Exercise Session n. 1

Algorithms and Data Structures

We start with a couple of exercises with variables and simple expressions. We then turn sequences of instructions into functions. We then work with conditional instructions.

Rotation

Assume that you have variables x, y that represent the Cartesian coordinates of a point P. Write Python instructions that move P by rotating the plane by 90 degrees counter-clockwise.

Examples

```
>>> x = 1
>>> y = 1
>>> # <<< your instructions go here...
>>> x
-1
>>> y
1
```

Simple, Linear Transformation

Assume that you have variables x, y that hold numeric values. For example, $x = 7, y = 2$. Write Python instructions that update x and y according to this linear map (you'll learn all about this kind of operations in Linear Algebra):

$$x' = 2x$$
$$y' = x + y$$

Notice that, in these equations, x' and y' represent the updated values of x and y, respectively. However, there are only two variables. So, in fact, x' and y' represent the values of x and y *after* the update operation.

Examples

```
>>> x = 7
>>> y = 2
>>> # <<< your instructions go here...
>>> x
14
>>> y
9
```

Half Adder with Simple Arithmetic Expressions

Write the instructions that, given two bits a, b, meaning two variables a and b each holding a binary value 0 or 1, compute two binary values s and c representing the *sum* and the *carry* values of $a + b$, respectively. Do this using only simple arithmetic expressions with addition, subtraction, and multiplication. You are not allowed to use other operations or conditional instructions.

Examples

```
>>> a = 0
>>> b = 1
>>> # <<< your instructions go here...
>>> s
1
>>> c
0
>>> a = 1
>>> b = 1
>>> # <<< your instructions go here...
>>> s
0
>>> c
1
```

Defining Functions

Write the instructions of the previous exercises as three functions `rotate90(x,y)`, `linear_map(x,y)` and `half_adder(a,b)`, respectively.

Examples

```
>>> half_adder(1,0)
(1,0)
```

Hint: in Python, you can return a pair of values, or more generally a tuple, meaning a list of values. You can do that by simply returning the values separated by commas. For example, the instruction `return 2,3` returns the pair $(2,3)$.

Adder with Conditional Instructions

Write a variant of the `half_adder(a,b)` function—that, given two bits a, b returns a “tuple” (s, c) where s and c are the *sum* and the *carry* values of $a + b$, respectively. Do this using only simple conditional instructions and no arithmetic expressions at all.

Full Adder with Conditional Instructions

Write a function `full_adder(a,b,c)` that, given three bits a, b, c, returns a pair of binary values (s, c) representing the *sum* and the *carry* values of $a + b + c$, respectively. Do this using only simple conditional instructions and no arithmetic expressions.

Examples

```
>>> full_adder(1,0,1)
(0,1)
```