Python exercises for Chapter 1

1. Write a function, `de2bi_a.py`, to convert the integer part of a base 10 number into a binary number. Use it for `x = 105.8125`.

   Hint: The following functions may be useful

   - `np.fix(x)`: rounds `x` towards `0`.
   - `a//b` : gives the quotient of the division.
   - `a%b` : gives the remainder of the division.
   - `np.flipud`: inverts the order of a matrix array up-down.

Solution:
```
[ 1 1 0 1 0 0 1]
```

2. Write a function, `de2bi_b.py`, to convert the fractional part of a base 10 number into a binary number. Use it for `x = 105.8125`.

   Hint: Take care avoiding infinite loops. Terminate if the number of binary digits is higher than some threshold, e.g. 64.

Solution:
```
[ 1 1 0 1]
```

3. Using the functions of Exercises 1 and 2, compute the IEEE 754 simple precision float point representation of 120.875 (check with Exercise 4 of the Exercises for Chapter 1).

Solution:
```
[0 1 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

4. (a) The largest double precission normalized number that Python may store in binary representation is

$$(+1) \cdot (1.11\ldots 11) \cdot 2^{1023}.$$

   Since we do not have to keep the first 1, there are 52 bits left to store the $1's$ of $0.11\ldots 11$. Therefore, in base 10 this number is

$$(1 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + \cdots + 1 \cdot 2^{-52}) \cdot 2^{1023}$$

   Write a code to compute this sum. Its value should coincide with that obtained with `sys.float_info.max`.

   (b) Write a code to compute the lowest representable normalized floating point number using the expression

$$(+1) \cdot (1.00\ldots 00) \cdot 2^{-1022}$$

   Its value should coincide with that obtained from `sys.float_info.min`.

   Use the following format for printing: `print('{:.16e}'.format(output))`, where `output` is your result.

Solution:
1.7976931348623157e+308
2.2250738585072014e-308

5. Compute the solution of $x^2 + 10^8 x + 1 = 0$ using the well known formula

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

and find its (very large) residual. Find another formula to compute x_1 with a lower residual.

Use the following format for printing: `print('{:.6e}'.format(output))`, where `output` is your result.

Hint: Multiply by the conjugate of x_1.

Solution:
2.549419e-01
1.110223e-16

6. The distance between `x` and the nearest adjacent number is given by `numpy.spacing(x)`. Knowing that

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!},$$

write a code to compute the number of terms we need to get $\cos 30^o$ with the lowest possible error.

Use the following formats for printing: `print('{:1d}'.format(n))` for the number of terms, and `print('{:.6e}'.format(output))`, for the value.

Note: Arguments of trigonometric functions must be given in radians.

Solution:
7
8.660254e-01