

Teacher(s)	Mens Kim ;
Language :	English > French-friendly
Place of the course	Louvain-la-Neuve
Prerequisites	Required#: concepts, paradigms, and semantics of programming languages as targeted in course LINFO1104
Main themes	In the course of a career, a computer scientist or software engineer will be confronted with many different programming languages and paradigms. To make informed design choices when selecting a particular language, he or she must understand the principles underlying how programming language features are defined, implemented and used. This course will examine, from a historical perspective, the guiding principles of the major programming paradigms, starting from the earliest programming languages until the most recent ones. As such it will highlight the major principles, strengths and differences of different programming languages and paradigms.
Learning outcomes	At the end of this learning unit, the student is able to : Given the learning outcomes of the "Master in Computer Science and Engineering" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes: • INFO1.2 • INFO2.4-5 • INFO6.4-5 Given the learning outcomes of the "Master [120] in Computer Science" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes: • SINF5.2, SINF5.4-5 • SINF6.5 Students completing this course successfully will be able to: • describe and differentiate the main programming paradigms (including procedural programming, functional programming, logic programming, object-oriented programming, concurrent programming, as well as more recent programming paradigms); • determine what paradigm a programming language belongs to; • identify and discuss the design principles of a given language or paradigm; • choose a language or paradigm suitable for solving a particular problem and argue this choice; • write small programs in a selection of the different languages and paradigms seen in the course; • place a programming language in relation to others from a historical perspective; • compare different programming languages from the point of view of their underlying design principles; • understand the impact of different language design choices (syntax, parameter passing, scoping, abstraction, ').
Evaluation methods	Throughout the year, in parallel with the theory and lab sessions, the students will work in pairs to study in detail several of the paradigms seen in the course, by carrying out three programming missions in three different languages representative of those paradigms. These missions will be evaluated through code reviewing by, interviews with, and presentations to the professor and course assistant(s). This evaluation replaces the traditional course exam. Each of the three mission will count for approximatively one third of the points. Student who fail the course will need to redo in second session individually all programming missions they failed.
Teaching methods	The course will consist of traditional theory sessions in which the characteristics and guiding principles of different programming languages and paradigms are explored in detail. The practical sessions complement these more theoretical course sessions with hands-on programming exercises in a selection of programming languages and paradigms seen in the theory course.
Content	Students completing this course successfully will be able to: • describe and differentiate the programming paradigms addressed in the course (e.g., functional programming, logic programming, reflection and metaprogramming) • determine what programming paradigm a given program or programming language belongs to; • identify and discuss the design principles of a given programming language or paradigm; • choose a language or paradigm suitable for solving and particular problem and argue this choices; • write small programs in the different languages and paradigms seen in the course;

	• compare different programming languages and paradigms from the point of view of their underlying design principles; • understand the impact of different language design choices.
Inline resources	The course slides as well as other relevant and practical information related to the course will be accessible on Moodle. The same platform will also be the means of communication between the teacher(s) and the students.
Bibliography	References As the programming languages studied in this course may vary from year to year, the recommended references for this course may also vary. Nevertheless, a very useful reference which covers a wide range of programming languages remains: "Principles of Programming Languages - Design, Evaluation and Implementation" by Bruce J. MacLennan. Références Comme les langages étudiés peuvent varier d'une année à une autre, les références conseillées pour ce cours pourront varier également. Néanmoins, une référence très utile qui couvre un large éventail de langages de programmation reste : "Principles of Programming Languages - Design, Evaluation and Implementation" par Bruce J. MacLennan.
Other infos	Background : • Having a healthy interest in programming language concepts, such as seen in earlier programming-language oriented courses. • The more different programming languages a student has been confronted with before, the more he or she will appreciate this course.
Faculty or entity in charge	INFO

Programmes containing this learning unit (UE)				
Program title	Acronym	Credits	Prerequisite	Learning outcomes
Master [120] in Computer Science and Engineering	INFO2M	5		
Master [120] in Computer Science	SINF2M	5		