UNIVERSITY OF WARSAW

# Course syllabus

| | |
|---|---|
| Course title | Introduction to programming in Python |
| Instructor(s) | Marcin Lesniak, PhD |
| Contact details | marcin.lesniak@psych.uw.edu.pl |
| Affiliation | Faculty of Psychology, University of Warsaw |
| Course format | class |
| Number of hours | 30 hours |
| Number of ECTS credits | **3 ECTS credits** |
| Brief course description | This is an introductory course to computer programming in Python. It does not require any previous programming experience. In the class students learn to think algorithmically and decompose problems into manageable parts. Through simple examples they become familiar with programming concepts such as variables, conditional statements, loops, functions, recursion. Basic Python syntax, standard data structures and flow control statements are introduced. Procedural style of programming is encouraged. |
| Full course description | Python is a high-level, general-purpose programming language which gained vast popularity among scientific community. It is a popular tool of choice for scientific computing, data analysis and computational modeling. In this course it is taught as the first programming language to an audience without prior experience with computer programming.<br><br>Clear Python syntax helps the beginners to become acquainted with standard programming concepts—such as variables, conditional statements, loops, functions, recursion—without being distracted by implementation details. Standard data structures (list, dictionaries, sets) will be also introduced. Those form the necessary foundations for algorithmic thinking. As the class progresses, students should develop intuitions and become more familiar with this style of thinking, allowing them to express real-world scenarios in algorithmic terms. They should be able to solve simple problems on their own, creatively applying introduced concepts.<br><br>This basic course focuses on imperative/procedural style of programming, where code is organized in (active) procedures and (static) data structures. Notions of object-oriented programming or functional programming are generally not introduced at this point. No additional software packages besides Python standard library are covered. Python 3 version of the language is used in classroom. |

UNIVERSITY OF WARSAW

| | |
|---|---|
| Learning outcomes | K_W04, K_W08<br>K_U03, K_U05<br>K_K01, K_K02 |
| Learning activities and teaching methods | The class will be conducted in a computer laboratory. It will consist of programming exercises interspersed with short lectures and demonstrations. Short homework assignments and readings may be expected. |
| List of topics/classes and bibliography | Topics:<br>1. Ingredients.<br>Basic data types: integers, floats, booleans, strings. Arithmetic expressions, string indexing. Basic string and number manipulation methods. Variables and assignments.<br>2. Structures.<br>Sequences (lists, tuples, ranges). List indexing. Assignment to variable vs assignment to list element. Using dictionaries to make key-value mappings.<br>3. Flow control.<br>Conditional statements, bounded loop (for loop over elements of a sequence) vs unbounded loop (while loop). List and dictionary comprehension.<br>4. Functions.<br>Organizing code into functions. Parameters and returned value. Scope of variables. Nested function calls. Writing simple functions operating on text and numbers.<br>5. Programs.<br>Writing a simple interactive program (for example, very simple text-based computer game). Responding to user's input, dealing with malformed input.<br>6. Algorithms.<br>Introducing simple algorithms (Sieve of Eratosthenes, Euclid's algorithm, etc.). Reading and analyzing algorithms. Thinking algorithmically.<br>7. Style.<br>Good Python programming practices. Exceptions.<br><br>Suggested reading:<br>• Algorithmics: The Spirit of Computing. David Harel. 3rd edition, Addison-Wesley, 2004. Chapters 1-2 (for topics 1-2), chapter 4 (for topic 5), chapter 6 (for topics 6-7).<br>• Python Programming: An Introduction to Computer Science. John Zelle. 3rd edition, 2016. Chapter 13 (for topics 5-7).<br><br>Full bibliography:<br>• Algorithmics: The Spirit of Computing. David Harel. 3rd edition, Addison-Wesley, 2004.<br>• Python Programming: An Introduction to Computer Science. John Zelle. 3rd edition, 2016. |

| | |
|---|---|
| | • Python Crash Course, a hands-on, project-based introduction to programming, Eric Matthes, 2016.<br>• Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code, Zed Shaw, 2013. |
| Assessment methods and criteria | **In-class quizzes (100%)** These assessments, conducted at the start of each class, will encompass material from the prior class and assigned homework (practical programming tasks), ensuring a holistic evaluation of your understanding. The quizzes will include multiple-choice and short-answer questions.<br><br>The final grade will be determined using the following grading scale:<br>95% or more = 5!<br>90-94% = 5<br>80-89% = 4.5<br>70-79% = 4<br>60-69% = 3.5<br>50-59% = 3<br>below 50% = 2 (fail) |
| Attendance rules | One unexcused absence is allowed in the semester. Further unexcused absences may result in lowering the grade. If due to absences students miss graded exercises in class, they may be given an additional homework assignment. |
| Prerequisites | Fluency in high-school level mathematics and logic. |
| Academic honesty | Students must respect the principles of academic integrity. Cheating and plagiarism (including copying work from other students, internet or other sources) are serious violations that are punishable and instructors are required to report all cases to the administration. |
| Remarks | Tuesday 8:30-12:30, Stawki 5/7 |