<table>
<tr><td colspan="2">NAME OF THE COURSE</td><td colspan="5">INTORDUCTION TO PROGRAMMING</td></tr>
<tr><td>Code</td><td>EUBB10</td><td>Year of study</td><td colspan="4">3.</td></tr>
<tr><td>Course teachers</td><td>Full professor Maja Ćukušić;
Assistant professor Tea Mijač,</td><td>Credits (ECTS)</td><td colspan="4">5</td></tr>
<tr><td rowspan="2">Associate teachers</td><td rowspan="2">Assistant professor Tea Mijač,</td><td rowspan="2">Type of instructions (number of hours)</td><td>L</td><td>S</td><td>E</td><td>F</td></tr>
<tr><td>26</td><td></td><td>26</td><td></td></tr>
<tr><td>Status of the course</td><td>Elective</td><td>Percentage of application of e-learning</td><td colspan="4">30%</td></tr>
<tr><td colspan="7">COURSE DESCRIPTION</td></tr>
<tr><td>Course objectives</td><td colspan="6">Acquisition of fundamental knowledge on programming and the importance of making the correct algorithmic solutions to the given problem. Adoption of theoretical knowledge and practical experience from fundamental aspects related to the development and both approaches and methods of solving various problems.</td></tr>
<tr><td>Course enrolment requirements and entry competences required for the course</td><td colspan="6">There are no prerequisites for enrollment.</td></tr>
<tr><td>Learning outcomes expected at the level of the course (4 to 10 learning outcomes)</td><td colspan="6">Learning outcome of the course:
To analyze and apply the basic concepts of software design.

Individual learning outcomes:
Identify, name and explain basic concepts related to historical development, role and principles of software programming.
Identify the different steps / phases of programming and understand the importance of going through each phase without neglecting any phase.
Apply knowledge and skills related to key aspects of programming in order to understand the algorithmic and software solutions.
Link the input process with the appropriate algorithm and evaluate its relevance, accuracy, and speed with regard to specific data input.
Formulate and apply the basic principles of object programming and Web 3 programming on simple business tasks.</td></tr>
<tr><td>Course content broken down in detail by weekly class schedule (syllabus)</td><td colspan="6"><table>
<tr><th colspan="2">Lectures</th><th colspan="2">Exercises</th></tr>
<tr><th>Topic</th><th>Hours</th><th>Topic</th><th>Hours</th></tr>
<tr><td>Introductory lecture. Fundamental concepts related to programming and software support. Algorithm design: flowchart, pseudocode, coding.</td><td>2</td><td>Assignment 1. Getting to know the interface of the software tool.</td><td>2</td></tr>
<tr><td>The concept of a variable. Basic data types (numbers, text, Boolean values). Operators and expressions.</td><td>2</td><td>Assignment 2. Work in software tool. Writing the first program. Declaring variables. Operators.</td><td>2</td></tr>
<tr><td>Branching structures (conditional statements). Logical conditions (if, else, else if structures).</td><td>2</td><td>Assignment 3. Solving business problems using branching (conditional) structures.</td><td>2</td></tr>
</table></td></tr>
</table>

	Loops (iterations). Repeating statements in a program (for and while loops). Processing larger datasets.	2	**Assignment 4.** Solving business problems using loop structures.	2
	Data structures. Lists, arrays, and sets. Working with data collections. Accessing, adding, and deleting elements.	2	**Assignment 5.** Arrays. Sum of an array. Finding minimum and maximum values. Sorting an array.	2
	Built-in (predefined) functions. What functions are and why we use them. The most common built-in functions. More efficient code writing. Field teaching – (virtual) visit to an IT company.	2	**Assignment 6.** Solving business problems using built-in (predefined) functions.	2
	Test			
	User-defined functions. Defining your own functions. Input parameters and return values. Modularity and program readability.	2	**Assignment 7.** Solving business problems by creating user functions.	2
	Working with libraries. What are programming libraries. Importing and using libraries. Examples of useful libraries for data analysis.	4	**Assignment 8.** Using and importing libraries to solve business problems.	2
	Basic concepts of Object-Oriented Programming (OOP). Classes and objects. Attributes and methods.	2	**Assignment 9.** Creating and using classes. Defining classes and methods through own examples.	2
	Object-oriented programming. Inheritance and encapsulation. Code reuse. Advantages of the OOP approach.	2	**Assignment 10.** Solving complex problems using OOP.	2
	Field teaching – (virtual) visit to an IT company on the topic: Introduction to Web3 technologies. Difference between programming for Web1, Web2, and Web3. Blockchain – basic concepts.	4	**Assignment 11.** Solving complex problems using OOP.	2
	Field teaching – (virtual) visit to an IT company on the topic: Introduction to Solidity and smart contracts; basic syntax of a programming language designed for smart contracts.	2	**Assignment 12.** Solving more advanced programming problems.	2
	Development of decentralized applications (dApps). Basics of dApp architecture. Presentation of seminar papers.	2	**Assignment 13.** Seminar paper presentations.	2
	Test			

Format of instruction	☑ lecturers ☑ seminars and workshops ☑ exercises ☐ *on line* entirely ☐ partial e-learning	☑ individual/independent assignments ☐ multimedia ☑ laboratory ☐ work with the mentor

	☑ filed work	☐ ……(other)

Student responsibilities	The student is obliged to attend and to follow the classes regularly, to perform given assignments, and to actively participate in all forms of teaching. Students who successfully preform 70% of assignments from exercises can take the first test, and analogously, the second test. To attain a signature, a regular student must successfully preform 70% of assignments, as well as participating in at least 50% of all class meetings (25% for the part-time students). In addition, students need to present and submit the seminar in the given deadline. The condition for accessing the exam is the signature.

Screening student work *(name the proportion of ECTS credits for each activity so that the total number of ECTS credits is equal to the ECTS value of the course)*						
	Class attendance	1,7 ECTS	Research		Practical training	
	Experimental work		Report		Individual assignments	0,3 ECTS
	Essay	1 ECTS	Seminar essay		(Other)	
	Tests	2 ECTS	Oral exam		(Other)	
	Written exam		Project		(Other)	

Grading and evaluating student work in class and at the final exam	During the semester two tests are conducted (80% of total grade). Students who successfully passed two tests were are exempted from the exam in the regular exam period. Tests are deemed to be passed if the average rating is 60% or more. A weaker percentage in one test can be offset on the other. If a student does not have enough points from tests during the semester, he or she is required to take the final exam. Final exam consists of written exam (60% of total grade) and oral exam (20% of total grade). Students can access the oral exam if the average rating is at least 60% of written exam. Seminar essay takes 20% of total grade. The grade will be determined as follows: 0-59 insufficient (1) 60-69 sufficient (2) 70-79 good (3) 80-89 very good (4) 90-100 excellent (5)

Required literature (available in the library and via other media)	**Title**	**Number of copies in the library**	**Availability via other media**
	Teaching material (2026)		Moodle

Optional literature	• FON (2024) Solidity – Ethereum Blockchain Development, handbook and accompanying video materials, Web3 Innovation (W3IH). • Chemuturi, Murali (2019). Computer Programming for Beginners. Chapman and Hall. • Bell, Aleksander (2019). Computer Programming: Fundamentals for absolute beginers. • Mijač, T., Jadrić, M. & Ćukušić, M. (2019) In Search of a Framework for User-Oriented Data- Driven Development of Information Systems. Economic and business review : for Central and South-Eastern Europe, 21 (3), 439-465 doi:10.15458/ebr.89. • Harvey M. Deitel, Paul J. Deitel, Tem R. Nieto: "Visual Basic.NET How to Program", Fifth Edition, Prentice Hall, 2010.

	• Granić, Andrina; Glavinić, Vlado.: Human Computer Interfaces: Teaching Students to Design for Real-Life Environments // Proc. of the First Edition of Information and Communication Technologies International Symposium ICTIS'05 / Essaidi, Mohamed ; Raissouni, Naoufal (ur.). Tetuan, Maroko: IEEE/Abdelmalek Essaadi University, 2005. 180-183 • Granić, Andrina; Glavinić, Vlado.: Visual Programming Languages as User Interface Performers // Proceedings of 22nd International Convention MIPRO'99, Volume 2 / Biljanović, Petar ; Skala, Karolj ; Ribarić, Slobodan ; Budin, Leo ; (ur.). Rijeka: MIPRO, 1999. 68-71
Quality assurance methods that ensure the acquisition of exit competences	• Monitoring attendance and performance of other student obligations (teacher) • Teaching Supervision (Vice dean for Education and student affairs) • Analysis of the success of studies in all subject studies (Vice dean for Education and student affairs) • Student Survey on the Quality of Teachers and Teaching for Each Subject Study (UNIST, Center for Quality Improvement) • The exam conducted by the subject teacher examines all learning outcomes of the subject. Periodic examination of the content of the exam is conducted on the basis of which the appropriateness of the method of checking the learning outcomes (Vice dean for Education and student affairs)
Other (as the proposer wishes to add)	-