Assignment 4: Programs and modules

Please remember to hand in the solutions in a single `.zip` file, where you give the answers for each part in separate files.

Exercise A: Debugging loops

Download the file `loops.py`, open it in Mu and use the debugger to understand the program. Answer the following questions:

- What is the difference between the variable called `ingredients` and the variable called `ingredient`?
- Why are there two `for` loops?

Finally, rewrite the code to do the same thing, but using one single `for` loop. In other words you need to replace line 13 and 14 with code that replaces the lines, but without using a loop.

- Hint: Do you remember how to convert a list of strings into a single string using `.join()`?

Exercise B: Save the turtles!

Turtle programs are cool because they can draw nifty things like rectangles. And we can even choose how big they should be! Even though this part is split in two, you can hand in a single `B.py` file.

Exercise B.1: Parameterising rectangles

Your task in this assignment is to draw rectangles, whose width and height are specified by the user. In other words, you will have to ask the user to input the width and height of the rectangle, and then draw it using turtles.

Hint: Before writing the program, write down the problem in pseudo code. How would you solve this in steps?

Exercise B.2: Crash the turtle!

Unfortunately, not all users are equally smart. Some users might not give you a number when you ask for it, and then what? Try to break your program by writing something like `'Anna, Viktor, Morten and Jana are awesome!'` instead of a number. What happens?

Likely you program doesn't survive that and crashes. We want to avoid that. Modify your program so you *catch* that error. Meaning, your program should not crash with an ugly exception. Instead, your program should write a neat error message to the user like: `'Please type a number'`.

Hint: If you don't know what we mean by *catching* an error, look at the slides about exceptions.

Exercise B.3: Save the turtle! (Bonus question: skip this if it takes too long)

If the user wrote gibberish instead of numbers your program now terminates in a better way than before, but it still terminates without completing its job. Can you modify the program so that you continue to ask the user for the width and height of your rectangle, until the user writes two meaningful numbers?

Hint: Do you know how to repeat some code, until you fulfill a certain condition (user typing two meaningful numbers)?

Exercise C: Understanding command line programs

Download the file `cli_program.py`, open it in Mu and describe what it does. Write a few lines of prose describing the program in general terms. Then answer the following questions:

- What does `sys.argv[1]` mean?
- What does `line.replace(...)` mean?

Exercise D: Creating a command line Python program

Your job is to write the code necessary for a program to take an argument from the command line and return its value increased by one.

Assuming your file is called `incrementor.py`, the goal is to make the following command give back the number `3`:

```
python incrementor.py 2
```

- Hint: Break this down into smaller steps. First, can you write a program that prints out the number you give it? (And **ONLY** prints it out, nothing else -- wait with the rest!). So if you write `python incrementor.py 9` can you make it print `9`? Next step: can you make it print out `10`?
- Hint: If you don't remember what an argument it, now's a good time to revisit the materials we prepared for you.

In [ ]:

```
1
```