

# Computer Science Laboratory 01

---

## Themes

---

1. A first approach to computational thinking:
  - a. Computers and computer programs
  - b. Well-formulated problems and algorithms
  - c. Flowcharts and pseudocode
2. Python first steps:
  - a. Environment set-up
  - b. Creating and executing programs
  - c. Print to terminal

## Discussion

---

- A. Explain the difference between using a computer program and programming a computer.
- B. A toaster is a single-function device, but a computer can be programmed to carry out different tasks. Is your cell phone a single-function device, or is it a programmable computer? (Your answer will depend on your cell phone model.)
- C. What are the characteristics of an algorithm that needs to be understood and executed by a computer?
- D. What is the relationship between the starting problem, the pseudocode, the computer program, the flowchart, and the programming language?

## Exercises

---

### Part 1 – From problem to algorithm

**Task:** Assess whether the following problems are formulated in such a way that they can be managed by a computer program. If they are, write an algorithm (as pseudocode and/or flowchart) to solve them. Otherwise, describe why they cannot be managed by a computer program. Complete at least the first four exercises during the laboratory session and finish the remaining ones at home. It is not required to write any Python code.

- 01.1.1 Scheduling.** In a scheduling program, we want to check whether two appointments overlap. For simplicity, appointments start at a full hour, and we use military time (with hours 0–23). The following pseudocode describes an algorithm that determines whether the appointment with starting time **start1** and ending time **end1** overlaps with the appointment with starting time **start2** and ending time **end2**.

```

If start1 > start2
    s = start1
Else
    s = start2
If end1 < end2
    e = end1
Else
    e = end2
If s < e

```

*The appointments overlap.*

*Else*

*The appointments don't overlap.*

Check this algorithm with the following example values used as test cases: an appointment from 10–12 and the second one from 11–13.

Second example: an appointment from 10–11 and one from 12–13. [R3.12]

**01.1.2 Seasons.** The following algorithm yields the season (Spring, Summer, Fall, or Winter) for a given month and day.

*If month is 1, 2 or 3*  
                                   *season = "Winter"*

*Else, if month is 4, 5 or 6*  
                                   *season = "Spring"*

*Else, if month is 7, 8 or 9*  
                                   *season = "Summer"*

*Else, if month is 10, 11 or 12*  
                                   *season = "Fall"*

*If month can be divided by 3 and day is >= 21*  
                                   *If season is "Winter"*  
   *season = "Spring"*

*Else, if season is "Spring"*  
   *season = "Summer"*

*Else, if season is "Summer"*  
   *season = "Fall"*

*Else*  
                                   *season = "Winter"*

Draw the corresponding flowchart. Verify the correctness of the algorithm with a series of test date. [R3.13]

**01.1.3 On the road.** You want to find out which fraction of your car's use is for moving to work, and which is for personal use. You know the one-way distance from your home to your work place. For a particular period, you recorded the beginning and ending kilometers on the odometer and the number of working days. Write an algorithm to settle this question. [R1.16]

**01.1.4 Checkboard tiling.** Make a plan for tiling a rectangular bathroom floor with alternating black and white tiles measuring 10 × 10 cm. The floor is of size 50 x 30 cm [Worked Example 1.1]

**01.1.5 Future Decisions.** A student that is about to finish his/her high school and wants to choose which University apply to. They can base their decision on other people suggestions, and on the mean salary obtainable one year after graduation. Most of the faculties they are interested in offer statistics on the employment of grad students. How can they decide?

**01.1.6 Know when to stop.** The value of  $\pi$  can be computed according to the following formula:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

Write an algorithm to compute  $\pi$ . Because the formula is an infinite series and an algorithm must stop after a finite number of steps, you should stop when you have the result determined to six significant digits. [R1.18]

**01.1.7 Back to square 1.** Consider the following algorithms:

Perhaps the first **algorithm** used for approximating  $\sqrt{S}$  is known as the **Babylonian method**. It proceeds as follows:

1. Begin with an arbitrary positive starting value  $x_0$  (the closer to the actual square root of  $S$ , the better).
2. Let  $x_{n+1}$  be the average of  $x_n$  and  $S/x_n$ .
3. Repeat step 2 until the desired accuracy is achieved.

It can also be represented as:

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{S}{x_n} \right),$$

Use this algorithm to compute  $\sqrt{3}$ .

## Part 2 – From algorithms to code

**Task:** Debug the following program to understand how variables are assigned and computed inside a python program.

### 01.2.1 Copy and paste the following code inside PyCharm:

```
def compute_operating_cost(annual_km_driven, years, fuel_cost, km_per_liter):

    annual_fuel_consumed = annual_km_driven / km_per_liter
    annual_fuel_cost = fuel_cost * annual_fuel_consumed
    operating_cost = annual_fuel_cost * years

    return operating_cost

car_1_price = float(input("What is the price of car 1?"))
car_2_price = float(input("What is the price of car 2?"))

car_1_km_per_liter = float(input("How many km does car 1 do with a liter of fuel?"))
car_2_km_per_liter = float(input("How many km does car 2 do with a liter of fuel?"))

years = int(input("How many years will you drive this car? "))
km_per_year = int(input("On average, how many km will you drive each year? "))
fuel_cost = float(input("On average, how much dollars does it coast a liter of fuel? "))

annual_km_driven = years * km_per_year

car_1_operating_cost = compute_operating_cost(annual_km_driven, years,
                                              fuel_cost, car_1_km_per_liter)
car_2_operating_cost = compute_operating_cost(annual_km_driven, years,
                                              fuel_cost, car_2_km_per_liter)

car_1_total_cost = car_1_price + car_1_operating_cost
car_2_total_cost = car_2_price + car_2_operating_cost

if car_1_total_cost < car_2_total_cost:
    print("Car 1 is less expenisve overall")
else:
    print("Car 2 is less expensive overall")
```

### Part 3 – From algorithms to code

**Task:** For each of the following sub-task, write a python program that prints out the required output. If you have enough time, complete the exercises in the lab, otherwise complete them at home.

**01.3.1** A phrase to celebrate the beginning of Computer Science laboratories. [P1.1]

**01.3.2** Write a program that prints the sum of the first ten positive integers,  $1 + 2 + \dots + 10$ . [P1.2]

**01.3.3** Write a program that prints the product of the first ten positive integers,  $1 \times 2 \times \dots \times 10$ . (Use `*` to indicate multiplication in Python.) [P1.3]

**01.3.4** Write a program that prints your name in large letters, such as:

```

  AAA  SSSSSS  CCCCCC  IIII  IIII
AA AA  SS    SS CC    CC  II  II
AA  AA  SS    CC      II  II
AA  AA  SSSSSS CC      II  II
AAAAAAAAA  SS CC      II  II
AA  AA SS    SS CC    CC  II  II
AA  AA  SSSSSS  CCCCCC  IIII  IIII

```

[P1.6]

**01.3.5** Your name in a column.

**01.3.6** Write a program that prints the balance of an account after the first, second, and third year. The account has an initial balance of \$1,000 and earns 5 percent interest per year. [P1.4]

**01.3.7** Write a program that displays your name inside a box on the screen, like this:

```

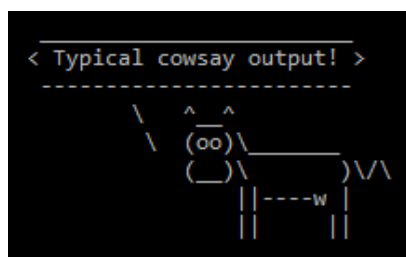
+-----+
|  Dave  |
+-----+

```

Do your best to approximate lines with characters such as `|` `-` `+`.

[P1.5]

**01.3.8** Write a program that prints an animal speaking a greeting, similar to (but different from) the following, without using the command `cowsay`:



```

< Typical cowsay output! >
-----
      \   ^__^
       (oo)\_____)
        (_____)  )\
           ||----w |
           ||     ||

```

[P1.10]

**01.3.9** An alphanumeric code of 16 characters alternating strings `"abcd"` and `"1234"`.

- 01.3.10** A checkboard of size 5x5 where the white squares are indicated by a "0", and the black squares by a "1".
- 01.3.11** O line of 100 dashes ("-").
- 01.3.12** A sequence of 100 zeros.
- 01.3.13** The fourth element of the Fibonacci sequence, where every element is the sum of the two preceding elements. The first two elements of the sequence are 0 and 1.
- 01.3.14** The first four element of the Fibonacci sequence in a column.
- 01.3.15** A closing phrase for this laboratory session, inside a box of your choosing, including the computation of the percentage of exercises you completed.