Computing Methods for Experimental Physics and Data Analysis

Introduction

L. Baldini, L. Bianchini, G. Lamanna, A. Retico, A. Rizzi

luca.baldini@pi.infn.it

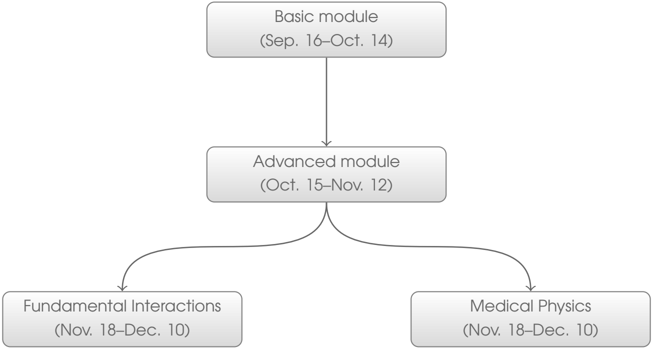
Università and INFN–Pisa

September 16, 2024

Goals and prerequisites

- What is this all about?
  - Automating repetitive tasks
  - Python basics, standard library and scientific ecosystem
  - Collaborative code development and best practices
  - Algorithms and data structures
  - Machine learning
  - Specific tools for high-energy physics or medical physics
- This is not so much about Python or C++—it is about how to write code for effective data analysis
- Will I be a professional data scientist at the end of the semester?
  - No, but hopefully you'll be able to poke around and find the right tool for the job at hand
- Pre-requisites
  - Have a vague idea of how a computer operates
  - If you have ever programmed before that would be great!

Basic structure of the course



- Modularity and standard paths:
  - Each module is worth 3 credits
  - 6 credits: basic + advanced
  - 9 credits: basic + advanced + fundamental interactions
  - 9 credits: basic + advanced + medical physics

- Collaborative tools
  - Version control, development workflow, development platforms
- Python basics
  - Coding conventions, structuring a package
  - Variables, native types, functions
  - The Python standard library
- Algorithms and data structures
  - Complexity and asymptotic running time
  - Python data structures and native algorithms
- Object-Oriented Programming (OOP)
  - Classes, inheritance, composition
  - Operator overload and emulation of Python builtin types
- The Python computing ecosystem
  - numpy: arrays, functions, broadasting
  - Vectorization
  - Scipy: plotting and fitting
  - Pandas

Advanced module

G. Lamanna, A. Rizzi

- Advanced code development
  - Unit testing, continuous integration, static analysis, documentation
- Advanced Python
  - Errors, exceptions, iterators and generators, decorators
  - Profiling and optimization
- Parallel computing
  - Computer architectures, memory, scaling laws, CPUs and GPUs
  - Parallel programming: concurrency and parallelism, threading in Python
- Machine learning
  - Classification and regression: boosted decision trees and multilayer perceptrons
  - Deep learning: neural networks, the keras library
  - Supervised and unsupervised training, reinforcement learning
  - Tensorflow

- Introduction to C++
  - Coding style and organization, declaration of interfaces
  - Classes: constructors, virtual functions, private and public, abstract classes, inheritance
  - References, pointers, dynamic memory allocation, memory ownership, smart pointers
  - Templates, standard template library
  - C++11 and C++14: lambda functions, auto variables
- More parallel computing
  - Cuda and OpenCL
  - Examples of algorithms for HEP
  - GPU in HEP Data Analysis
- The ROOT data analysis framework
  - ROOT toolkit
  - PyROOT, root-numpy, RDataFrame

- Medical data processing and feature extraction (python/MATLAB)
  - Tools for handling standard-format medical data (DICOM)
  - Data anonymization and visualization
  - Deriving features form images, image segmentation
  - Data quality control pipelines: outlier removal, dimensionality reduction
- Data analysis and classification (python/MATLAB)
  - Performance evaluations: figures of merit, cross-validation schemes, permutation test
  - Machine-learning and deep-learning tools for segmentation and classification
  - Data augmentation, transfer learning, retrieving localization information.

Logistics

Timetable and final exam

- Timetable: 5 + 1 hours a week
  - Monday, 16:30–19:30 (room A1)
    - If everybody agrees: start at 16:30 sharp(-ish), one 15-minutes break, try and be done by 7:00;
    - Last hour (18:30–19:30) typically for practical applications; we might skip it on specific weeks;
  - Tuesday, 08:30–11:30 (room M-Lab)
- Final exam
  - Development of a specific, reasonable-sized software project
    - Related to the topics covered in the course
    - We have a list of suggestions, but encourage everybody to come up with original projects—if you do so reach out to us well in advance to make sure the project is appropriate
    - Projects can be done individually or in pairs
  - Two-page description of the project and source code made available $\sim$ 1 week in advance
    - We expect a well-structure repository
    - Under no circumstance you should send code by email
  - Oral exam starts with a presentation of the project
    - Aim at 10 slides for 15–20 minutes
  - A few questions on the course material from a pre-compiled list
    - We expect the answers to the questions to be thorough and in-depth
- List of projects/questions to be updated over the next two weeks