GHENT
UNIVERSITY

Course Specifications

Valid as from the academic year 2024-2025

Scientific Computing (I002908)

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

Credits 4.0	**Study time 120 h**

Course offerings and teaching methods in academic year 2025-2026

A (semester 1)	Dutch	Gent	lecture	25.0h
			seminar	15.0h

Lecturers in academic year 2025-2026

Köse, Demir Ali	LA26	staff member
Verwaeren, Jan	LA26	lecturer-in-charge

Offered in the following programmes in 2025-2026	**crdts**	**offering**
Bachelor of Science in Bioscience Engineering	4	A

Teaching languages

Dutch

Keywords

Scientific computing, programming, Python

Position of the course

Bio-engineers often face scientific problems that require an advanced and/or automated processing of information, or problems that require extensive computations. Often, the mere size of the computations involved impedes the use of methods relying on manual computation. To be able to solve these problems, skills that allow an engineer to: (1) analyze a problem and propose a structured solution strategy; and (2) transform this strategy into a computer program (using a programming language) are crucial in modern-day engineering. In this course, student learn the principles of structured, modular programming and learn to apply these principles to solve problems in a time-efficient and automated manner. Aditionally, this course focuses on the translation of a task (presented in a natural language) into a
sequence of instructions that can be executed by a computer. The programming language Python will be used in this course.

Contents

This course is an introductory course to programming in Python, in which attention is given to programming skills that are important in an engineering environment. The student is introduced to the elementary programming concepts, the syntax and semantics of the Python programming language. Moreover, the student learns how to develop a solution strategy for a given problem and how to translate it into source code. In this course, the students are introduced to:

- Basic components of Python: data types, variables and operators
- Arithmetic and logical operations
- Control flow (conditional and repetitive execution)
- Use and design of functions
- String processing
- Built-in data structures: lists, dictionaries (and to some extent tuples and sets)
- Homogeneous arrays and vectorization (using numpy)
- Data visualization (using matplotlib)

Through a number of integrating PC-labs, the student also learns to use the acquired programming skills to solve problems that are common in engineering for the life sciences.

Initial competences

No initial (programming) competences are required
Basic skills w.r.t. modern-day use of a personal computer are recommended

Final competences

1 Translate a well-defined task, described in a natural language, into source code in the programming language Python (correct use of syntax, data structures, control, functions and scripts).
2 Design (and implement) a tailor-made algorithm to solve a specific (simple) problem.
3 Interpret, test and debug a given source code.
4 Have insight into the use of Python as a scientific programming environment.
5 Apply vectorization as a strategy to write efficient code
6 Use Python for data visualization (plotting)

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

Seminar, Lecture

Extra information on the teaching methods

- Plenary lectures (12 h) and practical exercises (30 h, PC-labs).
- The students will complete a group work (4 students per group). Each group submits a written report, the submission is followed by an oral defense. This group work is part of the permanent evaluation process.

Study material

Type: Syllabus

Name: Course notes Scientific Programming
Indicative price: Free or paid by faculty
Optional: no
Language : Dutch
Available on Ufora : Yes
Online Available : Yes
Available in the Library : No
Available through Student Association : Yes

Type: Slides

Name: Lecture slides Scientific Programming
Indicative price: Free or paid by faculty
Optional: no
Language : Dutch
Available on Ufora : Yes
Online Available : Yes

Type: Software

Name: Python interpreter and VSCode
Indicative price: Free or paid by faculty
Optional: no
Available on Athena : No
Online Available : Yes

References

H.P. Langtangen (2009), A primer on Scientific Programming with Python. Springer, 686p.
W. Punch and R. Enbody, 2nd edition (2013), The Practice of Computing using Python,
Pearson, 764p.

Course content-related study coaching

- The practical course notes allow students to complete the PC-labs at their own pace. Teaching assistants are available to assist the students during the PC-labs.

- Students have are stimulated to ask additional questions after each plenary lecture or PC-lab. Alternatively, an appointment can be made with one of the lecturers.
- Students are taught how to install a Python interpreter and IDE on their personal laptop.

Assessment moments

end-of-term and continuous assessment

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

Written assessment

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

Written assessment

Examination methods in case of permanent assessment

Oral assessment, Assignment

Possibilities of retake in case of permanent assessment

examination during the second examination period is possible

Extra information on the examination methods

- The end-of-term evaluation consists of an open book exam (17 out of 20 points) that consists of a number of hands-on programming exercises. A mark is given based on the correctness and style of the source code that is submitted.
- The permanent evaluation (3 out of 20 points) is based on the evaluation of a group work. The students need to complete a (small) programming project (4 students per group), submit a report and source code. The report and the source code provide the basis of an in-depth discussion in the format of an oral exam. Each student is evaluated independently.
- The submission deadline for the group work and the date of the oral exam will be communicated to the students at the start of the semester (the oral exam takes place around the 8th week of the semester).

Calculation of the examination mark

- End-of-term evaluation (17 out of 20 points): evaluation by means of an open book exam. Submitted source code is corrected and a mark is given.
- Permanent evaluation (3 out of 20 points): submitted source code and a written report (of the group work) provide the basis for a small oral exam. Students that fail to submit the source code or report, or do not participate at the oral exam get a mark of zero.
- The final mark is the sum of the scores obtained for the end-of-term and the permanent evaluation.
- In the second examination period, students are given the opportunity to re-submit the project that is part of the permanent evaluation (task). A new mark will be given based on an (individual) oral defense.