GHENT UNIVERSITY

Course Specifications

Valid as from the academic year 2024-2025

Programming (C003080)

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

Credits 5.0 **Study time 150 h**

Course offerings and teaching methods in academic year 2025-2026

A (semester 1)	Dutch	Gent	lecture seminar
C (semester 1)	English	Gent	lecture seminar

Lecturers in academic year 2025-2026

Dawyndt, Peter	WE02	lecturer-in-charge

Offered in the following programmes in 2025-2026	crdts	offering
Bachelor of Science in Psychology(main subject Theoretical and Experimental Psychology)	5	A, C
Bachelor of Arts in Archaeology	5	C
Bachelor of Science in Biochemistry and Biotechnology	5	A
Bachelor of Science in Biology	5	A
Bachelor of Science in Chemistry	5	A
Bachelor of Science in Engineering: Architecture	5	A
Bachelor of Science in Geography and Geomatics	5	A
Bachelor of Science in Geology	5	A
Bachelor of Science in Physics and Astronomy	5	A
Master of Science in Teaching in Science and Technology(main subject Biology)	5	C
Master of Science in Teaching in Social Sciences(main subject Communication Science)	5	A
Master of Science in Complementary Studies in Business Economics(main subject Business Economics)	5	C
Master of Science in Communication Science(main subject Journalism)	5	A
Master of Science in Communication Science(main subject New Media and Society)	5	A
Master of Science in Biology	5	C
Master of Science in Complementary Studies in Economics	5	A, C
Master of Science in Engineering: Architecture	5	C
Master of Science in Pharmaceutical Engineering	5	C
Master of Science in Statistical Data Analysis	5	C
Exchange programme Faculty of Sciences (bachelor's level)	5	C
Exchange Programme in Biology (master's level)	5	C
Exchange Programme in Mathematics (master's level)	5	C
Exchange Programme in Political and Social Sciences	5	C
Linking Course Master of Science in Chemistry(main subject (Bio)Organic and Polymer Chemistry)	5	A
Linking Course Master of Science in Chemistry(main subject Analytical and Environmental Chemistry)	5	A
Linking Course Master of Science in Chemistry(main subject Materials and Nano Chemistry)	5	A
Linking Course Master of Science in Bioinformatics	5	A
Linking Course Master of Science in Geography and Geomatics	5	A
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C

Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Ghent University Elective Courses	5	A, C
Preparatory Course Master of Science in Chemistry(main subject Analytical and Environmental Chemistry)	5	A
Preparatory Course Master of Science in Biology	5	A
Preparatory Course Master of Science in Business Engineering	5	C
Preparatory Course Master of Science in Geography and Geomatics	5	A

Teaching languages

English, Dutch

Keywords

programming, problem solving, Python programming language

Position of the course

Researchers are often confronted with time-consuming and repetitive tasks when processing and analyzing information using computers, namely, collecting data from websites, converting files from one format into another, and analyzing, summarizing and visualizing the obtained data. The exponential flow of newly incoming information requires modern researchers to be able to automate these tasks, in order to speed up their daily routine jobs. This course teaches you how to translate these time-consuming and repetitive tasks in such a way that these can be performed automatically by the computer. The necessary programming skills for that purpose will be acquired by learning to work and think in the programming language Python.

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, in writing computer programs one must learn how to think as a programmer. This computational thinking process, or in other words, learning the skill of problem solving by programming, is underlined throughout the whole course. The programming language Python is used in particular to solve problems in terms of

- basic components: instructions, variables, data types and operators
- control structures: conditional tasks, control loops and functions
- data structures: strings, lists, tuples, dictionaries, sets, files and modules
- object oriented programming: objects, classes, attributes, methods, inheritance, polymorphism and exceptions

Initial competences

Some basic computer knowledge is advantageous. Prior programming skills are not required at all.

Final competences

1 Translate a task described in natural language into a program in the Python programming language and have this program being executed by a computer in order to generate a correct result.
2 Test and debug a program.
3 Make the right choices between different alternatives when implementing a program, taking into account performance, coding style and correctness.
4 Have a working knowledge about the basis principles of object oriented programming.

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, Independent work

Extra information on the teaching methods

Interactive lectures and guided hands-on sessions in a PC-room. Use of the electronic learning environment Dodona (dodona.ugent.be) for automatic evaluation of programming exercises. Interactive coaching (between students themselves or between student and lecturer) is stimulated by making use of the electronic learning environment Ufora (ufora.ugent.be).

Study material

Type: Slides

Name: Slides shown during the lectures.
Indicative price: Free or paid by faculty
Optional: no
Language : Other
Available on Ufora : Yes
Online Available : Yes
Available in the Library : No
Available through Student Association : No
Additional information: Slides are available both in English and Dutch.

References

- William Punch, Richard Enbody, The Practice of Computing using Python (global edition). Pearson, ISBN-13: 978-1292166629.
- 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 Eduction - 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.

Course content-related study coaching

Syntax and semantics of the programming language Python are presented in the course book and the slides, and need to be acquired largely through self-study. Some programming exercises are solved during the lectures to that students learn how the programming skills can be applied in practice. During hands-on sessions, students learn themselves how to solve programming challenges by working on a series of mandatory exercises, that need to be solved independently. The online learning environment Dodona gives students instant feedback on their submitted programming exercises and contains additional exercises to practice further. Use of the Q&A module in Dodona and the discussion forum in Ufora is stimulated, allowing students, the lecturer and his assistants to solve certain problems outside contact hours. Permanent evaluation of mandatory exercises and evaluation exercises will be used as feedback moment, and after the weekly deadlines, solutions of all exercises will be made available.

Assessment moments

end-of-term and continuous assessment

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

Skills test

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

Skills test

Examination methods in case of permanent assessment

Skills test

Possibilities of retake in case of permanent assessment

examination during the second examination period is possible

Extra information on the examination methods

For the **first part of the permanent evaluation**, the students have to work on a series of 60 mandatory exercises. Based on the covered programming techniques, these exercises are subdivided into 10 series of 6 mandatory exercises each. The first exercise of each series always is a variation on the manipulation of ISBN numbers. A sample solution of this exercise is available from Ufora, and in an accompanying instruction video we explain how we came to this sample solution. As such, the ISBN exercises explain how the new programming technique from the series can be brought into practice. After these preparatory steps, students are ready to apply the new programming technique themselves in solving the other five programming exercises from the series. Students must submit their solutions of the mandatory exercises in each series (including the ISBN exercise) through the online learning environment Dodona before a set deadline (deadlines always fall at 22:00 on the Tuesday that follows the hands-on session dedicated to the series of exercises). Dodona gives students a nice overview of the mandatory exercises in each series, the submission deadlines and the current status of each exercise.

For the **second part of the permanent evaluation**, we organise two evaluation sessions during the hands-on sessions that follow after five exercise series. During these evaluations, students have to solve two new programming exercises within the time frame of two hours. It is allowed to make use of the Dodona platform during the evaluations, so that students can check the correctness of their solutions at all times. The submitted solutions for the evaluation exercises are manually evaluated by the lecturer or the teaching assistants and scored based on correctness, programming style used, choice made between the use of different programming techniques and the overall quality of the solution. The level of difficulty of the evaluation exercises is lower than those of the exercises that need to be solved during the periodic evaluation (exam), as we mainly want to check at this time during the semester if the students master the basic programming skills. In addition, these evaluation sessions follow the same procedure that is also used during the periodic evaluations, so that students can use this experience to adjust their approach towards the exam.

The **score for the permanent evaluation** is determined using the formula s * c / a, where s is the score a student obtains based on his submitted solutions for the evaluation exercises (expressed as a score out of 20), c is the number of mandatory exercices for which at least one correct solutions has been submitted before the weekly deadlines, and a is the total number of mandatory exercises (30 per evaluation series). A student that for example has obtained a score of 16/20 for his evaluation exercises and that has submitted correct solutions for all 30 mandatory exercises before the weekly deadlines, obtains a score of 16 * 30/30 = 16 out of 20 for the evaluation series. If that student still had obtained a score of 16/20 for his evaluation exercises, but only submitted 18/30 correct solutions for the mandatory exercises before the weekly deadlines, he sees his score for the evaluation series reduced to 16 * 18/30 = 9.6 out of 20.

Students will receive an email with their score for the evaluation series as soon as possible after each evaluation session. During the next hands-on session, students can collect their submitted solutions that will have been annotated with feedback that indicates where they can improve their code. They can use this feedback in solving other programming exercises.

During the **periodic evaluation (exam)** students are given 3.5 hours to solve three programming exercises. It is allowed to make use of the Dodona platform during the periodic evaluation, so that students can check the correctness of their solutions at all times and receive immediate feedback on the correctness and the programming style of their submitted solutions. To determine the score for the

periodic solution, the submitted solutions are manually evaluated by the lecturer or the teaching assistants and scored based on correctness, programming style used, choice made between the use of different programming techniques and the overall quality of the solution.

Calculation of the examination mark

In computing the examination mark we take into account both permanent evaluation (20%, 4/20) and periodic evaluation (80%, 16/20). The permanent evaluation has two components that both influence the score for the permanent evaluation.

To compute the examination mark we compute two scores. One score takes into account both the score for the permanent evaluation (weight 20%) and the periodic evaluation (weight 80%). The other score ignores the score obtained for the permanent evaluation and is only based on the score for the periodic evaluation. The final examination mark is the maximum of these two scores.