# Practical 4. Variables, Booleans, and loops

MI Stefan

IBI1, 2018/19

## 1 Learning objectives

- Explain how variables work within a piece of code
- Explain the concept of loops
- Understand and use Booleans
- Plan a project using pseudocode
- Create and use variables in Python
- Create and use loops and Booleans in Python

## 2 Working with Spyder

- You have installed Spyder as part of your anaconda-navigator installation. Start up Spyder.
- Explore the Spyder window. There is a console in the lower right part. You can use this to type in python commands directly and see the output, for instance `1+1`
- The main window is for scripts. A python script is a collection of commands. You can write several lines of code and then run them all together by clicking "Run". If there is a result, it will be shown in the console. Before you run the script for the first time, you will have to save it to a .py file.
- You can also run individual lines by selecting them and then clicking Run → Run selection or current line
- Try it out: Type the following in a file called `helloworld.py`

```
a = "hello "
b = "world!"
print(a+b)
```

- Python files are not that different from text files. In particular, they can be put under version control using git and gitkraken. For more complex projects in particular, make sure you commit changes on a regular basis, so you can go back to earlier versions if you run into problems.

## 3 Working with variables

- Start a new file called `variables.py`

### 3.1 Some simple math

- Think of a 3-digit number (e.g. 123). Store it in a variable called `a`.
- Create another variable called `b` and assign to it a six-digit number, created by writing the digits of `a` twice (in this example, 123123).
- Can `b` be divided by 7 (giving an integer)? Use `%` to check! Create a new variable `c`. Assign it to be `b` divided by 7.
- Create a new variable `d`. Assign it to `c` divided by 11.
- Create a new variable called `e` that is `d` divided by 13.
- Compare `e` to `a`. Which of them is greater? Why?

### 3.2 Booleans

- You learned in lecture that “either X or Y” is the same as “(X and not Y) or (Y and not X)”. Make Boolean variables X and Y. Make a variable Z that encodes “(X and not Y) or (Y and not X)” and verify that it true if either X or Y (but not both) are true. Make a variable W that encodes Zhiwen’s more elegant solution (“X !=Y”) and verify that W and Z are always the same, no matter the values of X and Y.

## 4 Collatz sequence

- Start a new file called `collatz.py`
- For any positive integer $n$, the next number in the Collatz sequence can be obtained by either dividing by 2 (if $n$ is even) or multiplying by 3 and adding 1 (if $n$ is one). It seems that Collatz sequences always end with $4-2-1-4-2-1-4-2-1-\ldots$, no matter where they started.
- Write a script that starts with a positive integer `n` and computes and displays the Collatz sequence of `n` and ends when you reach 1 for the first time. (Python may decide to display some figure(s) after the comma, e.g. “25.0” instead of “25”. This is nothing to worry about at this stage.) Before you write the actual code, plan your project using pseudocode and comments (lines starting with `#`). When you write the actual code, leave the pseudocode comments in.

## 5 Mystery Code

- Look at file `mysterycode.py`
- What does it do? Run it a few times and formulate a hypothesis.
- Check whether your hypothesis is correct by going through the code line by line. Sometimes, it’s helpful to do this using pen and paper, but it is up to you. If you find out what a particular line does, use comments (lines starting with `#`) to note your thoughts. We have done this at the beginning of the document, e.g. in lines 4-6 to explain line 7.
- Once you have confirmed what the code does, write a one-sentence description next to `# Answer:` on line 2.

## 6 Powers of 2

- Start a new file called `powersof2.py`
- Every number can be written as a sum of powers of 2. For instance,
  $2019 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^1 + 2^0$
- Write some code that starts with some number `x` (e.g. `x=2019`) and computes the powers of 2 that make up `x` (for simplicity, you can assume that $x$ is no larger than $8192 = 2^{13}$)

  The output should be a sentence on the screen that gives the binary composition of `x`. For instance,

  ```
  2019 is 2**10 + 2**9 + 2**8 + 2**7 + 2**6 + 2**5 + 2**1 + 2**0
  ```
- Before you write the actual code, plan your project using pseudocode and comments (lines starting with `#`). When you write the actual code, leave the pseudocode comments in.
- You may find the command `str()` helpful: It converts a number into a string. For instance, `str(5)` will give `"5"`

## 7 For your portfolio

The markers will look for and assess the following:

**File variables.py**

- The marker will look at your variables $a$ and $e$, confirm that $e$ has been created in the way that was specified in the instructions, and compare $a$ to $e$.
- The marker will test all possible values of $X$ and $Y$ and verify that $W$ and $Z$ behave as they should.

**File collatz.py**

- The marker will set n to some number and check that the code runs, terminates, and that the last entries are consistent with the Collatz sequence.
- The marker will verify that you have used pseudocode to plan and comment your project.

**File mysterycode.py**

- The marker will look at the answer and check whether it is correct.

**File powersof2.py**

- The marker will verify that you have used pseudocode to plan and comment your project
- The marker will test your script using the number `x=1750`
- Partial grades will be given if the project is incomplete.

You can add or edit things after the Practical session. We do not look at the commit date, we just want it all to be there!