ESCI upf. School of International Studies

TEACHING PLAN FOR

Programming and Algorithms I

1. Basic description

Name of the course: Programming and Algorithms I
Module: Computers Science
Academic year: 2016-2017
Year: 2016
Term: First
Degree / Course: First
Code: 51103
Number of credits: 6
Total number of hours committed: 150
Teaching language: English
Lecturer: José Luis Balcázar, Jorge Castro

Timetable: See official calendar

2. Presentation of the course

Contextualization. A degree in Bioinformatics must prepare students with an excellent command of Computing by definition. Computing has many facets, several of which will be needed by the bioinformatician. Among these, a clear understanding of programming and of algorithms is, probably, the most basic one.

Type and focus of the subject. Whereas a general perspective of the diverse approaches to programming and algorithms is convenient, and will be provided briefly, one of these existing options has been chosen for in-depth development in this first programming course; others will be covered in later courses. The usage of the programming language Python 3 will allow us to present, within sufficient simplicity, a perspective of both the object-oriented extension of imperative programming and glimpses of scripting and of functional programming; all of these among the most powerful tools available.

Key aspects. Our goal is that the students understand deeply the reasoning methods behind the programming activity, in a way that allows them both to grasp and enjoy the most of the subsequent courses in the degree, and to learn on themselves, if they are motivated, additional programming styles and languages.

General recommendations. The following observation has been attributed to Don Knuth: the main capability of the computing professional is the ability to change very fast among vastly different abstraction levels, from a syntactic detail to a general model of a system. Students and teachers alike are expected to adhere to very careful attitudes, a must for programming, where an extreme care for the details is displayed at all times; while, simultaneously, practicing the necessary abilities of abstraction and improving one's capabilities in this sense.

3. Competences to be worked in the course

General competences	**Specific competences**
CB2, CB5, CG1	CE2, CE3, CE4, CE5, CE8

I. General competences

CB2. That the students know how to apply their knowledge to their work or vocation in a professional manner and have competencies typically demonstrated through devising and defending arguments and solving problems within their field of study.

CB5. That the students have developed those skills needed to undertake further studies with a high degree of autonomy.

CG1. That the students will acquire an intra- and interdisciplinary training in both computational and scientific subjects with a solid basic training in biology.

II. Specific competences

CE3. To identify modeling and optimization of widely used programming languages in the field of Life Sciences, in order to develop and evaluate technical and/or computational tools.

CE5. To apply mathematical foundations, algorithmic principles and computational theories in the modeling and design of biological systems.

<u>Learning outcomes</u>

RA3.1. Know existing techniques and computational tools in a particular field.

RA3.4. Understand and develop algorithms with computer languages.

RA5.1. Recognize and use the basic tools of mathematical language.

4. Contents

This is the first of several courses covering the basics of algorithms and data structures, with emphasis on fundamental algorithms and basic control structures and applications.

The course will cover:
Organization of computation into functions and function composition.
Organization of related functions into classes by means of the object-oriented paradigm.

Handling of basic types: boolean, numeric, and textual. Handling of basic data structures within the chosen approach: tuples, lists, files, and dictionaries.
Advantages of the functional approach: map, reduce, list comprehension, iterators. Development of new classes with or without inheritance mechanisms.

5. Assessment

Along the whole course, the participation in group work on planned activities in the classroom will be subjectively assessed by the instructor, more precisely in each of the approximately weekly seminar sessions. This assessment will have a weight of 10% of the final grade.

There will be three exam-like events assessing individual capabilities with a strong emphasis on application: namely, within a limited time, the students will be required to produce Python 3 programs that work correctly for a task to be proposed on the spot, with automatized evaluation of the correctness, and which are subsequently also evaluated (by human eyes) in terms of each of the decisions leading to the final program. The first of these three events will require only command of the limited amount of material covered in the first few weeks of the course. The second and third will assume familiarity with the whole of the material covered, and will take place at the end of the course. The first event will contribute 20% of the final grade, the second 30%, and the fourth 40%.

Prerequisites for taking the subject

No prerequisites are needed

Terms and conditions for extraordinary exam sitting

Extraordinary exam will take place according to schedule fixed by the Degree Coordination. Failure to attend this exam imply the student will keep his initial score.

Assessment elements	Time period	Type of assessment		Assessment agent			Type of activity	Grouping		Weight (%)
		Comp	Opt	Lecturer	Self-assess	Co-assess		Indiv	Group (#)	
participation in group work	weekly	X		X			Conceptual and pursuit of the subject	X		10%
Practical exam 1	First weeks	X		X				X		20%
Practical exam 2	Middle-end	X		X				X		30%
Practical exam 3	End	X		X				X		40%

Working competences and assessment of learning outcomes:

	CB2	CB5	CG1	CE3	CE5
participation in group work	X	X	X	X	X
Practical exam 1	X	X	X	X	X
Practical exam 2	X	X	X	X	X
Practical exam 3	X	X	X	X	X

6. Bibliography and teaching resources

- Basic bibliography
- Think Python 2nd edition by Allen B. Downey (Green Tea Press 2015): http://greenteapress.com/wp/think-python-2e/
- Supplementary bibliography
- How to Think Like a Computer Scientist: Learning with Python Interactive Edition 2.0, based on an earlier (2002) version of the previous reference: http://interactivepython.org/courselib/static/thinkcspy/index.html
- Introduction to Programming in Python: An Interdisciplinary Approach by Robert Sedgewick, Kevin Wayne, and Robert Dondero (Pearson 2015): http://introcs.cs.princeton.edu/python/home/

- Teaching resources
- Other websites supporting interactive Python:
- interactivepython.org
- jupyter.org
- www.pythonanywhere.com
- trinket.io

- Source Python materials from python.org:
- The Python standard library: docs.python.org/3/library/index.html
- The Python Tutorial: docs.python.org/3/tutorial/
- Beginner's Guide: wiki.python.org/moin/BeginnersGuide
- Style Guide for Python Code: python.org/dev/peps/pep-0008
- Docstring Conventions: python.org/dev/peps/pep-0257/

7. Methodology

Conceptual materials necessary to understand the capabilities of each programming ingredient covered will be provided in the classroom (theory sessions); whenever possible and convenient, the students will have also a limited and closely guided first contact with the direct experience of using those concepts in the same sessions. On the basis of those materials, groups of students, with help from the instructors when required, will develop further applications (seminar sessions). Then, each student being likely to need their own pace, individual application and programming sessions will take place, again with help from the instructors whenever required, and employing as auxiliary validation mechanism the same automatized evaluation infrastructure that will be used for the validation of the exams (lab sessions).

Further, independent, individual study and work efforts will be necessary, in very varying amounts depending on the profit and capability of each student, in order to absorb the essentials of the conceptual developments provided and in order to complement the programming abilities with further interactions with the automatized evaluation tool.

8. Scheduling activities

Seminars
- Seminar group A -101
- Seminar group B- 102

Practical classes (lab)
- Practical group A -111
- Practical group B- 112

Week	Activity in the classroom Grouping/type of activity	Activity outside the classroom Grouping/type of activity
Week 1	1.Lecture Class (4h) 2.Seminar 101 (1h) 2.Seminar 102 (1h) course presentation, generalities on computing, Python, type 'str'	Individual study and programming practice
Week 2	1.Lecture Class (2h) 2.Seminar 101 (1h) 2.Seminar 102 (1h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h)	Individual study and programming practice

	'bool's; 'int's; 'tuple's; indexing; defining functions; 'for' loops	
Week 3	1.Lecture Class (2h) 2.Seminar 101 (1h) 2.Seminar 102 (1h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h) 'float's; 'list's; 'file's; 'while' loops	Individual study and programming practice
Week 4	1.Lecture Class (2h) 2.Seminar 101 (1h) 2.Seminar 102 (1h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h) 'class's (no inheritance for the moment) 'dict's (including 'json' technology); 'hash' schemes	Individual study and programming practice
Week 5	Exam’s week	
Week 6	1.Lecture Class (2h) 2.Seminar 101 (1h) 2.Seminar 102 (1h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h) 'dict's (including 'json' technology); 'hash' schemes light introduction to numpy: arrays, matrices	Individual study and programming practice
Week 7	1.Lecture Class (2h) 2.Seminar 101 (2h) 2.Seminar 102 (2h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h) more on lists: comprehensions, map,	Individual study and programming practice

	reduce, etc	
Week 8	1.Lecture Class (2h) 2.Seminar 101 (1h) 2.Seminar 102 (1h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h) class inheritance, mixins	Individual study and programming practice
Week 9	1.Lecture Class (2h) 2.Seminar 101 (1h) 2.Seminar 102 (1h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h) panoramic of all the topics covered and hints of how each of them is extended	Individual study and programming practice
Week 10	2.Seminar 101 (1h) 2.Seminar 102 (1h) 4.Practical class 111 (2h) 4.Prectical class 112 (2h) further practice and applications of all the topics covered	Individual study and programming practice
Week final exams		

Total or partial copy and/or plagiarism will imply a failure in the subject with a final grade of zero points and no access to the make-up exam. According to the academic regulations specified in the Disciplinary rules for students of Universitat Pompeu Fabra, other additional sanctions may apply depending on the seriousness of the offence.