Course Information: Programming in Python

General information

Course name:	Programming in Python
Code:	INF-22306
Credits:	6 ECTS (168 hours)
Language:	English
Schedule (period 1):	all afternoons during first six weeks of period 1, plus (as part of the pilot) Wednesday evenings 18:15–20:00 (weeks 2–6) • lectures on Wednesday 13:30–15:15 (week 1) and Monday 13:30–15:15 (weeks 2–6) • supervised study (class exercises) Tuesday 15:30–17:15 (weeks 2–6) and Thursday 13:30–15:15 (weeks 2–6) • rest of the time computer labs (except Wednesdays 13:30–15:15) interim exam: Friday October 28, 13:30–16:30 [This course is also offered in period 2 in the mornings.]
Coordinator:	ir. M.A. Zijp, Leeuwenborch room 6026, tel.: 484079
Examiner/Lecturer:	drs. M.R. Kramer
Other staff involved:	ir. A. Kassahun, ir. G. Moerland, dr S.A. Osinga

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)
- adequately use standard programming constructs: repetition, selection, functions, composition, modules, aggregated data (arrays, lists, etc.)
- explain what a given program (in Python) does
- identify and repair coding errors in a program
- understand and use object based software concepts (constructing OO software will be dealt with in the course Software Engineering)
- use library software for (e.g.) building a graphical user interface, web application, or mathematical software

Study materials

Textbook:

- Sarah Mount, James Shuttleworth and Russel Winder: “Python for Rookies” (ISBN 978-1-84480-701-7) [we use 10 out of 13 chapters]

Software:

- Python(x,y) including IDLE [free software, pre-installed in the PC rooms]

Materials on Blackboard:

- Tutorials on the software environment (IDLE)
- Short introductions to some Python libraries not covered in the book
- Reading guide to the book
- Additional exercises and assignments – and pointers to assignments from the book
- Lecture slides
- Example exam with answers

Activities

A major part of the course consists of computer lessons.
Students start with tutorials on Python and IDLE (software environment for Python), with embedded exercises.
After the first few days, work alternates in blocks of two hours between programming assignments and exercises (and tutorials). While one or more groups continue tutorials and exercises (later only exercises), the other group(s) do programming assignments. In the next block, the groups switch activities: the group(s) that started at assignments, work on tutorials and exercises again, and vice versa.
From the second week onwards, we schedule two blocks of class exercises – two hours per session. The emphasis of these sessions is 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 proper (during computer lessons) 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 encouraged to study the book along with these exercises. One slot of two hours per week is especially dedicated to studying the book.
To provide for enough assistance and feedback on assignments, more staff is available during assignments blocks (PI) than during exercises blocks (PE). Feedback on class exercises is incorporated in the class exercise sessions.
The emphasis of the lectures, on Mondays (except the first week on Wednesday), 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 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 staff 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.
Nowadays, a program without a **graphical user interface** (GUI) is almost unthinkable. The course shows how to use standard modules for building such GUIs.
In current software engineering practice, **testing** each program part is an indispensable aspect. The course pays attention to how to develop small tests in the context of **test driven development**.

Outline of the course

Week 1: Introduction to Python and IDLE. Introduction to values (numeric as well as string), variables, types, expressions and operators, functions, and control structures.
Week 2: More on choices, loops (also loop design), functions, and recursion. Introduction to indexing and slicing (on strings).
Week 3: Structured types. Changing the content of lists (arrays). Mutability and aliasing. Python specific types: tuples and dictionaries.
Week 4: More on defining functions. Defining modules. File processing, composing (formatting) output, and decomposing (parsing) input.
Week 5: Defining custom structured types (classes). Introduction to graphical user interfaces (GUIs).
Week 6: Testing and test driven development. More on GUIs, including threading.

Schedule

There are several groups. All groups start with the tutorials (see Blackboard) on Monday afternoon of the first week. After four blocks of two hours, groups alternate between assignments and tutorials/exercises. While one or more groups do assignments, other groups do exercises and vice versa. The last block (or for some groups the first block) on Friday afternoons is scheduled for studying the book.
From the second week onwards, class exercises are scheduled for the second half of Tuesday afternoons and the first half of Thursday afternoons.
Lectures are scheduled for the first half of Monday afternoons, except for the first week. The lecture for the first week is scheduled for the first half of Wednesday afternoon.
For rooms and schedules per group, please refer to information on Blackboard.
In 2016 period 1 this course is included in the pilot of evening classes, so some sessions of the assignments and tutorials/exercises are scheduled on Wednesday evenings, during weeks 2 through 6.