Programming

Academic Year: (2019 / 2020) **Review date: 22-07-2019**

Department assigned to the subject: Computer Science and Engineering Department

Coordinating teacher: GARCIA OLAYA, ANGEL

Type: Basic Core ECTS Credits : 6.0

Year : 1 Semester : 1

Branch of knowledge: Engineering and Architecture

REQUIREMENTS (SUBJECTS THAT ARE ASSUMED TO BE KNOWN)

None

OBJECTIVES

Specific competences:
- Knowledge of imperative programming basic concepts from an object oriented point of view (PO a,k).
- Basic knowledge of the syntax of an object oriented programming language (PO k).
- Ability to break down a real problem by following an object oriented methodology, in order to code it into a computer program (PO a,c,e,k).
- Knowledge of simple structures for information management (PO a).
- Ability to understand technical documents and to reuse third parties programming code (PO a,k).
- Basic knowledge of computational complexity (PO a,k).

General competences:
- Ability to self-organize and individual work and learning process planning (PO k).
- Ability to work in groups (PO d).
- Oral and writing skills (PO g).
- Analysis and synthesis abilities (PO a).
- Ability to decide among alternatives (PO e).
- Motivation for quality (PO k).

DESCRIPTION OF CONTENTS: PROGRAMME

1. Introduction
   a. Components of a program: data and algorithms
   b. Computer architecture
   c. Data storage and coding
   d. Brief introduction to the history of programming: from binary code to component oriented programming
   e. Compilation vs. Interpretation
   f. Programming paradigms: imperative, logical and functional programming
2. Flow diagrams
3. Introduction to Python
   a. The Python interpreter
   b. Basic data types
   c. Programs
   d. Variables and constants
   e. Input, output and random numbers
   f. Operators
   g. Casting
   h. Comments
   i. Debugging: errors
4. Flow control: conditionals and loops
   a. Conditionals
   b. Scope and blocks
   c. Loops
5. Simple data structures
   a. Lists and tuples
   b. Dictionaries
6. Functions

a. Decomposition
b. Code reuse
c. Implementation hiding
d. Pass by value and pass by reference
e. Introduction to recursion
7. Introduction to Object Oriented Programming
8. Algorithms
a. Introduction to computational complexity
b. Search (lineal and binary)
c. Sorting (bubble sort, insertion sort, selection sort)

LEARNING ACTIVITIES AND METHODOLOGY

1. Lectures (1 ECTS)(PO a,k)
- Devoted to teaching basic competences, especially those related to basic imperative programming techniques
2. Individual work in class and at home (3 ECTS)(PO a,c,e,g,k)
- Devoted to autonomous learning of the specific competences, guaranteeing a minimum knowledge background to contribute with to the group. A problem based learning (PBL) methodology will be followed.
- This activity will also contribute to the following generic competences: Oral and writing skills, Analysis and synthesis abilities, Ability to decide among alternatives, Motivation for quality and Ability to self-organize, and individual work and learning process planning capacity.
3. Group works and practices (2 ECTS) (PO a,c,d,e,g,k)
- It completes former activities, fixing both general and specific competences and focusing on the following generic competences: Ability to work in groups and Oral and writing skills. A PBL methodology will be also used.

METHODOLOGY

Classes will be divided into lectures and laboratory sessions. In addition, students are expected to work autonomously at home.

- Lectures: they will be used to present general programming concepts using Python as language. Example programs will be written during the class, to reinforce the theoretical concepts. Short test can be performed at the end of each class to evaluate the understanding of the main concepts presented in that class and the previous ones. These test questions will be collected and uploaded to Aula Global where students will be able to answer them as many times as they want. At least one mid-term exam will take place during these sessions (see weekly schedule).

- Laboratory classes: they will consist of three parts. In the first part, the students, randomly selected, will be asked to explain the previous week proposed exercises. Next, a problem will be posed and solved in collaboration by students and teacher. Finally, a series of problems will be presented for the students to solve them. Usually these problems will be performed in pairs of two students and will need to be uploaded to Aula Global by the end of the week. A final project will be presented to be performed also in groups of two. It could include partial deliveries (see weekly schedule). Both the final project and the weekly assignments will be performed in Python.

- Student's autonomous work: students are expected to finish at home the weekly assignments and to deliver them using Aula Global.

Office hours
Following university rules each professor will publish his/her preferred office hours in Aula Global. Despite this, students are encouraged to contact the professor by email to arrange an appointment any time they want, whether in the preferred hours or not.
Small doubts not requiring a face-to-face visit can be made by email. General doubts should be posted in the course forum, available in Aula Global. These doubts can be solved either by the professors or by other students.
For the final project, individual tutorship sessions will be arranged for each group.

ASSESSMENT SYSTEM

Combined and continuous evaluation of every activity performed by the student either individual or in group, considering the effort devoted by each student to each of the formative activities described.
A formative evaluation through continuous feedback will be performed, allowing the student to know which is his/her knowledge and what it is expected from him/her. Evaluation will be composed of:
- Mid-term exams and tests at the end of classes: 1.5 points (PO a,c,e,g,k) (competencies CGB4 and CECRI6).

- Individual oral presentations of the exercises: 0.5 points (competencies CGB4 and CECRI6).
- Weekly assignments: 0.5 points (if at least 80% of them are presented) (PO a,c,d,e,g,k) (competencies CGB4, CGB5, CECRI6 and CECRI7)
- Final project: 2.5 points (PO a,c,d,e,g,k) (competencies CGB4, CGB5, CECRI6 and CECRI7)
- Final exam, marked with 5 points (competencies CGB4 and CECRI6) (PO a,c,e,g,k).

To pass the subject, a total of 5 points are needed, which will be calculated in the following way, depending on the call:

In the January call the final mark will be the maximum between:
1) 60% of the mark of the final examination (out of 10 points maximum)
and
2) continuous evaluation mark + final examination mark (out of 5)

To apply the formula of item 2), the mark of the final examination must be bigger or equal than 2 out of 5 (or 4 out of 10). In case of plagiarism in any of the practical exercises, all the involved students will loose the right to follow continuous evaluation and formula 1) will be applied.

In the June/July call the final mark will be the maximum between:

1) the mark of the final examination (out of 10 points maximum)
and
2) continuous evaluation mark + final examination mark (out of 5)

As in the January period, to apply the formula of item 2), the mark of the final examination must be bigger or equal than 2 out of 5 (or 4 out of 10)

It will not be possible to present any kind of practical exercises in June exams period.

Individual activities will account for a 70% of the final mark, group activities will constitute the remaining 30%. Continuous evaluation will represent a 50% of final mark, while final examination will account for the remaining 50%.

% end-of-term-examination:	50
% of continuous assessment (assigments, laboratory, practicals…):	50

BASIC BIBLIOGRAPHY

- Ana Bell Get Programming Learn to code with Python, Manning publications, 2018
- John S. Conery Explorations in Computing: An Introduction to Computer Science and Python Programming, CRC Press, 2014

BASIC ELECTRONIC RESOURCES

- Guido van Rossum, Barry Warsaw, Nick Coghlan . PEP 8 -- Style Guide for Python Code: https://www.python.org/dev/peps/pep-0008/
- Python Software Foundation . Python for Beginners: https://www.python.org/about/gettingstarted/