UNIVERSITÀ
DEGLI STUDI DI MILANO-BICOCCA

COURSE SYLLABUS

Programming Languages

2627-2-E3102Q109

Aims

Knowledge and understanding
Students will acquire a solid understanding of the main programming paradigms, with particular focus on the logic and functional paradigms. They will understand the conceptual differences between these paradigms, as well as the theoretical and historical motivations behind their development, including aspects related to typing, problem modeling, and control flow.
Applying knowledge and understanding
Students will be able to apply their knowledge to the development of software using representative languages of the studied paradigms (e.g., Common Lisp for the functional paradigm, Prolog for the logic paradigm). They will use appropriate programming environments to design and implement solutions to problems of low to medium complexity, also in different application contexts.
Making judgements
Students will develop the ability to critically evaluate different programming paradigms in relation to the problem at hand, and to make informed decisions about the most suitable language and approach. They will be able to analyze and compare alternative programming strategies based on both theoretical and practical considerations.
Communication skills
Students will be able to clearly and accurately describe software solutions and design decisions, both orally and in writing, using appropriate technical terminology. They will also be able to explain and justify their choice of paradigm and language based on the characteristics of the problem.
Learning skills
The course will provide students with the theoretical and practical tools needed to independently explore new programming languages and paradigms. In particular, they will be equipped to critically and autonomously approach the learning of languages not explicitly covered in the course but belonging to the paradigms studied.

Contents

The course will provide students with a panoramic view of the main programming language paradigms: imperative, logic (declarative) and functional.

Detailed program

1. Programming languages paradigms: imperative, logical (declarative) and functional. Notions recall of “run-time”, program execution on an idealized stack architecture. Basic notions about programming environments for the different systems presented. Some historical perspectives.
2. The logic programming paradigm. Introduction to the Prolog programming language.
3. The functional programming paradigm. Introduction to the (Common) Lisp programming language. Notes on Haskell, Julia and Javascript.
4. Usage of the different paradigms in a variety of contexts: type checking, abstraction, abstract data types, concurrency and event handling.

Prerequisites

Basic concepts of recursive programming, mathematical logic and hardware/software architecture.

Teaching form

The course will be offered as a set of standard lectures during the term. “e-Learning” support will also be provided for the distribution of course material, exercises and project descriptions. Moreover there will be laboratory sessions where the students will learn to program with the programming languages introduced during the course.

As a possible guideline, subject to contingency, the course will be organized as

- 48 hours lectures: traditional lecture in presence.
- 8 3 hours laboratory: traditional and interactive laboratory activities in presence.

Textbook and teaching resource

Reference text:
Maurizio Gabbrielli, Simone Martini (2/ED): LINGUAGGI DI PROGRAMMAZIONE: Principi e Paradigmi", McGraw Hill, ISBN 8838665737 · 9788838665738

Suggested texts:

- Robert W. Sebesta (12th Edition). Concepts of Programming Languages. Pearson, 2025
- Leon Sterling and Ehud Shapiro, The Art of Prolog Advanced Programming Techniques - 2nd Edition;
- Abelson, Sussman e Sussman, Structure and Interpretation of Computer Programs (SICP);
- Abelson, Sussman, Henz e Wrigstad, Structure and Interpretation of Computer Programs (SICP) - Javascript edition;
- Peter Seibel, Practical Common Lisp (PCL);

- Brian W. Kernigham & Dennis M. Ritchie, C Programming Language (2?? Edition), Prentice Hall, 1988;
- Robert Sedgewick, Algorithms in C, Parts 1-5 (Bundle): Fundamentals, Data Structures, Sorting, Searching, and Graph Algorithms (3rd Edition), Addison Wesley, 2001
- Various online resources for the other programming language presented.

Semester

First semester

Assessment method

The Programming Languages exam requires passing a written test and a programming project. At the discretion of the instructors an oral exam may be also required. The written exam and the project corresponding to the first roll-call are organized into two "partial" tests.

The partial tests are two written exams offered in November (Logic Programming) and in January/February (Functional Programming; Imperative Programming), plus one or two projects, posted towards the end of October, to be electronically handed-in in January/February. The partial tests, as already said, comprise the first roll-call for the course and are not held over for the remaining roll-calls.

From the next roll-call (February/March) onward, the written exam will comprise all the course materiale. The same applies for the projects. The following roll-calls will be in June, July and September (in dates to be announced).

The written exam and the projects must be passed after the same roll-call; the final grade is the weighted average, at the discretion of the instructors, of the partial grades. In particular, a weighted arithmetic average equal to 18 does not guarantee the passing of the exam.

Projects can be done in groups (max 3 people). Whoever works on a project alone or in a pair, may have a bonus.

The final grade is in thirtieths, and expresses a comprehensive evaluation of everything that concurs to the achievement of the stated educational goals. That is, it is the result of a comprehensive evaluation of the tests. E.g., clarity, rigor, judgment abilities, argumentation choices and ability to illustrate them in an efficacious way.

Final note: every exam result will be recorded in the system: grade, withdraw, absent, insufficient, etc.

Office hours

Prof. Marco Antoniotti: by appointment via email.

Prof.Claudio Ferretti: by appointment via email.

Prof. Fabio Sartori: by appointment via email.

Sustainable Development Goals