Introduction to Course

Lect. PhD. Arthur Molnar

Babes-Bolyai University

Overview

1 Introduction to course

- Schedule
- Objectives
- Course content
- Bibliography
- Activity and grading

Guiding professors

- Lect. PhD. Arthur Molnar
- Lect. PhD. Andrei Mihai
- Lect. PhD. Mircea Ioan-Gabriel
- Assist. Briciu Anamaria
- Assist. Imre Zsigmond
- Vasilica Moldovan, PhD. student

Schedule

- **Lecture**: 2 hours/week
- **Seminar**: 2 hours/week
- **Laboratory**: 2 hours/week
- **Consultation**: optional, each teacher has a weekly time slot (will be announced on Teams)

Course materials

- **Teams**, **General** channel, **Files** section
- Public **FP** repository `https://github.com/cs-ubbcluj-ro/FP`

Contact us

Best way is using **Teams** chat

Objectives

What should you learn during this course?

- Key programming concepts
- A few introductory problem solving methods
- Basic concepts of software engineering (design, implementation and maintenance of software systems)
- Use basic software tools such as IDE's, source version control, documentation generators, testing tools
- Acquire and improve your programming style.
- The basics of programming using the Python language

Course content

How is this course organized?

- Programming in the small
- Programming in the large

Programming in the small

1. Recursion
2. Computational complexity
3. Searching. Sorting
4. Problem solving methods

Programming in the large

5. Procedural programming
6. Modular Programming
7. Test Driven Development
8. Design Principles for Modular Programs
9. User Defined Types and Exceptions
10. Introduction to UML
11. Design Principles for Object Oriented Programs
12. Program Testing. Refactoring.
13. Layered architecture. Inheritance.
14. Intro to building GUIs

Bibliography

1. Kent Beck - **Test Driven Development: By Example**; Addison-Wesley Longman, 2002.
2. Kleinberg and Tardos **Algorithm Design**; Pearson Educational; 2014 (http://www.cs.princeton.edu/ wayne/kleinberg-tardos/)
3. Martin Fowler - **Refactoring. Improving the Design of Existing Code**; Addison-Wesley, 1999. (http://refactoring.com/catalog/index.html)
4. Frentiu, M., H.F. Pop, Serban G. - **Programming Fundamentals**; Cluj University Press, 2006
5. Online Python resources - `https://docs.python.org/3/reference/index.html`, `https://docs.python.org/3/library/index.html`, `https://docs.python.org/3/tutorial/index.html`, `https://realpython.com/`

Activity and grading

- **40%** - Laboratory work (assignments and tests **(L)**
- **30%** - Written exam (during exam session) **(W)**
- **30%** - Practical test (during exam session) **(T)**
- **0 - 0.5p** Seminar activity (bonus to laboratory grade)
- **0 - 1p** Additional laboratory activity (bonus to laboratory grade)

Passing the course

- Mandatory attendance to enter examination during 2024
- **L** grade ≥ 5 to enter examination during regular session
- **L**, **T** and **W** grades all ≥ 5 to pass the course

Activity and grading

Grading example

Suppose your grades are:

- Laboratory - 7
- Written - 7.50
- Practical - 6.80
- Seminar bonus - 0.30
- Laboratory bonus - 1

Your grade is calculated as: $0.4 * (7 + 0.3 + 1) + 0.3 * 7.5 + 0.3 * 6.8 = 7.61$, final grade is 8

About the Practical Exam

About the Practical Exam

- Only **working** functionalities are graded
- Everything required for implementation will be studied
- Each problem will be interesting, in its own way
- Getting the extra points during the semester will help improve your grade

Course Rules

- Seminar attendance mandatory **(10/14)**
- Laboratory attendance mandatory **(12/14)**
- Without making attendance you can't enter the exam this year!
- Detailed rules for laboratory activities are on the **General** channel, **Files** section
- **Be honest, solve the graded assignments by yourself, do not plagiarize!**