Coletânea de disciplinas introdutórias de programação

Referem-se a cursos considerados entre os TOP50 em Engenharia e tecnologia no ranking THE

(http://www.timeshighereducation.co.uk/world-university-rankings/2012-13/subject-ranking/subject/engineering-and-IT)

MIT (Python)

6.00 Introduction to Computer Science and Programming (Python)

video 1a aula (excelente!!!!): Em particular a apresentação em video (2008) de 6.00 Introduction to Computer Science do MIT é muito esclarecedora. O curso é dado pelo entao atual Chefe de depto de computer science do MIT E pelo seu antecessor na chefia:

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/video-lectures/lecture-1/

primeiros 12 min, 16min-20min , 30min-40min

This course is aimed at students with **little or no prior programming experience** but a desire to understand computational approaches to problem solving. Now, by definition, none of you are under-qualified for this course. In terms of being over-qualified — if you have a lot of prior programming experience, we really don't want you wasting your time, and in this case we would suggest that you talk to me about how well this class suits your needs, and to discuss other options. In addition, we want to maintain a productive educational environment, and thus we don't want over-qualified students making other students feel inadequate, when in fact they are only inexperienced.

Help students (who may or may not intend to major in computer science) to feel justifiably confident of their ability to write small programs. Map scientific problems into computational frameworks. Position students so that they can compete for jobs by providing competence and confidence in computational problem solving. Prepare college freshmen and sophomores who have no prior programming experience or knowledge of computer science for an easier entry into computer science or electrical engineering majors. Prepare students from other majors to make profitable use of computational methods in their chosen field.

Objectives:

- Learning a language for expressing computations—Python
- Learning about the process of writing and debugging a program
- Learning about the process of moving from a problem statement to a computational formulation of a method for solving the problem
- Learning a basic set of "recipes"—algorithms
- Learning how to use simulations to shed light on problems that don't easily succumb to closed form solutions
- Learning about how to use computational tools to help model and understand data

6.01 Introduction to EECS I (Python)

http://mit.edu/6.01/www/index.html

6.01 explores the application of key engineering principles, such as abstraction and modularity, in the design of systems that operate in the natural world. Topics include measuring and modeling system behaviors; assessing errors in sensors and effectors; specifying tasks; designing solutions based on analytical and computational models; planning, executing, and evaluating experimental tests of performance; refining models and designs.

Stanford (Java)

CS106A: Programming Methodology

Video curso (primeiros 15min e 44:30min a 50min):

http://www.youtube.com/watch?index=0&v=KkMDCCdjyW8&feature=PlayList&list=PL84A56BC7F4A1F852

CS198 – Teaching Computer Science:

https://cs198.stanford.edu/cs106/ProgramStructure.aspx

Carnegie Mellon – Electrical and Computer Eng: (Phyton -> C)

http://coursecatalog.web.cmu.edu/catalogcontents/

http://coursecatalog.web.cmu.edu/carnegieinstituteoftechnology/departmentofelectricalandcomputerengineering/

15-112	Fundamentals of Programming and Computer Science
15-122	Principles of Imperative Computation

15-112 Fundamentals of Programming and Computer Science (Python)

All Semesters: 12 units: A technical introduction to the fundamentals of programming with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, informal analysis, and effective testing and debugging. Starting from first principles, we will cover a large subset of the Python programming language, including its standard libraries and programming paradigms. We will also target numerous deployment scenarios, including standalone programs, shell scripts, and web-based applications. This course assumes no prior programming experience. Even so, it is a fast-paced and rigorous preparation for 15-122. Students seeking a more gentle introduction to computer science should consider first taking 15-110. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.

15-122 Principles of Imperative Computation (C)

All Semesters: 10 units: For students with a basic understanding of programming (variables, expressions, loops, arrays, functions). Teaches imperative programming and methods for ensuring the correctness of programs. Students will learn the process and concepts needed to go from high-level descriptions of algorithms to correct imperative implementations, with specific application to basic data structures and algorithms. Much of the course will be conducted in a subset of C amenable to verification, with a transition to full C near the end. This course prepares students for 15-213 and 15-210. NOTE: students must achieve a C or better in order to use this course to satisfy the pre-requisite for any subsequent Computer Science course.

Prerequisite: 15-112; Corequisite: 21-127.

Cambridge: (ML/Java)

http://www.cl.cam.ac.uk/teaching/1213/FoundsCS/

Foundations of Computer Science

Aims: The main aim of this course is to present the basic principles of programming. As the introductory course of the Computer Science Tripos, it caters for students from all backgrounds. To those who have had no programming experience, it will be comprehensible; to those experienced in languages such as C, it will attempt to correct any bad habits that they have learnt.

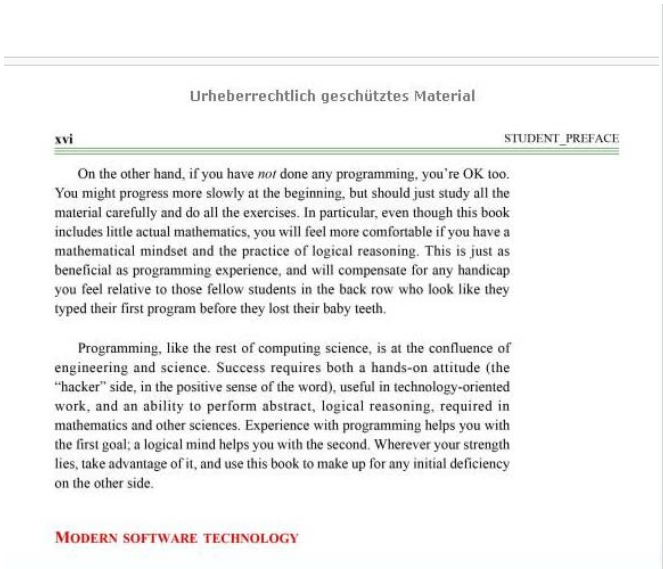
ETH Zurich (Eiffel)

http://www.vvz.ethz.ch/Vorlesungsverzeichnis/lerneinheitPre.do;VvzSessionId=FLLrRLSJHvFrT2BQyGY8X5C5Z79vyGZ3wZG58yMvXbY8b44yr4BP!1230770183?lerneinheitId=74634&semkez=2011W&lang=en
http://se.inf.ethz.ch/courses/2011b_fall/eprog/english_index.html

252-0021-00L Introduction to Programming

Introduction to fundamental concepts of modern programming and operational skills for developing high-quality programs, including large programs as in industry. The course introduces software engineering principles with an object-oriented approach based on Design by Contract as present in Eiffel, including programming exercises and a project involving advanced graphics and multimedia applications.

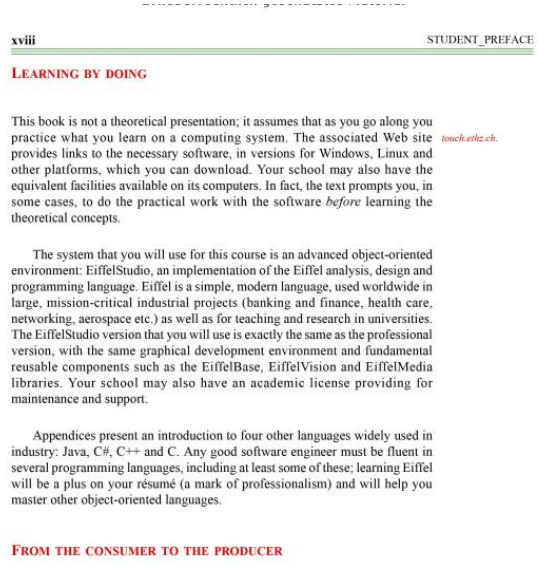
Book: Touch of Class - From object technology pioneer and ETH Zurich professor Bertrand Meyer, winner of the Jolt award and the ACM Software System Award, a revolutionary textbook that makes learning programming fun and rewarding. Meyer builds his presentation on a rich object-oriented software system supporting graphics and multimedia, which students can use to produce impressive applications from day one, then understand inside out as they learn new programming techniques. Unique to Touch of Class is a combination of a practical, hands-on approach to programming with the introduction of sound theoretical support focused on helping students learn the construction of high quality software. The use of full color brings exciting programming concepts to life. Among the useful features of the book is the use of Design by Contract, critical to software quality and providing a gentle introduction to formal methods. Will give students a major advantage by teaching professional-level techniques in a literate, relaxed and humorous way.



On the other hand, if you have *not* done any programming, you're OK too. You might progress more slowly at the beginning, but should just study all the material carefully and do all the exercises. In particular, even though this book includes little actual mathematics, you will feel more comfortable if you have a mathematical mindset and the practice of logical reasoning. This is just as beneficial as programming experience, and will compensate for any handicap you feel relative to those fellow students in the back row who look like they typed their first program before they lost their baby teeth.

Programming, like the rest of computing science, is at the confluence of engineering and science. Success requires both a hands-on attitude (the "hacker" side, in the positive sense of the word), useful in technology-oriented work, and an ability to perform abstract, logical reasoning, required in mathematics and other sciences. Experience with programming helps you with the first goal; a logical mind helps you with the second. Wherever your strength lies, take advantage of it, and use this book to make up for any initial deficiency on the other side.

MODERN SOFTWARE TECHNOLOGY



LEARNING BY DOING

This book is not a theoretical presentation; it assumes that as you go along you practice what you learn on a computing system. The associated Web site provides links to the necessary software, in versions for Windows, Linux and other platforms, which you can download. Your school may also have the equivalent facilities available on its computers. In fact, the text prompts you, in some cases, to do the practical work with the software *before* learning the theoretical concepts.

tosich.ethz.ch.

The system that you will use for this course is an advanced object-oriented environment: EiffelStudio, an implementation of the Eiffel analysis, design and programming language. Eiffel is a simple, modern language, used worldwide in large, mission-critical industrial projects (banking and finance, health care, networking, aerospace etc.) as well as for teaching and research in universities. The EiffelStudio version that you will use is exactly the same as the professional version, with the same graphical development environment and fundamental reusable components such as the EiffelBase, EiffelVision and EiffelMedia libraries. Your school may also have an academic license providing for maintenance and support.

Appendices present an introduction to four other languages widely used in industry: Java, C#, C++ and C. Any good software engineer must be fluent in several programming languages, including at least some of these; learning Eiffel will be a plus on your résumé (a mark of professionalism) and will help you master other object-oriented languages.

FROM THE CONSUMER TO THE PRODUCER

Ecole Polytechnique (Java)

INF311: Introduction à l'informatique

http://catalogue.polytechnique.fr/cours.php?id=2385

INF421: Fundamentals of programming and algorithms

http://catalogue.polytechnique.fr/cours.php?id=2471
http://www.enseignement.polytechnique.fr/informatique/INF421/

National University of Singapore (Unix C in EE)

Para estudantes de Eng. Eletrica. Para outras áreas não encontramos informações sobre linguagens (http://www.ece.nus.edu.sg/academic/undergraduate/ee/Structure.html):

CS1010E Programming Methodology (UNIX C)

http://www.comp.nus.edu.sg/~cs1010e/

This module introduces the fundamental concepts of problem solving by computing and programming using an imperative programming language. It is the first and foremost introductory course to computing and the first part of a three-part series on programming and problem solving by computing, which includes CS1020 and CS2010. Topics include problem solving by computing, writing pseudo-codes, problem formulation and problem solving, program development, coding, testing and debugging, fundamental programming constructs (variables, types, expressions, assignments, functions, control structures, etc.), fundamental data structures: arrays, strings and structures, simple file processing, and basic recursion. This module is appropriate for FoE students.

Para Civil Eng (não fala qual linguagem)
(http://www.eng.nus.edu.sg/cee/programmes/BEng_Civil.html#Degree_Requirement):

CE2409 Computer Applications in Civil Engineering

This module is designed to give civil engineering students an introduction to computer organization and operation, a knowledge of mathematical problem description and algorithm formulation, a competence in engineering problem solving using computers and equips them with fundamental knowledge and skill in computer-aided engineering graphics. The computer-aided engineering graphics includes the basic concepts in general engineering drawing, with additional focus on the drawings for Civil engineering profession. This includes the structural plan and cross section drawing, structural detailing, etc. The use of CAD software will be emphasized through hands-on sessions.

University of Toronto (C em geral)

http://www.apsc.utoronto.ca/Calendars/2012-2013/Curriculum_and_Programs.html#AECHEBASC

APS106H1 S: Fundamentals of Computer Programming

An introduction to computer systems and software. Topics include the representation of information, algorithms, programming languages, operating systems and software engineering. Emphasis is on the design of algorithms and their implementation in software. Students will develop a competency in the C programming language. Laboratory exercises will explore the concepts of both Structure-based and Object-Oriented programming using examples drawn from mathematics and engineering applications.