Zurich University
of Applied Sciences

School of
Engineering

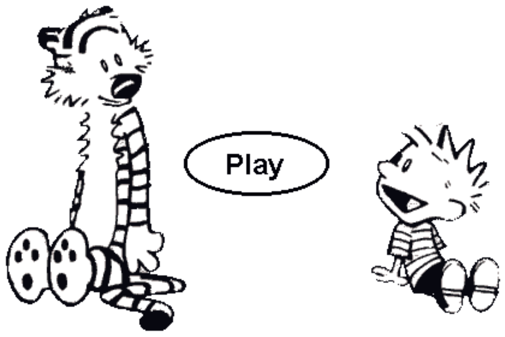
P03 – Loops

1. Loops

1.1 Countdown

Write a program that asks the user for a number and then counts down from the given number to zero. Every step should be printed in the console. Write both as a for-loop and as a while-loop.

1.2 Guess the Number



In this exercise you'll create a little game. Here is how to play:

- Your program will pick a secret number between 1 and 10.
- The user guesses what number it is.
- If the user's guess is too high or too low, the program will give the user a hint.

Hint:

- Loop as long as the secret number is not found.

Optional:

- Create a program that simulates 10'000 runs of the "Guess the Number" game (both players are now simulated by the computer) and compute the average number of guesses needed to find the secrete number.
- Before you run the simulation, think about what average number you would expect.

1.3 Average Age (optional)

Write a program that will calculate the average age of a group of people. First ask how many persons are in the group, and then ask for all the individual ages of the persons. Your program should calculate and display the average of those ages.

1.4 Compute Pi

The mathematician Srinivasa Ramanujan found an infinite series that can be used to generate a numerical approximation of π :

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \cdot \sum_{n=0}^{\infty} \frac{(4n)!}{(n!)^4} \cdot \frac{1103 + 26390n}{396^{4n}}$$

Write a script called `estimate_pi.py` that uses this formula to compute and return an estimate of π. It should use a while loop to compute terms of the summation until the last term is smaller than `1e-15` (which is Python notation for 10^{-15}). You can check the result by comparing it to `math.pi`.

Hint: Use existing code like `math.factorial(n)` to ease your solution.

2. Nested Loops

2.1 Domino

Write a program that creates all possible Domino stones (0|0), (0|1), …, (6|6). No duplicates are allowed. That means if there is a stone (2|6) than (6|2) is not allowed.

2.2 Draw a board

Write a program that outputs this 15 x 15 board on the console:

```
. . x x x x x x x x x x x x x
. . . x x x x x x x x x x x x
x . . . x x x x x x x x x x x
x x . . . x x x x x x x x x x
x x x . . . x x x x x x x x x
x x x x . . . x x x x x x x x
x x x x x . . . x x x x x x x
x x x x x x . . . x x x x x x
x x x x x x x . . . x x x x x
x x x x x x x x . . . x x x x
x x x x x x x x x . . . x x x
x x x x x x x x x x . . . x x
x x x x x x x x x x x . . . x
x x x x x x x x x x x x . . .
x x x x x x x x x x x x x . .
```

Use a nested loop to do this.

2.3 Extended Game of Life (optional → replaces up to 4 of the previous tasks)

“Game of Life” is a simple simulation of the “life” and “death” of cells on a grid that produces astonishing results. It has been developed by John Horton Conway in 1970 and is said to having consumed the majority of CPU cycles of all office computer of that era due to its vast distribution and viewership.

Take the minimal example from lecture V06 and extend it in several ways to appear more interesting to watch. For example, you could

- You could indicate the “age” of alive cells, e.g. by color
  → How could some notion of age be computed? When is its display “valuable” for an observer?
- You could add some “artificial evolution” that randomly switches the state of cells, thereby skipping the rules.
  → How much randomness is helpful to make the simulation more interesting to watch without disturbing the order of that universe too much?
- You could read the Wikipedia article to get inspired for more ideas:
  http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
- You could port the script to your favorite iDevice using e.g. Pythonista:
  http://omz-software.com/pythonista/

→ In any case, discuss all of your results and findings with your advisor before the end of the second week’s lab session. If you choose to work on task 2.3, ask your advisor in advance how much “points” you might earn with your approach.