algorithm
- An algorithm can be implemented in many ways

WHAT IS AN ALGORITHM?

- ***An algorithm should be:***
  - ***Detailed*** : Provide enough detail to be implementable. Can be tricky to define completely, relies on "common sense"
  - ***Effective***: The algorithm should eventually halt, and halt in a "reasonable" amount of time. "reasonable" might change under different circumstances (faster computer, more computers, etc.)
  - ***General Purpose***: Algorithms should be idealized and therefore general purpose. A sorting algorithm should be able to sort anything (numbers, letters, patient records, etc.)

ASPECTS OF A PROGRAM

- When writing program it is important to be aware of that the code you write will be read by other people, even if the "other people" are you in the future!
  - Write a program so that you can read it, because it is likely that sometime in the future ***you will*** have to read it!
- Choose descriptive names for variables and functions and so on
  - use names for the items you create that reflect their purpose
  - It is all right if the names are long, as long as they are descriptive

ASPECTS OF A PROGRAM

- These two programs do the same thing! Which one is more readable?

```
a = input("give a number: ")
b,c=1,0
while b<=a:
    c = c + b
    b = b + 1
print(a,b,c)
print("Result: ", c/b-1)
```

```
limit_str=input("Range is from 1 to your input:")

limit_int = int(limit_str)
count_int = 1
sum_int = 0
while count_int <= limit_int:
        sum_int = sum_int + count_int
        count_int = count_int + 1
average_float = sum_int/(count_int - 1)
print("Average of sum of integers from 1 to",limit_int,"is", average_float)
```

ASPECTS OF A PROGRAM

- Comments
  - Comments can be good and they can be bad
  - “The code is the comment” is often said about code that is easily readable
  - Don´t overdo the comments
  - But if you write anything ***tricky*** bear in mind that If it took you a while to write, it is quite possible that it is hard to read it and therefore it might need a comment

ASPECTS OF A PROGRAM

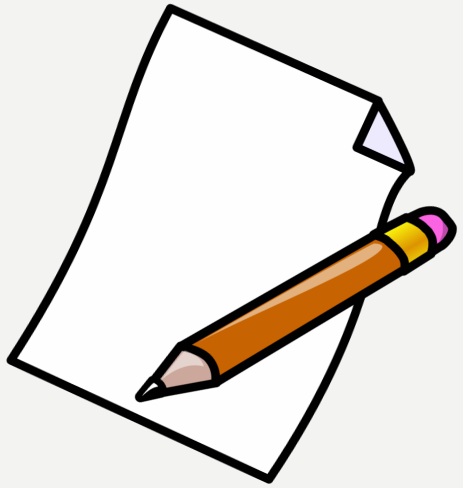
- More on readability
  - Indenting is a visual cue to say what code is "part of" other code.
  - This is not required in all programming languages but thankfully it is in Python, because Python forces you to indent your code
  - Indenting can aid readability tremendously

PROBLEM SOLVING

- There are two parts to our goal when solving problems with computer programs:
  1. Understand the problems to be solved
  2. Encode the solution in a programming language (This is the tricky part)

PROBLEM SOLVING

- It is very important that we impress on you *to try and understand how to solve the problem **<u>first</u>** before you try and code it.*
- *Have you forgotten your two best friends?*
  - *They can be very helpful when trying to figure out how to solve programming problems*



PROBLEM SOLVING

- Here are the steps to problem solving from the course's text book
  - Engage/Commit
  - Visualize / See (the pen and paper helps here ☺)
  - Try it/Experiment
  - Simplify
  - Analyze / Think
  - Relax
- This takes time! But the more you practice the easier it becomes
  - Don't give up easily, try taking a break instead and come back to the problem with a fresh mind

PROBLEM SOLVING

- Find a way that works for you,
  some way to make the problem tangible.
- Draw pictures
- Try to "see" the problem somehow

PROBLEM SOLVING

- Don‘t be afraid to try out your code
  - For some reason, it is quite common for people to be afraid to just ***try*** out some solution. Perhaps they fear failure, but experiments, done just for you, are a great way to learn.
- Don´t be afraid to fail, just keep on trying your solutions.
  - Get started, don’t wait for enlightenment!

PROBLEM SOLVING

- Simplify
  - Simplifying the problem so you can get a handle on it is a very powerful problem solving tool
  - Given a hard problem, make is ***simpler*** (smaller, clearer, easier), figure that out, then ramp up to the harder problem
  - An example:
    - Create a program that calculates the sum of a 100 numbers that the user inputs
    - Why not simplify this by creating a program that calculates the sum of 3 numbers that the user inputs?
    - Once you understand how to handle 3 numbers, expanding to more will be easier

PROBLEM SOLVING

- If your solution isn’t working:
  - Take a break
  - Analyze your code and what is actually going on in your code
- Remember!
  - Take your time. Not getting an answer right away is not the end of the world. Put it away and come back to it.
  - It can be surprising how easy it is to solve a programming problem if you let it go for a while. That’s why ***starting_early*** is a luxury you should afford yourself.

PROBLEM SOLVING

- Finally, some guidelines / rules from the text book
    1. Think before you program!
    2. A program is a human-readable essay on problem solving that also happens to execute on a computer.
    3. The best way to improve your programming and problem solving skills is to practice!
    4. A foolish consistency is the hobgoblin of little minds
    5. Test your code, often and thoroughly.
    6. If it was hard to write, it is probably hard to read. Add a comment.