Universitat Autònoma de Barcelona

2020/2021

Foundations of Programming

Code: 104337
ECTS Credits: 6

Degree	Type	Year	Semester
2503758 Data Engineering	FB	1	1

The proposed teaching and assessment methodology that appear in the guide may be subject to changes as a result of the restrictions to face-to-face class attendance imposed by the health authorities.

Contact

Name: Robert Benavente Vidal
Email: Robert.Benavente@uab.cat

Use of Languages

Principal working language: catalan (cat)
Some groups entirely in English: No
Some groups entirely in Catalan: Yes
Some groups entirely in Spanish: No

Prerequisites

Given that this is an introductory course, it is assumed that students do not have any prior knowledge of the subject.

However, it is advisable to

- have a basic knowledge of any basic platform (windows, mac or linux)
- have access to a computer (ideally, a laptop)

Objectives and Contextualisation

This subject has a general and introductory character to programming. The study of the methodological aspects of programming and the learning of a high level language will be deepened. Therefore, the general objectives proposed for the subject are the following:

- Understanding the life cycle of the software: analysis of the problem (understanding what is being asked), design (proposing a solution to the problem), implementation (coding in a programming language the chosen solution), and test (carrying out a test in a systematic way to ensure the correctness of the implemented solution).
- Providing the student with the ability to design algorithms for problem solving, introducing a rigorous and structured programming methodology in a progressive and systematic way (based on the technique of descending algorithm design).
- Introducing the student to a real programming language. It is intended that the student perceives the difference between the flexibility of the pseudo-algorithmic notation used in the first topics and the strict syntax of a real programming language in different aspects: lexical (valid words of the language), syntactic (rules to combine them) and semantic (meaning of them).
- Making the student familiar with developing programs following some norms of style tending to obtain high quality programs. These style rules include those that facilitate the comprehension of the code, such as the use of comments, the indentation of the code, the use of appropriate names for data, etc.

Competences

- Design efficient algorithmic solutions to computational problems, implement them in the form of robust software developments which are structured and easy to maintain, and verify their validity.
- Make a critical evaluation of work carried out.
- Plan and manage the available time and resources.
- Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

Learning Outcomes

1. Apply strategies for debugging, testing and correcting programmes.
2. Apply the basic principles of computer structure and programming.
3. Develop programmes that are well documented, using a good programming style.
4. Implement medium-difficulty algorithmic problems in a programming language.
5. Make a critical evaluation of work carried out.
6. Plan and manage the available time and resources.
7. Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.

Content

Unit 1: Introduction to the computer

History. Functional structure of the computer. Programs / instructions. Conceptual levels of the computer.

Unit 2: Problem solving: introduction to algorithmics and programming.

Introduction to problem solving. Algorithm concept Phases in the development of algorithms. Programming as an engineering discipline. Software life cycle. Basic elements of an algorithm. Tools for the representation of algorithms. Programming languages. Classification. Language translators: Compilers and interpreters.

Unit 3: Basic concepts and control structures

Definition of variables and constants. Type of fundamental data. Sequential structure. Selection or conditional structures. Iterative or repetitive structures.

Unit 4: Data structures

Unidimensional arrays: strings, tuples and lists. Search and traverse.

Unit 5: Subprograms

The concept of subprogram as an abstraction of operations. Location, nesting, scope and visibility. Definition of functions and procedures. Calls to functions and procedures. Modular design descending.

Unit 6: Files

Basic Definitions. Input / output of data in files. Types of access to files.

Unit 7: Prevention and detection of errors

Types of errors. Exceptions and *asserts.* Preventive programming. Debugging programs.

Unit 8: Introduction to object-oriented programming

Classes and objects. Attributes and methods. Encapsulation. Definition of classes.

Unit 9: Complex data structures

Lists: iterators, generators, functional paradigm, and list comprehensions. Sets. Dictionaries.

Methodology

The management of the teaching of the subject will be done through the documentary manager Caronte (http://caronte.uab.cat/),which will serve to see the materials, manage the groups of practices, make the corresponding deliveries, see the marks, and communicate with the teaching staff, etc. In order to use it, it is necessary to do the following steps:

1. Register by giving the name, NIU, and a photo in JPG format. If you have already registered for another subject, you do not need to do it again; You can go to the next step.
2. Enroll in the type of teaching "Fundamentals of Programming", giving as a subject code the one provided on the first day of class.

The teaching of the subject takes place in two sessions of two hours each in a computer classroom. In these sessions, the following teaching activities can be differentiated:

MD1 Exhibition of contents in class: Presentation of the theoretical contents to work in the class. Some of these contents must have been prepared before the class from reading texts, viewing videos, searching for information, etc. The contents presented will be directly related to the problems proposed in other teaching activities, so that they will be the basis on which other activities of the course will be developed.

MD2 Participative lectures: Joint resolution of the set of problems proposed to students. All topics will be accompanied by a list of problems that the student must solve. In this sense, and as the student progresses in the depth of their knowledge, these problems will be little by little more complex, allowing in this way to clearly appreciate the advantages of using the methodological tools taught during the course. All the problems developed in class and others that can be proposed can be found on the Caronte platform, and will be self-evaluable. These activities should allow the student to deepen understanding and personalize knowledge. The fact that they are self-evaluable allows us to adjust the pace of consolidation and reflect on our own learning.

MD3 Tutoring sessions: Hours freely available for questions about aspects in which students need additional help from the faculty.

MD4 Programming short projects: Realization of short practical projects to deepen the applied aspects of the theory. These projects will be resolved in small groups, where each member must do a part of the work and put it in common with the rest of the group to have the final solution.

MD5 Evaluation activities: See evaluation section of this teaching guide.

Transversal competences

Competences T06 and T08 will be worked on and evaluated throughout the course in the following activities:

Competence	MD1	MD2	MD4
T06 - Make a critical evaluation of work carried out	Personal work of preparation of materials Evaluation: Theoretical Exam (Tp and Tf)	Evaluation: Problem Solving (P)	Evaluation: Practical projects (PLg and PLi)

T8 - Plan and manage the available time and resources	The guide of the project will give indications of how to do the planning. Evaluation: Group practical projects (PLg)

Activities

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Laboratory classes	11	0.44	2, 1, 5, 3, 4, 6, 7
Problems classes	26	1.04	2, 1, 5, 3, 4, 7
Theory classes	10	0.4	2, 1, 5, 3, 4, 7
Type: Supervised			
Programming projects	30	1.2	2, 1, 5, 3, 4, 6, 7
Type: Autonomous			
Classes preparation and personal work	20	0.8	2, 1, 3, 4, 6, 7
Solving self-assessment problems	48	1.92	2, 1, 5, 3, 4, 7

Assessment

Scheduled evaluation process and activities

The subject is evaluated through continuous evaluation. Throughout the course the evaluation activities will be carried out:

Activity	Date	Recovery	Percentage	Minimum grade
Partial Theory Exam (Tp) Individual	Check Planning	Check Planning	15%	No
Final Theory Exam (Tf) Individual	Check Planning	Check Planning	35%	Tf >= 5
		No	20%	No

Delivery of Problems (P) Individual	Each Week			
Laboratory Practice (PLg)	Check Planning	Check Planning	20%	PLg >= 5
Laboratory Practice (PLi) Individual	Check Planning	Check Planning	10%	PLi >= 5

To be able to approve the subject, through continuous assessment, the result of the weighted sum of the evaluation activities must be greater than or equal to 5, and a minimum grade of 5 must be obtained in activities 2 (Tf), 4 (PLg) and 5 (PLi).

Programming of evaluation activities

The dates for evaluation and submission of works will be published in the documentary manager Charon and may be subject to programming changes for reasons of adaptation to possible incidents. Always be informed by the document manager about these changes as it is understood that this is the usual platform for exchange of information between teachers and students.

Recovery procedure

The student may apply for recovery whenever he has submitted to a set of activities that represent at least two thirds of the total grade of the subject.

The theoretical exams (Tp and Tf) will be able to be recovered in a single exam of recovery in the dates fixed by the coordination of the Degree. The exam of recovery will have a percentage of 50% on the final grade. The mark of the exam of recovery, in case of being realized, will substitute to the marks of the partial examinations (Tp) and final (Tf) realized during the course.

In accordance with the coordination of the Degree and the direction of the School of Engineering, the Individual Problem Resolution (P) activity can not be recovered.

Grade review procedure

Students will have the right to review the theoretical exams (Tp and Tf). The place, date and time of revision will be provided on the day the marks are published. If the student does not appear in this review, this activity will not be reviewed later. Only in justified cases may a review be made after the date set and always up to a maximum of 7 calendar days.

There will not be review of the marks of the practical projects (PLg and PLi) because the evaluation is done in front of the students.

Qualifications

Non-Evaluable (NA): Any student who delivers a practice or a scheduled assessment will have a grade. It will only be considered not evaluable in the case of not delivering any evaluable activity.

Final Mark: Is obtained with the weighted sum according to the criteria set in the section on evaluation activities. If in any activity the minimum grade is not reached, the grade will come out of the following formula:

Minimum (weighted sum of grades, 4.9)

Honors (MH): To award a grade of honor is the decision of the faculty responsible for the subject. The regulations of the UAB indicate that MH can only be granted to students who have obtained a final grade equal to or greater than 9.00. You can grant up to 5% of MH of the total number of students enrolled. If the number of students with a grade greater than or equal to 9 is more than 5% of the total enrollment, students were prioritized according to the following rules (in order):

1. Students who have made fewer recoveries of activities (because they failed the activity or because they tried to improve their mark).
2. Students with more marks higher than 9 in activities 1, 2, and 5.
3. Students with the best overall mark.

Irregularities on the part of the student, copy and plagiarism

Without prejudice to other disciplinary measures that they deem appropriate, will be scored with a zero the irregularities committed by the student that may lead to a variation of the rating of an evaluation act. Therefore, copying, plagiarism, cheating, letting copy, etc. in any of the evaluation activities will involve suspending with a zero. The evaluation activities qualified in this way and by this procedure will not be recoverable. If it is necessary to pass any of these evaluation activities to pass the subject, this subject will be suspended directly, without the opportunity to recover it in the same course. In this case, the numerical grade of the file will be the lower value between 3.0 and the weighted average of the marks.

Evaluation of repeating students

The repeating students who have passed the subject "Advances Programming" the previous course, will be able to pass the subject presenting some exercises of synthesis of the course. The final grade in this case will be equal to 5. Students who want to take this way must inform the faculty of the subject the first week of class.

The rest of the repeating students must take the complete course. No marks from previous courses will be maintained.

Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
1. Partial theory exam	15%	2	0.08	2, 1, 5, 3, 4, 7
2. Final theory exam	35%	2	0.08	2, 1, 5, 3, 4, 7
3. Delivery of problems	20%	0	0	2, 1, 3, 4
4. Programming projects (group assessment)	20%	0.5	0.02	2, 1, 3, 4, 6
5. Programming projects (individual assessment)	10%	0.5	0.02	2, 1, 5, 3, 4

Bibliography

J. Guttag. *Introduction to Computation and Programming Using Python: With Application to Understanding Data. Second Edition*. MIT Press. ISBN-10: 9780262529624

S. Chazallet *Python 3. Los fundamentos del lenguaje*. Eni, ISBN-10: 2409006140

E. Matthes. *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*. No Starch Press ISBN-10: 1593276036

M. Myers. *A Smarter Way to Learn Python: Learn it faster. Remember it longer*. Createspace Independent Pub ISBN-10: 1974431479

A. Prieto, A. Lloris, J.C. Torres. *Introducción a la Informática*. Mc Graw-Hill ISBN-10: 8448146247

A. Prieto, B. Prieto. *Conceptos de Informática*. Mc Graw-Hill, Schaum ISBN-10: 8448198573

L. Joyanes Aguilar. *Fundamentos de Programación: Algoritmos, Estructuras de Datos y Objetos*. Mc. Graw-Hill. ISBN-10:8448161114