Introduction to Course

Lect. PhD. Arthur Molnar

Babes-Bolyai University

arthur@cs.ubbcluj.ro

Overview

Guiding professors

- Lect. PhD. Arthur Molnar
- Lect. PhD. Radu Gaceanu
- Lect. PhD. Mircea Ioan-Gabriel
- Assist. Imre Zsigmond
- Assist. Sergiu Nistor
- Assist. Briciu Anamaria

Schedule

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

Course materials

- **Teams**, **General** channel, **Files** section
- **FP** repository on GitHub Classroom

Contact us

Best way is using **Teams** chat

Objectives

What should you gain from this course?

- Learn key programming concepts
- Learn the basic concepts of software engineering (design, implementation and maintenance of software systems)
- Learn to use basic software tools such as IDE's, documentation generators, testing tools
- Acquire and improve your programming style.
- Learn the basics of programming using the Python language

Course content

How is this course organized?

- Programming in the large
- Programming in the small

Programming in the large

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

Programming in the small

11 Recursion
12 Computational complexity
13 Searching. Sorting
14 Problem solving methods

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

Activity and grading

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

Passing the course

- Mandatory attendance to enter examination during 2021
- **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 - 6.55
- Written - 7.50
- Practical - 6.80
- Seminar bonus - 0.40
- Laboratory bonus - 0.20

Your grade is calculated as: $0.3 * 6.55 + 0.4 * 7.5 + 0.3 * 6.8 + 0.4 + 0.2 = 7.00 + 0.4 + 0.2 = 7.60$, final grade is 8

Course Rules

- Seminar attendance mandatory **(10/14)**, measured according to your commits for the **Seminar** repository
- Laboratory attendance mandatory **(12/14)**, measured according to handing in assignments or interacting with teachers
- Without making attendance you can't enter the exam this year!
- Do not plagiarize
- Detailed rules for laboratory activities are on the **General** channel, **Files** section

About the Practical Exam

About the Practical Exam

- You are graded only for **working** functionalities
- 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