

## SYLLABUS

### 1. Program details

1.1 Higher education institution	West University of Timișoara
1.2 Faculty / Department	Psychology and Educational Sciences
1.3 Department	Psychology
1.4 Field of study	Psychology
1.5 Cycle of studies	Bachelor's Degree
1.6 Study program / Qualification	Psychology – Cognitive Science

### 2. Discipline details

2.1 Discipline name	<b>Introduction in Programming</b>						
2.2 Tenured teacher - course activities	Gabriel Fischmann, Ph.D.						
2.3 Tenured teacher – seminar / laboratory activities	Gabriel Fischmann, Ph.D.						
2.4 Study year	1	2.5 Semester	1	2.6 Type of assessment	Exam	2.7 Discipline regime	DO
2.5 Google Classroom code	<b>axivrddr</b>						

### 3. Estimated total time (hours per semester) of teaching activities

3.1 Number of hours per semester	4	Of which: 3.2 course	2	3.3 seminar/laboratory	2
3.4 Total hours from the curriculum	56	Of which: 3.5 course	28	3.6 seminar/laboratory	28
Time fund distribution:					hours
Study based on the textbook, course material, bibliography, and notes					14
Preparing seminars/labs, homework, papers, portfolios, and essays					25
Examinations					4
Other activities					2
3.7 Total hours of individual study	45				
3.8 Total hours per semester	100				
3.9 Number of credits (ECTS)	4				

### 4. Prerequisites (where necessary)

4.1 for curriculum	• Not necessary
4.2 for competencies	• Not necessary

### 5. Conditions (where necessary)

5.1 for conducting the course	<ul style="list-style-type: none"> <li>• Audio-video equipment with Internet connectivity</li> <li>• Minimum required attendance 50%</li> <li>• Platform: <a href="https://classroom.google.com">https://classroom.google.com</a></li> </ul>
5.2 for conducting the seminar/laboratory	<ul style="list-style-type: none"> <li>• PCs with access to GitHub and Codespaces.</li> </ul>

	<ul style="list-style-type: none"> <li>• Student accounts for GitHub and Codespaces.</li> <li>• Minimum required attendance 50%</li> <li>• Platform: <a href="https://classroom.google.com">https://classroom.google.com</a></li> </ul>
--	---

## 6. Discipline objectives - expected learning outcomes to which the discipline's study and promotion contributes

Knowledge	<ul style="list-style-type: none"> <li>• To understand how computers work, from a high-level computer architecture viewpoint.</li> <li>• To understand the relationship between software and hardware.</li> <li>• To understand the basic elements of computer code: variables, constants, functions, conditionals, loops, and exceptions.</li> </ul>
Skills	<ul style="list-style-type: none"> <li>• To be able to write basic code in Python.</li> <li>• To be able to debug code in a modern IDE.</li> <li>• To be able to use GitHub Copilot AI for writing and improving code.</li> </ul>
Responsibility and autonomy	<ul style="list-style-type: none"> <li>• To work independently, as well as in groups, while coding, according to the assignment type.</li> <li>• To use available resources, such as publicly available code.</li> </ul>

## 7. Contents

7.1 Course	Teaching methods	Observations
1. Introduction.	Lecture, discussions	
2. Inside a computer: the hardware.	Lecture, discussions, exercises	White, Ron. <i>How Computers Work</i> . Chapter 4, How Motherboards Conduct a Symphony of Data.
3. Inside a computer: the operating system.	Lecture, discussions, exercises	White, Ron. <i>How Computers Work</i> . Part 2 – Overview (pp. 54-63).
4. Inside a computer: bits, bytes, files. Numeral systems.	Lecture, discussions, exercises	White, Ron. <i>How Computers Work</i> . Chapter 2, How Computers Remember.
5. Data and logic.	Lecture, discussions, exercises	Gutttag, John. <i>Introduction to Computation and Programming Using Python</i> . Chapter 2.2, The Basic Elements of Python.
6. Code flow: conditionals.	Lecture, discussions, exercises	Gutttag, John. <i>Introduction to Computation and Programming Using Python</i> . Chapter 2.3, Branching Programs.
7. Code reuse. Functions and parameters.	Lecture, discussions, exercises	Miles, Rob. <i>Begin to Code with Python</i> . Chapter 7, Using functions to simplify programs.
8. Code flow: loops.	Lecture, discussions, exercises	Gutttag, John. <i>Introduction to Computation and Programming Using Python</i> . Chapters 2.5, While Loops; and 2.6, For Loops and Range.
9. Code flow: exceptions.	Lecture, discussions, exercises	Gutttag, John. <i>Introduction to Computation and Programming Using Python</i> . Chapter 9, Exceptions and Assertions.
10. Libraries.	Lecture, discussions, exercises	Gutttag, John. <i>Introduction to Computation and Programming Using Python</i> . Chapters

		7.1, Modules; and 7.2, Using Predefined Packages.
11. Testing.	Lecture, discussions, exercises	Guttag, John. <i>Introduction to Computation and Programming Using Python</i> . Chapter 8, Testing and Debugging.
12. Data analysis and visualization.	Lecture, discussions, exercises	Guttag, John. <i>Introduction to Computation and Programming Using Python</i> . Chapters 13.1, Plotting Using Matplotlib; and 23, Exploring Data with Pandas.
13. Data structures: a preview	Lecture, discussions, exercises	
14. Recap	Discussions, exercises	
<p>References (all mandatory reading sections will be provided for each course in the online platform):</p> <ul style="list-style-type: none"> <li>• Guttag, John. <i>Introduction to Computation and Programming Using Python</i>. Third edition. MIT Press, 2021. ISBN: 9780262542364.</li> <li>• Miles, Rob. <i>Begin to Code with Python</i>. First edition. Microsoft press, 2017. ISBN: 9781509304523.</li> <li>• The Python Tutorial: <a href="https://docs.python.org/3/tutorial/">https://docs.python.org/3/tutorial/</a></li> <li>• White, Ron. <i>How Computers Work</i>. Tenth edition. 2014. ISBN: 9780789749840.</li> </ul>		
7.2 Seminar / laboratory	Teaching methods	Observations
1. Introduction. The software development environment.	Presentation, discussions	
2. Computer hardware.	Examples, demos, exercises	
3. The Windows operating system. The file system and file formats.	Examples, demos, exercises	
4. Working with GitHub Codespaces, and Copilot. First program.	Examples, demos, exercises	
5. Python data types. F-strings.	Examples, demos, exercises	
6. Conditionals.	Examples, demos, exercises	
7. Python functions. Passing parameters in Python.	Examples, demos, exercises	
8. Loops.	Examples, demos, exercises	
9. Exceptions.	Examples, demos, exercises	
10. Implementing tests.	Examples, demos, exercises	
11. End of term project start.	Examples, demos, exercises	
12. Data visualization in Python.	Examples, demos, exercises	
13. Recap.	Discussions	
14. Test.	Evaluation	
References:		

- The Python Language Reference: <https://docs.python.org/3/reference/>

## 8. Correlation of discipline contents with the expectations of the representatives of the epistemic community, professional associations and representative employers in the field related to the program

Psychology graduates are often expected to have programming experience to access jobs that require data analysis and interpretation. Since most Psychology students do not have any prior knowledge related to programming, this class will teach them the necessary basics. More advanced classes, such as *Object-Oriented Programming* (planned for the second year) will build on the knowledge acquired here, to enable students to access more diverse ways of meeting the expectations of the labor market. Learning to use AI tools, such as GitHub Copilot is essential in most jobs today, to help employees maintain the expected high level of productivity.

## 9. Assessment

Activity type	9.1 Assessment criteria	9.2 Assessment methods	9.3 Weight of final mark
9.4 Course	Acquired knowledge and proven understanding of the course topics.	Two multiple-choice exams.	50%
9.5 Seminar/laboratory	Assimilation and understanding of programming techniques taught in the lab, proven by the capacity to write a program in Python.	Writing a program, presenting, and explaining the written code.	50%
9.6 Minimum performance standard			
<p><b>Course:</b> Two 18-questions multiple-choice tests, one after course five, and one at the end of the semester, the first one covering the material for the first five courses (including <i>Data and Logic</i>), and the second one the rest.</p> <p>For each test, grading starts from 1, with 0.5 points awarded for each correct answer and no points deducted for wrong answers.</p> <p>The grade for the course is calculated as the average of the grades for the two tests.</p> <p>Students who do not attend either of the tests will have the opportunity to take these tests in Session B.</p> <p><b>Lab:</b> The task consists of writing and presenting a Python program, as specified. The student will write the code during their individual study time and will present it during the last lab of the semester.</p> <p>Grading starts from 1, and points are awarded as follows:</p> <ul style="list-style-type: none"> <li>- The program compiles and runs without errors and warnings. (1 point)</li> <li>- The program execution gives the expected outcome. (1 point)</li> <li>- The program contains at least one meaningful test for its correct functioning. (1 point)</li> <li>- The student correctly and promptly implements a modification to the program, as requested by the teacher. A maximum of 5 (five) minutes is allowed for implementing the change. (3 points)</li> <li>- The student provides correct and complete answers to questions about the written code. A maximum of 1 (one) minute is allowed for each answer. (3 points, maximum point 1 for a question answered correctly and completely)</li> </ul> <p>The lab evaluation can be undertaken by the student during the last lab meeting of the semester, or in any of the available exam sessions.</p> <p>The students may decide to work for the lab assignment individually, or in teams of two or three. In case the students decide to work in teams, each member of the team will answer at least one of the</p>			

questions during the evaluation of the project. The grade will be awarded to the project, and all team members will receive the same grade.

The final grade is computed as the arithmetic mean of the grades for course and lab, without any rounding of these grades. The final grade is calculated even if either of the two grades is below 5.

The minimum performance standard to pass *Introduction in Programming* is a final grade of at least 5 (five). Not meeting the minimum performance standard will result in failing this class. Failing the class will result in the need to recontract it during the next academic year and redo all activities, in the form defined for that respective year (both content and evaluation may change from one year to the next).

Academic dishonesty of any kind (e.g., plagiarism, cheating, breaking the AI-related rules) will result in the minimum grade, regardless of the criteria listed above.

### 10. Usage of generative AI tools

During this class students will be taught how to use GitHub Copilot, with the purpose of generating code. Students are allowed to use GitHub Copilot and any other generative AI tools while working on their lab assignments. Nevertheless, students are responsible to understand all the code they include in their lab assignments, and they need to be able to explain it line by line. Failing to do so for the final lab evaluation will severely affect the grade, since most of the points are awarded for being able to present, discuss, explain and modify the code.

When turning in the lab assignment, students/teams will also provide a signed declaration of transparency specifying how exactly they used generative AI tools. The form for this declaration will be available in Google Classroom, and the teacher will explain it during the first lab activity.

Failing to provide this form or offering incorrect or incomplete information is considered a breach of academic integrity rules and will be treated according to the university's regulations.

Students are encouraged to use generative AI tools such as ChatGPT for the purpose of getting more details for the topics discussed during the course and lab activities. Students should critically assess the answers they receive from the AI tools and consult with the teacher in all cases when there are apparent discrepancies compared to what is taught in class.

Date of completion  
15.09.2025

Tenure teacher  
Gabriel Fischmann, Ph.D.

Date of approval in the department

Head of Department  
Delia Vîrgă, Ph.D.  
Professor