Course Information: Programming in Python

General information (2020/2021 period 2)

Course name:	Programming in Python
Code:	INF-22306
Credits:	6 ECTS (168 hours)
Language:	English
Schedule:	all mornings during the six weeks of period 1, some in week 7. • Lectures on Mondays (except first week: Tuesday: pre-recorded • Supervised study (Class exercises) on Tuesdays (except week 1) and Thursdays, several groups: online in Brightspace VC. • Computer labs on Mondays, Wednesdays, and Fridays and on one Tuesday (week1), several groups: online using Discord. Details: see schedule on Brightspace
Contact:	**Please mail only to** inf-22306@wur.nl
Coordinator:	ir. M.A. Zijp, Leeuwenborch room 6026
Examiner:	ir. M.A. Zijp
Lecturers:	drs. M.R. Kramer, dr. T. Alskaif, dr. S.A. Osinga, ir. M.A. Zijp, dr. ir. A. Kassahun
Other staff involved:	ir. G. Moerland

Keywords

Programming, algorithms, Python (programming language), software libraries

Profile of the course

Software plays an important role in many domains. Very often, scientists are writing or adapting computer programs to process or analyze their data and present their results in a suitable format (e.g. on the Internet). This course does not aim to produce professional programmers, but rather to build understanding of the underlying principles and equip future academics with basic skills to create computer programs for small-scale use. The same principles are needed for writing custom code in many simulation, modeling, and engineering tools.
The programming language Python serves a broad application domain ranging from short scripts to full-blown software systems (e.g. Google uses Python). The course gives an introduction to libraries of available components, and how to use these for building your own software.

Learning outcomes

After the course, students should be able to:

- implement a given algorithm as a computer program (in Python);
- adapt and combine standard algorithms to solve a given problem (includes numerical as well as non-numerical algorithms);
- apply standard programming constructs for a given goal: repetition, selection, functions, composition, modules, aggregated data (arrays, lists, etc.);
- explain what a given piece of programming code (in Python) does;
- identify and repair coding errors in a given piece of programming code;
- demonstrate how to apply object based software concepts (constructing OO software will be dealt with in the course Software Engineering);
- demonstrate how to apply classes and functions from library software used during the course for (e.g.) building a graphical user interface or data visualization.

Study materials

Textbook:
Think Python, how to think like a computer scientist, by Allen B. Downey, available on-line at : https://greenteapress.com/wp/think-python-2e/

Also available as printed book: 2nd edition by O'Reilly, ISBN: 978-1-491-93936-9. There will be a number of copies available in the WURshop (StudyStore) for about 39 euro.

Software:

- PyCharm and Anaconda: installation instructions on Brightspace
- Discord: installation instructions on Brightspace

Materials on Brightspace:

- Tutorials on the software environment
- Short introductions to some Python libraries not covered in the book
- Reading guide to the book
- Additional exercises and assignments
- Lecture slides
- Results of Class Exercises
- Example exam with answers

Activities

A major part of the course consists of computer lessons. This is fully online with support using Discord (see the instructions on Brightspace). Students start with tutorials on Python and the software environment used for this course (PyCharm). From the first introduction, your own computer is used for assignments. Further tutorials and exercises must be done at home (also pre-Corona).
Assignments during computer labs are small in the first week, and gradually increase during the course, to assignments similar to exam questions, and occasionally more complicated assignments at the end of the course.
During the first session of class exercises (in Brightspace's Virtual Classroom), we will illustrate some aspects of the software environment used for this course. Later class exercises will focus on the concepts of programming and recurring programming structures. Students have to write (small) programs on paper first, and then a teacher will demonstrate how to build up the program by means of explicit input from students' work.
Assignments and class exercises focus on learning how to program. Together they cover the learning outcomes. Exercises (to be done independently at home) focus more on elements of the Python language and particular libraries. Often, those exercises prepare for using a technique in the assignments. Many exercises are strongly linked with the book. Students are strongly encouraged to study the book along with these exercises.
Assignments during the online computer labs are not graded. Students can ask for feedback during the sessions in Discord. If students encounter difficulties with exercises or tutorials, they can also ask their questions during computer labs in Discord or in Discord during catch-up sessions. There are also possibilities to meet 1 to 1 with a student assistant or staff member in a special room with a large screen keeping 1.5-meter distance. Details on Brightspace.
Feedback on class exercises is incorporated in the class exercise sessions.

The emphasis of the lectures is on illustrating key topics in programming, as well as zooming into some details of the language. Attending the lectures does not substitute for studying the book, however.

Assessment

For formal assessment we use a written exam (closed book). The exam consists of fill-in the

blanks and open questions, both programming and explaining/correcting problems in a given piece of program. A written exam for a programming course may seem strange at first, but the key issues of programming are no different on paper than at the computer screen. Moreover, during a written exam, candidates are not distracted by small errors in the details: at the screen they may cost hours, on paper they might cost one tenth of a point – if anything at all.
Assignments and exercises during the course will not be graded. Students can ask student-assistants to check their programs during the hours scheduled for assignments. Also, a selection of assignments is discussed explicitly during class exercises. Together, the assignments cover the learning outcomes. A student who has completed all assignments should be confident about the exam result.

Principal themes

Programming is about manipulating data in some form. Even for the most elementary operations, data has to be stored in the memory of the computer. The terms **values**, **variables**, **types**, **expressions and operators** give a more precise view on storing and manipulating data. **Functions** are the key building blocks of computer programs. They are, usually small, pieces of program code that perform one coherent task, e.g. computing a value from some other values.

The term **control structures** denotes all programming constructs that help to repeat or skip some parts of code depending on values of variables. One aspect is how to write such control structures. Another, much more important aspect is how to use such structures appropriately.

Software libraries (modules). Modern programming languages consist of a relatively small core and many additional parts. Those additional parts are called libraries or (especially in Python) modules. The course teaches how to use a number of the modules that come with Python, and also how to define your own modules.

Structured types. For dealing with large amounts of data (in memory rather than in files), programming languages use structured data often referred to as arrays. In Python, even more powerful data types (lists, tuples, dictionaries) are integrated in the language.

File processing. Data in computer memory is present for as long as the program that handles the data is running. Files are the primary means to store data outside memory. The techniques for reading and writing files are collectively called file processing.

In current software engineering practice, **testing** each program part is an indispensable aspect. The course pays attention to how to develop small tests.

Outline of the course

- Week 1: Introduction to Python and the programming environment. Introduction to values (numeric as well as string), variables, types, expressions and operators, functions, and control structures.
- Week 2: More on choices and loops (also loop design), functions. Introduction to indexing and slicing.
- Week 3: Structured types. Changing the content of lists (arrays). Mutability and aliasing. Python specific types: tuples and dictionaries, debugging.
- Week 4: File processing, composing (formatting) output, and decomposing (parsing) input and testing.
- Week 5: Defining custom structured types (classes). Introduction to graphical user interfaces (GUIs).
- Week 6+7: Advanced techniques, including visualization of data.

Schedule

See the complete schedule on Brightspace