Academic year 2021-22
PhD in Statistics and Computer Science, Bocconi University

# 40973 COMPUTER SCIENCE I (PROGRAMMING) - Syllabus

**Instructor:** Carlo Baldassi

**Prerequisites:** Operational familiarity with the basics of programming, in some main-stream language. Examples of languages: C, C++, C#, Python, MATLAB, R, Java, Javascript, Julia. Examples of basic concepts required: assignment, types, operators, syntactic rules, conditionals and loops, functions, recursion, composite types (e.g. C structs, Python classes).

**Course description:** This will be an intermediate-to-advanced level course, mainly aimed to scientific programming, numerical analysis, simulations, etc. We will use the Julia programming language, which is both i) particularly suitable for these kinds of tasks since and ii) especially rich in terms of available constructs, programming paradigms, coding patterns, both low- and high-level. No prerequisite knowledge about Julia is assumed, the language will be taught during the course. The course is very practice-oriented and hopefully highly interactive: after the first few lectures, students are expected to start working on projects (either that they care about or suggested by the instructor). Whenever possible, the students' projects will be the source of discussions, examples of optimization opportunities, design principles and approaches, etc.

**Course objectives:** The main aim of the course is to create a common programming background for students of the PhD course that will be applicable throughout the rest of their studies. Students should end up with a sufficient understanding of how computers operate under the hood to allow them to associate each line of code with its corresponding low-level operations. Whenever confronted with a computational task, students should be able, after the course, to write good, clean, reasonably efficient code, knowing what programming tools are available and how to make use of them. Whenever students encounter more sophisticated concepts in their future career, be it from theoretical Computer Science or from Statistics, they should be able to implement them in practice and, if needed, optimize them to work efficiently and make the most effective use of the hardware.

**Course topics (tentative):** The topics discussed during the course will be decided interactively based on the students' backgrounds and interests. Here is a non-comprehensive list of topics that may be good candidates:

- Hardware and software; levels of abstraction; machine code; memory layout
- Processes; stack and heap; threads
- Compiled vs interpreted languages and everything in between; IR code
- Basics of the git version-control software
- Basics of Julia: basic syntax elements; just-in-time compilation; workflow at the interactive REPL or in a notebook; standard library basics
- Peculiar features of Julia: type hierarchy and multiple dispatch; separation of data and interface; broadcasting
- Advanced features of Julia: parametric types; invariant, covariant, contravariant types; Union and TypeVar; dispatch tricks (e.g. Holy traits)
- Code optimization via profiling and fixing computational bottlenecks; demos of optimization workflow
- Specialized types; examples from LinearAlgebra special matrix types and factorization types
- Parallel computing: SIMD; multiprocessing; multithreading; multiple stacks; tasks and channels; GPUs
- Automatic differentiation for machine learning: forward mode, backward mode, source-to-source

**Course materials:** Code written will be shared on a private Gitlab instance; there is no textbook for this course, resources will be indicated along the way depending on the topics

**Final exam:** Each student or pair of students will propose their own project; if approved, they will submit it within a specified deadline. Projects will include a descriptive part, detailing what problem they aim to solve and how, and a code part (including documentation and demos).