Course Specifications

Valid as from the academic year 2016-2017

Informatics (O000096)

Course size *(nominal values; actual values may depend on programme)*

Credits 10.0 **Study time 300 h** **Contact hrs** 120.0 h

Course offerings and teaching methods in academic year 2016-2017

A (year)	seminar: practical PC room classes	60.0 h
	lecture	60.0 h

Lecturers in academic year 2016-2017

De Neve, Wesley	TW06	lecturer-in-charge

Offered in the following programmes in 2016-2017	crdts	offering
Bachelor of Science in Food Technology	10	A
Joint Section Bachelor of Science in Environmental Technology, Food Technology and Molecular Biotechnology	10	A
Bachelor of Science in Environmental Technology	10	A
Bachelor of Science in Molecular Biotechnology	10	A

Teaching languages

English

Keywords

Computational thinking, Command line, Creative problem solving, Linux, Programming, Python, Scientific problem solving, Scripting

Position of the course

Scientists and engineers are often confronted with time-consuming and repetitive tasks when making use of computers to process and analyze data. These tasks may include collecting data from websites, converting files from one format into another, and analyzing, summarizing, and visualizing the data obtained. The exponential flow of newly incoming data forces present-day scientists and engineers to automate these tasks, so to be able to speed up their daily job routines.

This course teaches you how to translate time-consuming and repetitive tasks in such a way that they can be performed automatically by a computer. To that end, the necessary skills for computer-based creative problem solving will be acquired (1) by learning to work and think in Python and (2) by learning to work with the Unix command line. The programming problems that need to be solved are taken from different scientific disciplines, including biology, chemistry, physics, computer science, and mathematics.

In order to take this course, students do not need to have prior programming experience. However, in order to successfully complete this course, students need to have an aptitude for mathematics and logic. In addition, given that this course follows a 'learning by doing' and a 'learning from mistakes' approach, students need to have a willingness to solve programming problems on a weekly basis.

Contents

Programming is the process of designing, writing, testing, debugging and maintaining the source code of computer programs. This requires knowledge of the syntax and semantics of a programming language and the skills to write programs in that language. Additionally, and maybe most importantly, when writing computer programs, one must learn how to think as a programmer. This process of computational thinking, or in other words, learning the skill of problem solving by programming, is a common theme throughout the whole course.

In this course, students learn how to make use of the Python programming language to solve a plethora of scientific problems. To that end, attention is paid to:
- basic components: instructions, variables, data types, and operators;
- control structures: conditional statements, repetitive statements, and functions;
- data structures: strings, lists, tuples, dictionaries, sets, modules, and files;
- object-oriented programming: objects, classes, attributes, methods, encapsulation, polymorphism, and inheritance; and
- databases and SQL.

Furthermore, in this course, students learn how to make use of Unix-based tools to automate repetitive or complex tasks. To that end, attention is paid to:
- principles of Unix-based operating systems;
- interactive command line usage; and
- shell scripting and regular expressions.

Initial competences

An aptitude for mathematics and logic.
An interest in solving scientific problems.
Prior programming skills are not required.
Some basic computer knowledge is advantageous.

Final competences

The student will be able to translate a task described in natural language into a program written in Python, and s/he will subsequently be able to execute this program by means of a computer, generating the required results.
The student will be able to test and debug a program (module) and make the right choices between different alternatives when implementing a program, taking into account performance (efficiency), coding style, and correctness.
The student will have a working knowledge about the basic principles of object-oriented programming.
The students will be able to automate repetitive or complex tasks using the Unix command line, shell scripting, and regular expressions.
The student will be able to transfer the computational concepts learned to other computational environments (e.g., environments making use of MATLAB or R).

Conditions for credit contract

Access to this course unit via a credit contract is determined after successful competences assessment

Conditions for exam contract

This course unit cannot be taken via an exam contract

Teaching methods

Lecture, seminar: practical PC room classes

Learning materials and price

Handbook: The Practice of Computing using Python, William Punch, Addison Wesley, ISBN-13: 978-0136110675.
Handbook: Mark G. Sobell, A Practical Guide to Linux: Commands, Editors, and Shell Programming, Second Edition. ISBN-13: 978-0133085044.
Slides shown during the lectures will be made available on Minerva, together with additional learning materials (e.g., background information and links to relevant websites). Digital tools like Eclipse for writing and debugging Python source code, the Online Python Tutor for visualizing code execution, and an online platform for automated verification of the correctness of solutions written in Python.
Students are required to have a personal laptop for use in this course.

References

Mark Lutz (2009). Learning Python: Powerful Object-Oriented Programming (4th edition). O'Reilly Media, ISBN-13: 978-0596158064.
Mark Pilgrim (2009). Dive into Python. CreateSpace, ISBN-13: 978-1441413024. (free download @ http://diveintopython.org).
Hans Peter Langtangen (2009). A Primer on Scientific Programming with Python. Springer, ISBN-13: 978-3642024740.
Tony Gaddis (2009). Starting Out with Python. Pearson Education - Addison Wesley, ISBN-13: 978-0321549419.
Michael H. Goldwasser (2007). Object-Oriented Programming in Python. Prentice Hall, ISBN-13: 978-0136150312.
Jason Kinser (2008). Python for Bioinformatics. Jones & Bartlett Publishers, ISBN-13: 978-0763751869.
Sebastian Bassi (2009). Python for Bioinformatics. Chapman & Hall, ISBN-13: 978-1584889298.

Mark G. Sobell (2012). A Practical Guide to Linux: Commands, Editors, and Shell Programming. Second Edition. Prentice Hall, ISBN-13: 978-0133085044.
William Punch and Richard Enbody (2012). The Practice of Computing using Python. Second Edition. Addison Wesley, ISBN-13: 978-0136110675.
Steven Haddock and Casey Dunn (2010). Practical Computing for Biologists. First Edition. Sinauer Associates, Inc, ISBN-13: 978-0878933914.
Ashley Shade, Tracy K. Teal (2015). Computing Workflows for Biologists: A Roadmap. PLOS Biology.
Pavel A. Pevzner (2004). Educating Biologists in the 21st Century: Bioinformatics Scientists versus Bioinformatics Technicians. Bioinformatics, Vol.20, No.14, pages 2159–2161.
Alejandra J. Magana, Manaz Taleyarkhan, Daniela Rivera Alvarado, Michael Kane, John Springer, and Kari Clase (2014). A Survey of Scholarly Literature Describing the Field of Bioinformatics Education and Bioinformatics Educational Research. CBE—Life Sciences Education, Vol. 13, pages 607–623.

Course content-related study coaching

Evaluation methods

end-of-term evaluation and continuous assessment

Examination methods in case of periodic evaluation during the first examination period

Open book examination, skills test

Examination methods in case of periodic evaluation during the second examination period

Examination methods in case of permanent evaluation

Assignment

Possibilities of retake in case of permanent evaluation

examination during the second examination period is possible in modified form

Extra information on the examination methods

During the first examination period, the periodic evaluation accounts for 75% of the final score and the non-periodic evaluation (hands-on sessions) accounts for 25% of the final score. To qualify for passing, both the score of the periodic and the non-periodic evaluation should be at least equal to 8/20. If that is not the case, the total course score will be subject to an upper limit of 7/20.
The periodic evaluation consists of a partial exam at the end of the first semester and a final exam at the end of the second semester. If the score of the partial exam at the end of the first semester is higher than or equal to 10/20, then the final exam at the end of the second semester only covers the course content of the second semester. In addition, the score of the periodic evaluation is equal to the average of the score of the partial exam at the end of the first semester and the score of the final exam at the end of the second semester. If the score of the partial exam at the end of the first semester is lower than 10/20, then the final exam at the end of the second semester covers the course content of both the first and the second semester. In addition, the score of the periodic evaluation is equal to the score of the final exam at the end of the second semester.
Students who passed the partial exam at the end of the first semester are allowed to retake the exam questions related to the course content of the first semester during the final exam at the end of the second semester. The computation of the final score will then make use of the last of the two scores obtained for the course content of the first semester.
During the second examination period, the non-periodic evaluation cannot be retaken. Therefore, the final score for the second examination period is computed twice. The first computation takes into account both the score of the non-periodic evaluation (that is, the score obtained during the first examination period, on a maximum of 5) and the score of the second examination period (on a maximum of 15). The second computation only takes into account the score of the second examination period (on a maximum of 20). The final score for the second examination period is then equal to the maximum of the above two computations.

Calculation of the examination mark

Periodic evaluation: open book examination, skills test - 75%
Non-periodic evaluation: assignment - 25%