Abstract of **"Numba library for Python programs acceleration"** presentation at the programming course seminar
(original presentation in Russian)

Artem Sevastopolsky, 2015

Presentation *"Numba library for Python programs acceleration"* was prepared at the seminar of programming course. The presentation was prepared in short, 10-minute style. Author wished to pursuit two goals: first, to give complete overview of the only one (but the most important) tool of Numba library – `@jit` decorator, and, second, to encourage listener to use this library when it is appropriate.

In the first part it is explained why programs written in Python language execute generally slower than programs written in some well-known compiled languages, such as C/C++. Number of factors that produce such effect are enumerated, such as dynamic typing, automatic type inference, multiple accesses by pointer. The explanation is also supported by example where main low-level operations made by Python interpreter when it is needed to simply sum two numbers are enlisted. Then, task of two NumPy arrays[1] addition is examined. It is shown how to use `@jit` decorator of Numba library with function that adds up two NumPy arrays, and how can it produce acceleration of about 100 times.

In the second part, it is shown what acceleration can Numba produce for programs with `for` cycles. Problem of computing pairwise distance matrix between points three different ways is examined: pure Python solution (contains 3 `for` cycles), solution that makes use of NumPy library (contains 2 `for` cycles over points) and solution using appropriate highly optimimzed function from SciPy[2] library. Each solution are compared with each other and their JIT-compiled modifications. Experimental results are provided which show that either JIT-compiled second (Python+NumPy+JIT) solution or third (SciPy) solution is the fastest, so it is possible to write Python code that executes at the times comparable to the precompiled code. It is also shown that JIT-compilation using Numba library can speed pure Python solution up to 460 times.

In the end of the presentation, it is briefly explained how Numba works and how to use`@jit` decorator in a few different ways. `@cuda.jit` decorator for efficient GPU computations is also presented and link with detailed analysis of this decorator is given.

[1] *Numpy* is a library for Python for scientific computing; it has its own efficient array type. `numpy.org`
[2] SciPy is a library that offers efficient numerical routines. `http://www.scipy.org/scipylib/index.html`