Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Master in Applied Linguistics (LM-39)

Course title:	Computer Programming
Year:	1st year
Semester:	1st semester - 2nd semester
Course code:	54102
Scientific sector:	INF/01
Course teacher:	Gennari Rosella gennari@inf.unibz.it
Modular:	YES
Other teachers	Syed Mehdi Rizvi syedmehdi.rizvi@polimi.it srizvi@unibz.it
Credits:	8
Total lecture/laboratory hours:	**Module 1 (R. Gennari):** 60 hours; **Module 2 (M. Rizvi):** 30 hours.
Total office hours:	18 (Module 1) + 6 (Module 2)
Office hours:	By previously taking appointment **via mail**, with subject "**course in BX**".
Attendance:	according to regulation
Teaching language:	English
Prerequisites:	none
Course description:	The course is for students of the humanities area. It offers an introduction to the basics of computer programming. The course uses the Python programming language and Natural Language Processing (NLP) packages or modules.
Specific educational objectives:	The aim is to provide students with an adequate knowledge of general computer science concepts, and the acquisition of specific knowledge and mastery of the basics of Python programming for their degree course. For specific disciplinary objectives, students are referred to list of topics.
List of covered topics:	This course introduces students to the basics of Python programming, and topics relevant for applied linguistics with Python. Topics of Module 1 are as follows: (1) what computer science is; (2) how a computing device/computer interacts; (3) how to write basic Python programs with atomic statements; (4) how to interpret Python programs; (5) how to test Python programs; (6) how to import relevant Python packages (time, csv, string,...); (7) how to update/install new Python packages;

	(8) how to write Python programs with compound statements: a. branching for decision making (if, else, elif); b. bounded/unbounded iteration for repetitions (for, while); (9) simple and compound data in Python; (10) how to manage Python compound data: a. lists, dictionaries, tuples, sets and further data; b. how to create them; c. how to manipulate them; d. how to traverse them with bounded iteration; e. how to create filter and map with comprehension; (11) how to manage user-defined functions, without/with parameters, in Python; (12) how to manage raw text-files in Python; (13) how to manage csv text-files in Python; (14) how to manage basic exceptions in Python; (15) how to manage regular expressions (re) in Python; (16) how to process text with natural-language rule-based approaches in Python (e.g., via nltk) in order to: a. segment text, b. process text with lexical patterns such as stop-words, c. tokenize text, d. pos-tag text, e. process text with pos-tags (e.g., until Chapter 5 of http://www.nltk.org/book/). Topics of Module 2 are project-based. The following list is only indicative of possible topics, which may change according to the students' projects and skills: (1) basics of html and json for the web; (2) how to process html data in Python; (3) how to process json data in Python; (4) how to plot data in Python, e.g., with pandas (basics, optional) (5) how to process speech in Python, e.g., with Google services; (6) how to play sound files in Python; (7) how to create packages in Python; (8) how to create mash-up projects in Python.
Teaching format:	The course adopts experiential teaching, besides constructionism. It has three main types of classes. **FRONTAL LECTURE CLASSES** Frontal lectures use slides, videos and code snippets as main material. Each frontal lecture is: • c. 50 minute long in in-presence classes, • c. 30 minute long in online classes via the TEAMS app. Frontal lectures mainly take place in Module 1. **CHALLENGE BASED CLASSES** Challenge-based classes span the entire course. The reason is that programming is learnt by "doing", that is, by experiencing it, hands-on, over and over. Such classes challenge students to work on: • brief programming exercises, with program snippets to correct, comment, test or complete; they aim at making students focus on a specific part, explained in the course;

	• longer programming exercises, with programs to complete or write from scratch according to given specifications; they aim at making students connect different parts, explained in the course. The TEAMS app is used to distribute challenges, which are held during class hours. Their resolutions are also discussed during class hours. **WORKSHOP BASED CLASSES** Workshops are held in Module 2. They are similar to challenge-based classes, in that students are asked to program. Whereas in the latter classes the teacher gives specific programming exercises, in workshop-based classes students need to choose a programming project to tackle. The teacher offers students a range of challenges and possible resolutions. Students need to start from these and mash them up in their own programming project, according to the given requirements.
Learning outcomes:	**Knowledge and understanding:** 1. understanding the fundamentals of computer science, 2. understanding a Python program. **Analysis and application of knowledge:** 3. analysing Python programs for resolving computational problems, 4. and applying simple resolution algorithms, by writing short Python programs. **Making judgments;** 5. acquiring critical thinking and making judgments related to the use of Python for tackling computational problems: (a) how to abstract away details, (b) how to model a computational problem (e.g., what data to use), (c) how to resolve it (what algorithm to use), (d) how to resolve it optimally (e.g., when and how to define a function), (e) how to take a critical thinking stance towards one's approach to resolving problems. **Learning and communicating:** 6. ability to learn and work independently, 7. ability to learn and work collaboratively, 8. knowing how to reflect and communicate one's thoughts on a problem and how to solve it computationally.
Assessment:	There are two **alternative** assessments: Intermediate Assessment; Final Assessment. **Intermediate Assessment** Students can take an intermediate exam, split in two parts, one per Module: - programming exercises, based on material of Module 1, at the end of Module 1; - a mashup project, based on material of Module 2, discussed at the end of Module 2. Passing the part related to Module 1 is **necessary** for discussing the mashup project at the end of Module 2. **Final Assessment**

	The final exam is held during the exam day. It consists of two parts, one per Module: - challenges, namely, programming exercises, based on material of Module 1; - a mashup project, based on material of Module 2. Students tackle programming exercises of Module 1 during the exam day. Students work on their mashup project before the exam day and they submit it during the exam day.
Evaluation criteria and criteria for awarding markings:	The evaluation of the course is based on: (1) the evaluation for the part of the assessment related to Module 1; the minimum for passing it is **12** out of **21**; (2) a positive evaluation for the part of the assessment related to Module 2; the minimum for passing it is **6** out of **11**. Marks for Module 1 considers the correctness of resolutions of programming exercises, the quality of resolutions, as well as the displayed analytical and reflective skills. Marks for Module 2 considers whether the mashup project satisfies the requirements given during the course, the overall quality of the project (e.g., its complexity) and its presentation.
Required reading:	Given during classes.
Supplementary reading:	Given during classes.
Software:	**TEAMS** for managing the course material and communication. **Goole Colaboratory** for running Pyton programs, accessed via the student's **unibz** account (not others).