Assignment 3

Due: 23th February.

1 Python Performance

(*20 Points*) Go to
https://wiki.python.org/moin/PythonSpeed/PerformanceTips
and based on that documentation

- (*5 Points*) speed up a loop by inventing your own example, following the section "Loops"
- (*5 Points*) speed up a code where data aggregates, following the section "Data Aggregation"
- (*10 Points*) Write your own examples where Python is compared with C, something like what is shown in the section "Python is not C".

2 Python Decorators

(*30 Points*) As we discussed in class, the symbol @ is Python decorator syntax. Python decorators are normally used for tracing, locking, or logging. Here we will study an example.

The following function computes the ith Fibonacci number for a given value of i.

```
def fib(i):

    if i < 2:
    return 1 return fib(i-1) + fib(i-2)
```

Using the following code, we can create a decorator that saves each intermediate value in memory rather than calculating it every time.

```
from functools import wraps
def cache(f):

    cache = { }
    @wraps(f)
    def wrap(*arg):
        if arg not in cache: cache[arg] = f(*arg)
        return cache[arg]
    return wrap
```

- (*5 Points*) What is @wraps and what is it doing?
- (*20 Points*) Using the magic[1] function *%timeit* time how long it takes to find *fib(20)*, *fib(25)*, *fib(30)*, *fib(35)* and plot the results.
- (*5 Points*) Now time how long the same *fib* function takes if it is decorated with @cache. Explain what is happening?

[1]You can read the documentation of the time execution of a Python statement or expression at https://ipython.readthedocs.io/en/stable/interactive/magics.html