GHENT UNIVERSITY

Course Specifications

Valid as from the academic year 2024-2025

Scripting Languages (C002178)

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

Credits 6.0 **Study time 180 h**

Course offerings and teaching methods in academic year 2024-2025

A (semester 2)	Dutch	Gent	seminar
			lecture

Lecturers in academic year 2024-2025

Dawyndt, Peter	WE02	lecturer-in-charge

Offered in the following programmes in 2024-2025	**crdts**	**offering**
Bachelor of Science in Computer Science	6	A

Teaching languages

Dutch

Keywords

python, javascript, scripting languages, dynamic typing

Position of the course

Scripting languages support scripts: programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted rather than compiled. Primitives are usually the elementary tasks or API calls, and the language allows them to be combined into more complex programs. Environments that can be automated through scripting include software applications, web pages within a web browser, the shells of operating systems (OS), embedded systems, as well as numerous games. As a result, scripting languages can be viewed as domain-specific languages for a particular environment and in the specific case of scripting an application they are also known as extension languages.

The spectrum of scripting languages ranges from very small and highly domain-specific languages to general-purpose programming languages used for scripting. Standard examples of scripting languages for specific environments include: Bash (for the Unix or Unix-like operating systems), ECMAScript/JavaScript (for web browsers) and Visual Basic for Applications (for Microsoft Office applications). Python is a general-purpose language that is also commonly used as an extension language, while ECMAScript/JavaScript is still primarily a scripting language for web browsers, but is also used as a general-purpose language.

Contents

You explore the possibilities of scripting languages by learning Python and JavaScript (ECMAScript). These two scripting languages use dynamic typing, in contrast for example to the programming language Java that uses static typing. In addition, you also experience how Python and JavaScript combine ideas from object-oriented programming with ideas from functional programming and event-driven programming in one and the samen language. Apart from learning the scripting languages themselves, you also learn how they can be applied to automate tasks in specific environments.

Initial competences

Proven programming experience in at least one procedural or object-oriented programming language (e.g. C, C++ or Java).

Final competences

1 Master the Python scripting language.
2 Apply the Python scripting language for automating tasks in a specific environment.

3 Master the JavaScript (ECMAScript) scripting language.
4 Apply the JavaScript (ECMAScript) scripting language for automating tasks in a browser.
5 Implement a client-server application that uses server-side Python and client-side JavaScript (browser).

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

Electronic learning environment Ufora is used to encourage individual contributions of the students and to disseminate background material and pointers to alternative scripting languages. Assignments are distributed via GitHub (github.ugent.be), can be worked out on the interactive Linux environment Helios (helios.ugent.be) and must be submitted to Indianio (indianio.ugent.be). Automated feedback on programming assignments is given immediately by using the interactive learning environments Pythia (pythia.ugent.be; Python) and Dodona (dodona.ugent.be; JavaScript).

Study material

Type: Slides

Name: Slides shown during the lectures.
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 : No
Additional information: The course only makes use of tutorials that are freely available online, together with video tutorials and tips & tricks for specific programming assignments. Miller B, Ranim D. How to Think Like a Computer Scientist (freely available online). Crockford D. Crockford on JavaScript (lecture series freely available online). MDN JavaScript guide (freely available online). Bootstrap tutorial (freely available online).

References

Crockford D (2008). JavaScript: The Good Parts. O'Reilly Media. ISBN 978-0-596-51774-8.

Flanagan D (2010). jQuery Pocket Reference. O'Reilly Media. ISBN 978-1-4493-9722-7.

Course content-related study coaching

Through a combination of classroom lectures and computer seminars, the student gains insight in the usage of different scripting languages. He or she is stimulated to practice these languages by means of a series of given automation problems. Solutions to exercises and tasks are evaluated during computer seminars. Consultation with lecturer or one of his assistants by email appointment gives the possibility of additional explication on an individual basis. Interactive coaching (among students and between students and the lecturer) is encouraged by making use of the electronic learning environment Ufora. Automated feedback on programming assignments is given immediately by using the interactive learning environments Pythia (pythia.ugent.be; Python) and Dodona (dodona.ugent.be; JavaScript).

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, Assignment

Possibilities of retake in case of permanent assessment

examination during the second examination period is not possible

Extra information on the examination methods

In computing the final score we take into account both the permanent evaluations (35%, 7/20) and the periodic evaluation (65%, 13/20). There are three permanent evaluation that together make up the score for the permanent evaluations.

For the first permanent evaluation (Python), the students have to work on a series of 44 mandatory exercises. Based on the covered programming techniques, these exercises are subdivided into 10 series of mandatory exercises. 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 practive. After this 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 Pythia before a set deadline (deadlines always fall at 22:00 on the Friday that follows the hands-on session dedicated to the series of exercises). Students can use the Pythia scoresheet to get an overview of the exercises for which they have already submitted a correct solution. The scoresheet provides a nice overview of the mandatory exercises in each series, the submission deadlines, the current status of each exercises, and the time a first correct solution was submitted for the exercise.

The first permanent evaluation (Python) ends with an evaluation session during the hands-on sessions that follows the submission deadline of the last exercise series. During this evaluations, students have to solve two new Python programming assignments within the time frame of two hours. These exercises are in line with the mandatory exercises the students had to solve during the hands-on sessions. The submitted solutions for the evaluation exercises are manually evaluated by the lecturer or the teaching assistants and scored based on both correctness and the overall quality of the solution. The level of difficulty of the evaluation exercises corresponds to the the level of the assignments that need to be solved during the periodic evaluation (exam). In addition, this evaluation session follows 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.

For the second permanent evaluation (JavaScript), the students have to work on a series of mandatory exercises. This allows students to bring their JavaScript knowledge and skills into practice. Students must submit their solutions of the mandatory exercises through the online learning environment Indianio before set deadlines (deadlines always fall at 22:00 on the Friday that follows the hands-on session dedicated to the series of exercises). Students can use the electronic learning environment Dodona to test the correctness of their submitted solutions and get immediate and automated feedback on their submitted solutions.

The second permanent evaluation (JavaScript) ends with an evaluation session during the hands-on sessions that follows the submission deadline of the last exercise series. During this evaluations, students have to solve two new JavaScript programming assignments within the time frame of two hours. These exercises are in line with the mandatory exercises the students had to solve during the hands-on sessions. The submitted solutions for the evaluation exercises are manually evaluated by the lecturer or the teaching assistants and scored based on both correctness and the overall quality of the solution. The level of difficulty of the evaluation exercises corresponds to the the level of the assignments that need to be solved during the periodic evaluation (exam). In addition, this evaluation session follows 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 third permanent evaluation is an assignment in which the students have to implement a client-server application that uses server-side Python and client-side JavaScript (browser). The solution of this assignment must be submitted before a set deadline, and is evaluated based on correctness, choices made in designing and implementing the application, and user-friendliness of the application.

The score for the permanent evaluations is determined using the formula s * c / a, where s is the score a student has obtains based on his submitted solutions for the evaluation exercises (expressed as a score out of 20), c is the number of mandatory exerices for which at least one

correct solutions has been submitted before the weekly deadlines, and a is the total number of mandatory exercises. A student that for example has obtained a score of 16/20 for his evaluation exercises and that has submitted correct solutions for all 44 mandatory exercises before the weekly deadlines, obtains a score of 16 * 44/44 = 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 30/44 correct solutions for the mandatory exercises before the weekly deadlines, he sees his score for the evaluation series reduced to 16 * 30/44 = 10.9 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 exercises.

It is not possible to retake the permanent evaluation during the second examination period. To compute the score for the second examination period, we compute two scores. One score takes into account the score for the permanent evaluations (with weight 35%, as was done during the first examination period). The other score ignores the score obtains for the permanent evaluations, and is only based on the score for the periodic evaluation. The final score for the second examination period is the maximum of these two scores.

During the periodic evaluation (exam) students are given four hours to solve some two Python programming assignments and two JavaScript programming assignments. These assignments are in line with the mandatory exercises the students had to solve during the hands-on sessions and the evaluation assignments. 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 both correctness and the overall quality of the solution.

Calculation of the examination mark

In computing the final score we take into account both the permanent evaluations (35%, 7/20) and the periodic evaluation (65%, 13/20). There are three permanent evaluation that together make up the score for the permanent evaluations.

It is not possible to retake the permanent evaluation during the second examination period. To compute the score for the second examination period, we compute two scores. One score takes into account the score for the permanent evaluations (with weight 35%, as was done during the first examination period). The other score ignores the score obtains for the permanent evaluations, and is only based on the score for the periodic evaluation. The final score for the second examination period is the maximum of these two scores.