**Lab 8: Decimal and binaries**

**Submission:**

This lab has to be submitted on Week 7, together with all other labs published before Week 7. You will have to explain your code, so make sure it is well documented and you understand it.

**Objective:**

The objective is to practice the use of if-elif-else, while, strings and string slicing applied in a real problem.

**The problem:**

We are going to look at some conversions from integer to binary and then from binary back to integer. Prompt the user for an integer, convert the integer to a binary number string (there is no type for actual binary numbers so we just represent it as a string). If you are not familiar with binary numbers have a look on this link <https://www.mathsisfun.com/binary-number-system.html>

In the second part of the exercise we’ll take the binary string representation and we will convert it back into an integer.

Things to remember:

1. If the integer is 0, then we are done since conversion back and forth of 0 is still 0. The program simply prints a note saying it is 0 and quits.
2. If the integer is negative, then we probably don’t know how to do it, so the program prints a message saying it is negative and quits.
3. Otherwise, we do the conversion of the integer to a binary string (a string of 1s and 0s) and then convert that same string back to an integer to make sure we did it right.

***Hints***

**How do we get a binary string from an integer?**

To convert an integer to binary, start with the integer in question and divide it by 2 keeping notice of the quotient and the remainder. Continue dividing the quotient by 2 until you get a quotient of zero. Then just write out the remainders in the reverse order.

Here is an example of such conversion using the integer 12.

First, let’s divide the number by two specifying quotient and remainder:

12 / 2 = 6, remainder 0

6 / 2 = 3, remainder 0

3 / 2 = 1, remainder 1

1 / 2 = 0, remainder 1

Now, we simply need to write out the remainder in the reverse order — ‘1100’. So, 12 in the decimal system is represented as ‘1100’ in binary.

**How do we get an integer from a binary string?**

We know it is a string, so the elements are ‘1’ and ‘0’. Every time we grab a 1 or a 0 (a bit), we are adding a power of two to the overall integer value. Which power of 2? If you grab bits from the right, they are increasing orders of powers of 2. The far right position of the string, or, better said, the last bit in the string (how do you get the last bit??) is 2\*\*0. The next bit 2\*\*1. The next bit 2\*\*2. And so on. If the bit is a ‘1’, then we add that power of 2 to the overall sum; if it is 0 we do nothing.

For example, start with ‘1100’

* + last bit is ‘0’ and 2\*\*0 \* 0 is 0. Sum is 0
  + next bit is ‘0’ and 2\*\*1 \* 1 is 2. Sum is 0
  + next bit is ‘1’ and 2\*\*2 \* 1 is 4. Sum is 4
  + next bit (first bit) is ‘1’ and 2\*\*3 \* 1 is 8. Sum is 12
  + Or ‘1100’ = int(‘0’) \* 2 \*\* 0 + int(‘0’) \* 2 \*\* 1 + int(‘1’) \* 2 \*\* 2 + int(‘1’) \*\* 2^3 = 12

**Implement the algorithms**

Create two files “decimalToBinary.py” and “binaryToDecimal.py” with the solutions of the two problem. On the first file ask the user for a decimal number and return its equivalent in binary. On the second file ask for a binary number and return the equivalent in binary.