Algorithms and Software Concepts
Introduction

Marc-Antoine Weisser

CentraleSupélec

September 12, 2024

Objectif du cours

Introduction to Python programming

1. Basics of Python: interpreter, script, `if`, `while`, sequences, `for`, list ...
2. Object and classes: inheritance, encapsulation
3. Advanced data type: dictionary, set, queue, file management

Organisation du cours

Cours et TD

- 3 course to discover the concepts
- 5 tutorial sessions to refine your skills

Feel free to interact and ask questions, both in tutorial sessions and lectures!

Ressources

All course materials are available at this address. In particular,

- the slides slides;
- a course booklet;
- all tutorials.

Algorithm & programming

Algorithm

Gérard Berry (professor at Collège de France, 1948-)

- "Un algorithme, c'est tout simplement une façon de décrire dans ses moindres détails comment procéder pour faire quelque chose. Il se trouve que beaucoup d'actions mécaniques, toutes probablement, se prêtent bien à une telle décortication. Le but est d'évacuer la pensée du calcul, afin de le rendre exécutable par une machine numérique (ordinateur)."
- "An algorithm is simply a way of describing in minute detail how to do something. It turns out that many mechanical actions, likely all of them, fit this definition. The goal is to remove thought from the calculation, making it executable by a computer."

Algorithm – Definition

Donald Knuth (Professor at Stanford, 1938–), "The Art of Computer Programming"

- An algorithm is a finite and **unambiguous** sequence of instructions and operations that allows solving a **class of problems**.
- Operations of an algorithm must all be sufficiently basic that they can in principale be done exactly by someone using pencil and paper
- An algorithm has zero or more **inputs**: quantities that are given initially before the algorithms begins, or dynamlically as the algorithms runs
- An algorithm has one or more **outputs**: quantities that have a specified relation to the inputs
- An algorithm is said to be **correct** when, for each instance of the problem, it terminates by producing the correct output.

Example – Primality Test

Algorithm 1: Determine if a given integer is prime or not

input : An integer $n > 1$

output: Return True if n is prime and False if not

for *all integer values i between 2 and* $\sqrt{n}$ **do**

    r is the remainder of the Euclidean division of n by i;

    if r *equals* 0 **then**

        return False;

return True;

Example – Fermat Test

Algorithm 2: Determine if a given integer is prime or not

input : An integer $n > 1$

output: Return True if n is probably prime and False if it is not prime

if 2^{n-1} *equals* 1 *mod* n **then**

    return True (n is probably prime);

else

    return False (n is definitely not prime);

Properties of algorithms

Various algorithms for a same problem

Each algorithm has different properties

- Running time
- Probabilistic or not
- ...

These properties are inherent to the algorithms themselves. They are not based on how they are implemented.

We will learn how to go from an algorithm to Python code.

What is Python?

- Python is a high-level, interpreted programming language.
- Developed by Guido van Rossum in the late 1980s.
- Main features
  - Easy to learn and read (clear syntax).
  - Interpreted language (no need to compile).
  - Dynamically typed (no need to declare variable types).
  - Supports multiple programming paradigms (object-oriented, procedural, functional).
  - Extensive standard libraries.