Exercise Set 5

More Python

University of Oslo - INF3331/INF43331

Fall 2018

Note: These exercises are not mandatory. You don't have to put your work into your Github repository.

Exercise 1: Numba

Speed up the following function as much as possible using Numba

```
from decimal import Decimal

def test(arr):
    for i in range(len(arr)):
        arr[i] = i % Decimal(100)
```

Run the code with `nopython=True` in your call to `numba.jit`. The code will fail because it has to fall back on Python runtime.

Exercise 2: Generators

Python allows you to write generators. Put simply, these are functions that act like iterators. In Python 3, `range` is an iterator. Instead of generating every number we need, allocating memory for a list containing them, range will not generate the next number until we ask for it.

A simple implementation of `range` can look something like this:

```
def range(n):
    i = 0
    while i < n:
        yield i
        i += 1
```

The `yield` keyword behaves a lot like `return`, except that it sort of "pauses" the function, and continues the computations when we ask it for the next number with `next()` or get to the next iteration in e.g. a for loop. This also allows us to write generators that never stop. Try to write a generator that yields every fibonacci number, i.e. 1, 1, 2, 3, 5, 8, 13, 21...

Familiarize yourself with some of the functions in the module `itertools`. Use one of the functions to get every fibonacci number below 1000, by using your fibonacci generator.