Course Specifications
Valid as from the academic year 2018-2019

Lecturers in academic year 2018-2019
De Neve, Wesley
KR01
lecturer-in-charge

Course offerings in academic year 2018-2019
A (year)
English

Offered in the following programmes in 2018-2019

<table>
<thead>
<tr>
<th>Programme</th>
<th>crds</th>
<th>offering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bachelor of Science in Environmental Technology</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>Bachelor of Science in Food Technology</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>Bachelor of Science in Molecular Biotechnology</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>Joint Section Bachelor of Science in Environmental Technology, Food Technology and Molecular Biotechnology</td>
<td>10</td>
<td>A</td>
</tr>
</tbody>
</table>

Teaching languages
English

Keywords
Command line, Computational thinking, Creative problem solving, Programming, Python, SQL, Scientific problem solving, UNIX

Position of the course

Scientists and engineers are often confronted with time-consuming and repetitive tasks when processing and analysing data, namely collecting information from websites, converting files from one format to another, and analysing, summarizing and visualizing the information obtained. In addition, the exponential flow of newly incoming information requires present-day scientists and engineers to be able to automate these tasks, in order to speed up their daily job routines.

This course teaches students how to describe 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 through learning to work and think in Python, a popular programming language, and in UNIX, the workhorse operating system of science and engineering. The programming problems that need to be solved are taken from different scientific disciplines, including mathematics, biology, chemistry, physics, and computer science.

In order to attend this course, students do not need to have any prior programming experience. However, in order to be successful for 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 regular 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 ability 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

(Contact hrs 120.0 h; Study time 300 h; Credits 10.0)
solve a plethora of 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;
- text files: reading, processing, and writing data; and
- object-oriented programming: objects, classes, attributes, methods, encapsulation, polymorphism, and inheritance.

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;
- consulting technical documentation;
- file systems;
- interactive command line usage;
- text file formats (HTML, XML, CSV, FASTA)
- filters, redirection, and pipes;
- interactive text editing;
- automated text editing using regular expressions; and
- the basics of scripting.

Finally, in this course, students learn how to make use of the Structured Query Language (SQL) to communicate with a relational database.

Initial competences

An aptitude for mathematics and logic.

An interest in solving (scientific) problems.

Some basic computer knowledge is advantageous (prior programming skills are not required).

Final competences

1. Translate a task described in natural language into a program written in Python.
2. Execute a program written in Python by means of a computer, generating a correct result.
3. Test and debug a program written in Python.
4. Make the right choices between different alternatives when writing a program in Python, taking into account performance (efficiency), coding style, and correctness.
5. Demonstrate a working knowledge about the basic principles of object-oriented programming.
6. Automate repetitive and complex tasks by means of UNIX-based tools.
7. Work interactively and non-interactively with operating systems, computer networks, file systems, and text editors.
8. Understand the structure of various text file formats, including HTML, XML, CSV, and FASTA.
9. Apply SQL to communicate with a relational database.

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

Guided self-study, demonstration, lecture, seminar: practical PC room classes

Learning materials and price


Slides shown during the lectures will be made available on Minerva (in English), together with additional learning materials (e.g., background information and links to relevant websites).
Free digital tools like Eclipse and PyCharm for writing and debugging Python source code, the Online Python Tutor for visualizing code execution, the Dodona online platform for automated verification of the correctness of solutions written in Python, and a remote UNIX environment (e.g., Helios).

Students are required to have a personal laptop for use in this course.

References


Course content-related study coaching

The syntax and the semantics of the programming language Python, the data management language SQL, and selected UNIX tools are presented in the course handbooks and in the course slides, and need to be acquired largely through self-study.

Solutions for selected computer exercises are discussed during the theory lectures so that students learn how computational skills can be applied in practice.

During the supervised hands-on sessions, students themselves learn how to tackle computational challenges by working on a series of mandatory computer exercises that need to be solved independently.

Dodona, a digital learning environment, gives students instant feedback on their solutions submitted for the Python programming challenges, containing additional exercises for further practicing.

After each deadline, the solutions of all exercises are made available on Minerva.

Announcements on Minerva are used for counselling, giving feedback, and providing background information.

The lecturer and the teaching assistants have weekly office hours for answering questions about the course in general, the theory, and the exercises. Individual

(Approved)
appointments can also be scheduled via email.

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
Open book examination, skills test

Examination methods in case of permanent evaluation
Assignment

Possibilities of retake in case of permanent evaluation
examination during the second examination period is not possible

Calculation of the examination mark

During the first examination period, the score of the periodic evaluation (partial examinations) accounts for 75% of the examination mark and the score of the non-periodic evaluation (hands-on sessions) accounts for 25% of the examination mark. 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 examination mark for the first examination period will be subject to an upper limit of 7/20.

The periodic evaluation consists of a partial examination at the end of the first term and one or two partial examinations at the end of the second term. The partial examination at the end of the first term accounts for 75% of the final score obtained at the end of the first term, and the hands-on sessions that took place during the first term account for 25% of the final score obtained at the end of the first term. To qualify for passing, both the score of the partial examination at the end of the first term and the score of the hands-on sessions at the end of the first term should be at least equal to 8/20. If that is not the case, the final score obtained at the end of the first term will be subject to an upper limit of 7/20.

If the final score obtained at the end of the first term is higher than or equal to 10/20, then one partial examination is used at the end of the second term, covering the course content of the second term. In addition, the score of the periodic evaluation is then equal to the average of the score of the partial examination at the end of the first term and the score of the partial examination at the end of the second term. On a similar note, the score of the non-periodic evaluation is then equal to the average of the score of the hands-on sessions at the end of the first term and the score of the hands-on sessions at the end of the second term.

If the final score obtained at the end of the first term is lower than 10/20, then two partial examinations are used at the end of the second term: a first partial examination covering the content of the first term and a second partial examination covering the content of the second term. In addition, the score of the periodic evaluation is then equal to the average of the scores obtained for the two partial examinations at the end of the second term, whereas the score of the non-periodic evaluation is then equal to the average of the score of the hands-on sessions at the end of the first term and the score of the hands-on sessions at the end of the second term. During the second term, the hands-on sessions of the first term cannot be retaken.

Students who passed the partial examination at the end of the first term may decide to retake this partial examination at the end of the second term. When doing so, the last score obtained is used for calculating the score of the periodic evaluation.

The second examination period makes use of a periodic evaluation that always consists of two partial examinations: a first partial examination covering the content of the first term and a second partial examination covering the content of the second term. During the second examination period, the non-periodic evaluation cannot be retaken. Therefore, the examination mark for the second examination period is calculated twice. For the first calculation, the score of the non-periodic evaluation, as obtained during the first examination period, accounts for 25% of the examination mark and the score of the periodic evaluation, as obtained during the second examination period, accounts for the remaining 75% of the examination mark. For the second calculation, the examination mark is equal to the score of the periodic evaluation, as obtained during the second examination period. The final examination mark for the second examination period is then equal to the maximum of the above two calculations.
Scores for partial examinations that took place in the first examination period can never be transferred to the second examination period.

Students who eschew period aligned and/or non-period aligned evaluations for this course unit may be failed by the examiner.