Computer Modeling of Physical Phenomena
Short Introduction

Jakub Tworzydło

Institut of Theoretical Physics
Jakub.Tworzydlo@fuw.edu.pl

28/02 and 01/03/2023 ul. Pasteura, Warszawa

Plan

1. Course organization
2. Introduction: Python

Plan

1. Course organization

2. Introduction: Python

Plan

1. Course organization
2. Introduction: Python

Earning credits

- we start at 2.15PM on Tuesday
- lecture contains inroduction to the lab
- during the lab you can earn 1.0 point; one week later 0.8 points
- written exam/test 3.2 points (on 13.06)
- one short (5min) presentation is mandatory; final presentations on 13.06

Scheme of our meetings

- review/seminar like lecture
- presentation: results of the previous lab
- practical intro to the next lab
- 1 or 2 tasks to complete on your own
- extra take-home task (0.2 bonus points)

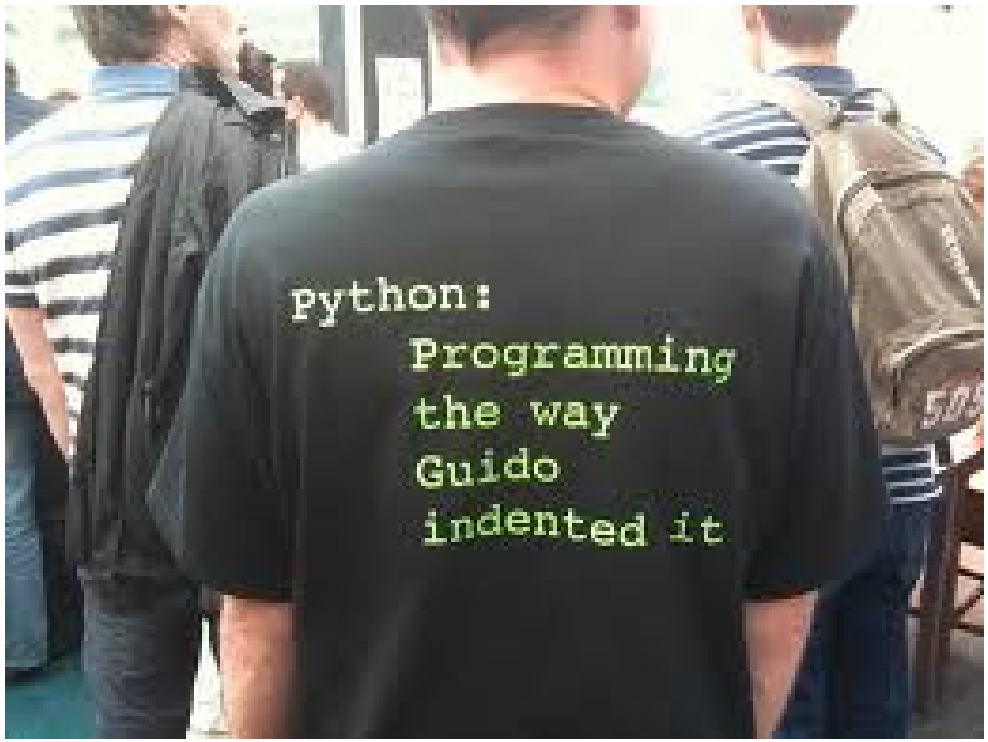
Plan

1 Course organization

2 Introduction: Python

We use Python

- interactive and interpreted (scripting language)
- is a real high-level language
- supports many data structures
- paradigms: procedural, object-oriented, functional



Python features

- batteries included philosophy: don't reinvent the wheel
- easy to learn, clear syntax, readable code
- portable: can ran on many platforms
- expandable (one can add modules from C, C++ or Fortran)

More on features

- **Batteries included** Rich collection of already existing **bricks** of classic numerical methods, plotting or data processing tools. We don't want to re-program the plotting of a curve, a Fourier transform or a fitting algorithm. Don't reinvent the wheel!
- **Easy to learn** Most scientists are not payed as programmers, neither have they been trained so. They need to be able to draw a curve, smooth a signal, do a Fourier transform in a few minutes.
- **Easy communication** To keep code alive within a lab or a company it should be as readable as a book by collaborators, students, or maybe customers. Python syntax is simple, avoiding strange symbols or lengthy routine specifications that would divert the reader from mathematical or scientific understanding of the code.
- **Efficient code** Python numerical modules are computationally efficient. But needless to say that a very fast code becomes useless if too much time is spent writing it. Python aims for quick development times and quick execution times.
- **Universal** Python is a language used for many different problems. Learning Python avoids learning a new software for each new problem.

Learning curve

- “A Byte of Python” by C.H.Swaroop
  good point to start (free online)
- “Python Tutorial 3.6” by Guido
  a classic, but too much detailed
- “Dive into Python”
  good for consulting when we know something
  (free online)
- “Learning Python” by M. Lutz
  long text if one prefers a slower approach (our library)
- “Python Crash Course”
  a very no-nonsense book

Just what we need

- Matplotlib tutorial – a way to control the plot
  http://www.loria.fr/~rougier/teaching/matplotlib
- Numpy tutorial – tricks with arrays
  http://www.loria.fr/~rougier/teaching/numpy/
- Comprehensive scientific Python
  http://scipy-lectures.github.io/

Running Python

Different ways to use Python:

- interactive interpreter from command line
- running a script
  `$ python3 script.py`
- from a graphical user interface (GUI) environment;
  we recommend `IDLE`, some prefer `Spyder`,
  `PyCharm` is popular on mimuw
  see also `http://wiki.python.org/moin/IntegratedDevelopmentEnvironments`
- new trend: using Jupyter notebook running on a server
  (or Google Colab)