Courses presented by different faculties - BMEEODH001

I. Subject Specification

1. Basic Data

1.1 Title

Programming

1.2 Code

BMEVIHIA061

1.3 Type

Module with associated contact hours

1.4 Contact hours

Type	Hours/week / (days)
Lecture	2
Lab	4

1.5 Evaluation

Exam

1.6 Credits

8

1.7 Coordinator

name	Péter Rucz
academic rank	Assistant professor
email	rucz@hit.bme.hu

1.8 Department

other

1.9 Website

https://epito.bme.hu/BMEEODH001
https://fiek2.mywire.org/course/view.php?id=3561

1.10 Language of instruction

english

1.11 Curriculum requirements

Recommended elective in the Construction Information Technology Engineering (MSc) programme

1.12 Prerequisites

1.13 Effective date

1 September 2022

2. Objectives and learning outcomes

2.1 Objectives

This course aims at teaching the basics of programming and algorithmic problem solving in general to students who are not specialized in informatics. The primary objective of the course is to supply the students with a working knowledge of programming, which they can easily and efficiently apply for solving problems related to the courses in the following semesters as well as in engineering practice. This knowledge base includes the principles of imperative programming, data types and manipulation, input / output operations, object oriented programming, state machines, and event controlled programs. After introducing the basics in the first half of the semester, the second half of the course focuses on practical use cases built around the basics, including graphical programs, networking, and the basics of web programming. Our intention is to use a high-level programming language in this course, which is wide-spread, platform-independent, and has a reasonably large standard library. Therefore, we use the Python language. The course aims to be very practical, thus, each lectures is followed by two laboratories: the first serves for trying out the new elements discussed in the lecture, while the second is for looking into more advanced use-cases.

2.2 Learning outcomes

Upon successful completion of this subject, the student:

A. Knowledge

1. Knows the general principles, rules and methods of mathematics, natural sciences and information technology required to practice engineering tasks related to construction, facility design and implementation.
2. Knows and understands information and communication technologies required for the design and construction of facilities.
3. Has the necessary information technology knowledge to develop technical systems and process automation.

B. Skills

1. Is able to apply the necessary principles of natural sciences and information technology in the design and construction of structures.
2. Applies and develops processes, models and information technologies used by various trades in the design, construction, and operation of facilities.
3. Applies effectively the information and communication technologies required for the design and construction of facilities.
4. Applies integrated knowledge, contributes to solving multidisciplinary problems.

C. Attitudes

1. Is open to solve the tasks individually and cooperate with other participants of the project.

2. Strives to design effective and sustainably operating building information model.
3. Uses the system-based approach for her/his thinking to select an appropriate technical solution which can automatically operate in the long-term.
4. Is willing to acquire the ability of self-learning and self-development.
5. Is open to apply new IT tools, methods and procedures related to a particular field.

D. Autonomy and Responsibility

1. Makes responsible professional decisions concerning the design, construction,maintenance, operation, entrepreneurship and authority tasks of structures.
2. Is willing to initiate solving engineering and IT problems of structures.

2.3 Methods

N/A

2.4 Course outline

Week	**Topics of lectures and/or exercise classes**
1.	Introduction to the Python language. Some basic concepts: algorithms, imperative and declarative programming, compiled and interpreted programming languages. Exercises: Writing simple programs in Python. Using an integrated development environment. Basic input and output interactions with the user.
2.	What is structured code? Control statements, forks, cycles. Storing the state of the program in variables. The string data type, lists, tuples. Exercises: Writing programs using loops and conditions. Debugging programs, running the code step by step.
3.	Functions, parameters, return values. Main and side effects of functions. Passing parameters to functions, variadic parameter list, named parameters. Recursion and recursive traversal, and problem solving strategies. Exercises: Creating simple functions. Passing functions parameters using different settings. Writing functions with multiple outputs as tuples.
4.	Introduction to basic algorithms: searching, selection, counting of elements in a container. Linear and logarithmic search. Some sorting algorithms: bucket sort, bubble sort. Complexity of some algorithms. Exercises: Trying out simple algorithms: selection, extremal elements, sorting.
5.	The principles of object oriented programming. The encapsulation principle. Classes and objects, methods. Constructors. Mutable and immutable objects. Python packages.

	Exercises with creating simple classes
6.	Lists and trees and operations associated with them. The dictionary data type and some typical uses. The binary search tree. Basic tree and graph traversal algorithms. Exercises: Algorithms using lists and dictionaries, list comprehension, traversing the fájl system as a tree.
7.	Input / output operations, file management and exceptions. Processing text and binary files. Handling user input. Creating own exception classes. Exercises: Working with text and binary files, examples of exception handling.
8.	Programming using the state machine approach. The general theory of state machines. Regular expressions. Event controlled programming. Exercises: Simple examples of state machines: e.g. smart traffic light, comment remover. Using regular expressions.
9.	Creating graphical programs. Windows, display, user interfaces, creating different types of widgets. Handling user events. Exercise: Creating a simple game with graphics.
10.	The numpy package. Using arrays. Matrix operations. Combining Python and Matlab. The scipy package. Creating plots using the matplotlib package. Exercise: Using numpy for basic matrix manipulation. Solving linear systems of equations, finding eigenvalues.
11.	The basics of network programming. Sockets. Server-client architectures. Creating a simple network service. Exercise: Programming a simple network service: time server. More advanced example: implementing a network chat program.
12.	Wep programming: Http requests and responses. Creating a web server application. Exercise: Assembling a web server application.
13.	Python and database programming. Handling databases, storing data in SQLite. Managing database operations. Exercise: Using a database for data persistence in our application. Implementing queries for communicating with the database engine.
14.	**Case study:** Building a complex application.

The above programme is tentative and subject to changes due to calendar variations and other reasons specific to the actual semester. Consult the effective detailed course schedule of the course on the subject website.

2.5 Study materials

1. Mark Summerfield - Programming in Python 3: A Complete Introduction to the Python Language, Second Edition. Addison-Wesley (2009)
2. The Python standard library documentation https://docs.python.org/3/library/

2.6 Other information

2.7 Consultation

Consultations will be provided as needed, based on prior agreement.

This Subject Datasheet is valid for:

Inactive courses

II. Subject requirements

Assessment and evaluation of the learning outcomes

3.1 General rules

The final grade results from

- two mid-term tests (both can be substituted)
- preparation of the homework (specification, coding, documentation, acceptance by lecturers)
- presenting the homework in the form of an oral exam

3.2 Assessment methods

Evaluation form	Abbreviation	Assessed learning outcomes
Mid-term test 1	MT1	A.1-A.2; C.1
Mid-term test 2	MT2	A.1-A.2; C.1
Homework project	HW	A.3; B.1-B.4; C.2-C.3, C.5; D.1-D.2
Oral exam	EX	A.1-A.3; C.1

The dates of deadlines of assignments/homework can be found in the detailed course schedule on the subject's website.

3.3 Evaluation system

Abbreviation	Score
MT1	20%
MT2	20%
HW	30%
OE	30%
Sum	**100%**

3.4 Requirements and validity of signature

The requirement of the signature is successfully passing both mid-term tests (MT1 and MT2).

3.5 Grading system

Grade	Points (P)
Excellent	P > 85
Good	70 < P <= 85
Mediocre	55 < P <= 70
Passed	40 < P <= 55
Fail	P < 41

The two mid-term tests give 2x20 pts.
The homework and the oral exam gives 30-30 pts.
Extra points may be gained during the semester by submitting solutions to extra tasks.

3.6 Retake and repeat

The midterm tests MT1 and MT2 can be retaken / repeated during the repetition week.

3.7 Estimated workload

Activity	**Hours/semester**
Contact classes	84
Midterm preparation	28
Preparation for midterm tests	28
Preparation of the homework	50
Learning from the assigned supplementary material	40
Preparation for the exam	10
Sum	**240**

3.8 Effective date

1 September 2022

This Subject Datasheet is valid for:

Inactive courses