STUDY MODULE DESCRIPTION FORM

Name of the module/subject **Technologies of Software Development**			Code **1010512311010517176**
Field of study **Computing**	Profile of study (general academic, practical) **general academic**		Year /Semester **1 / 1**
Elective path/specialty **Software Engineering**	Subject offered in: **English**		Course (compulsory, elective) **obligatory**
Cycle of study: **Second-cycle studies**	Form of study (full-time,part-time) **full-time**		
No. of hours Lecture: **30** Classes: **-** Laboratory: **30** Project/seminars: **-**			No. of credits **5**
Status of the course in the study program (Basic, major, other) **major**	(university-wide, from another field) **from field**		
Education areas and fields of science and art **technical sciences**			ECTS distribution (number and %**)** **5 100%**

Responsible for subject / lecturer:

dr inż. Adam Wojciechowski
email: Adam.Wojciechowski@cs.put.poznan.pl
tel. 61 6652983
Institute of Computing Science
Piotrowo 2 Str., 60-965 Poznan

Prerequisites in terms of knowledge, skills and social competencies:

1	**Knowledge**	Learning objectives of the first cycle studies defined in the resolution of the PUT Academic Senate, especially K_W1-2, K_W4, K_W6-15, K_U1-2, K_U4, K_U7-8, K_U14-20, K_U22-23, K_U26, K_K1-9 that are verified in the admission process to the second cycle studies ? the learning objectives are available at the website of the faculty www.fc.put.poznan.pl Student starting this module should have a basic knowledge regarding basic algorithms and computational complexity, object-oriented programming, design patterns, human-computer interaction, databases, software testing and web applications.
2	**Skills**	Should have skills allowing solving basic problems related to data safety, requirements analysis, creating software specification, data modeling, designing systems and skills that are necessary to acquire information from given sources of information.
3	**Social competencies**	Student should understand the need to extend his/her competences / has the willingness to work in a team. In addition, in respect to the social skills the student should show attitudes as honesty, responsibility, perseverance, curiosity, creativity, manners, and respect for other people.

Assumptions and objectives of the course:

1. Provide students knowledge regarding Python programming language, creating websites using Django framework, natural language processing
2. Develop students? skills in solving problems related to creating application using Python
3. Present students a set of development technologies for modeling data layer, designing interace layer, defining communication layer between several applications
4. Develop students? teamwork skills in the context of developing software systems

Study outcomes and reference to the educational results for a field of study

Knowledge:

1. has well-established theoretical knowledge of implementation of algorithms, evaluation of their complexity, architecture of software systems, role of script languages in operating systems, network technologies in web applications, databases - [K_W4]

2. has detailed theoretical knowledge related to selected areas of computer science, Python programming language, web applications, script testing, testing code which isn?t compiled - [K_W5]

3. has knowledge regarding trends and the most important new developments in computer science and related disciplines - [K_W6]

4. has basic knowledge regarding life-cycle of software or hardware systems - [K_W7]

5. knows the fundamental methods, techniques and tools employed to solve complex engineering tasks in creating scripts - [K_W8]

Skills:

1. is able to acquire, combine, interpret and evaluate information from literature, databases and other information sources (in mother tongue and English); draw conclusions, and formulate opinions based on it. - [K_U1]

2. is able to plan and arrange self-education process - [K_U5]

3. has language skills at B2+ level in accordance with the requirements set out for level B2+ Common European Framework of Reference for Languages - [K_U6]

4. is able to combine knowledge from different areas of computer science (and if necessary from other scientific disciplines) to formulate and solve engineering tasks; and use system approach that also incorporates nontechnical aspects - [K_U10]

5. is able to formulate and test hypotheses regarding engineering problems and basic research problems - [K_U12]

6. is able to assess usefulness and possibility of employing new developments (methods and tools) and new IT products - [K_U13]

7. is able to choose appropriate programming language and use it to solve a particular task - [K_U26]

8. is able to design (according to a provided specification which includes also non-technical aspects) a complex device, an IT system, or a process; and is able implement it (at least partially) using appropriate methods, techniques, and tools (including adjustment of available tools or developing new ones) - [K_U27]

Social competencies:

1. understands that knowledge and skills related to computer science quickly become obsolete - [K_K1]

2. is able to correctly assign priorities to own tasks and tasks performed by others - [K_K6]

Assessment methods of study outcomes

Formative assessment:

a) lectures:

- based on the answers to the questions which test understanding of material presented on the lectures

b) laboratory classes / tutorials / projects / seminars:

- based on the assessment of the tasks done during classes and as a homework

Summative assessment:

a) verification of assumed learning objectives related to lectures:

- assessment of knowledge and skills, examined by a written test with multiple choices and problem questions. Student can gain 100 points, to pass minimum 50 points are needed
- discussing the results of the examination

b) verification of assumed learning objectives related to laboratory classes / tutorials / projects / seminars:

- assessment of student?s preparation to particular laboratory classes and assessment of student?s skills needed to realize tasks on these classes
- continuous assessment of student?s work during classes ? rewarding ability to use learned principles and methods
- assessment of projects realization, including ability to work in team

Possibility to gain additional points by activity on classes:

- elaboration of additional aspects regarding the subject
- effectiveness of applying acquired knowledge to solve problems
- ability to cooperate with the team during solving problems
- providing additional remarks for the lecturer how to improve teaching materials highlighting the problems with students? perception to improve the teaching process

Course description

The program of the lecture:

Introduction to Python programming language with demonstration of its application areas. Presentation of the language syntax with highlighted differences between other languages as Java and C. Demonstration of example programs and common errors made by developers. Code documenting. Regular expressions in Python. Description of Python libraries? compatibility with Posix standard. Applying Python to creating web pages using Django framework. Description of application lifecycle, data model and rules of generating views. Creating generic webpage templates using Django and its influence on server performance. Description of optimization techniques for static data. Advanced programming techniques in Python, such as the lambda keyword, enrichment object with new functionality and operations on sets of data. Introduction to the analysis of natural language using the Natural Language Toolkit library (NLTK). Presenting ideas and basic operations on the text. The introduction of the tool and its capabilities. Creating graphical applications using PyGTK library. Discussion of the differences between the use of RAD tool for creating applications and the creation of the GUI from the code. Overview of technologies for creating data model layer. Overview of technologies for creating communication between applications. Overview of the visual technologies for creating user interfaces.

The course consists of fifteen 2-hour laboratory classes and it starts with an instructional session at the beginning of a semester. Students work individually or in teams of 2-4. The program of laboratory classes is following:

Presentation of sample programs in Python. Creating different types of programs in order to practice the syntax and commands. Presentation of projects to be implemented. Exercise of documenting code. Creating regular expressions. Using expressions for common tasks (text search, validate data entered by the user). Techniques for improving application performance using regular expressions. Creating projects and applications in Django framework. Using regular expressions to control the flow of HTTP requests. Generating simple web sites. Presentation of projects to be implemented . Using generic templates to increase code reuse and improve performance of web applications. Presentation of code allowing to increase the efficiency of programs written in Python. Analysis of the text using the NLTK library. Exercises with sentence dissection, analysis of sentence parts, providing modifications to sample sentences. Using Wordnet and PlWordnet databases. Presentation of projects to be implemented. Creating applications using the GTK+ library. Creating a data model layer. Developing applications that communicate with other using learned technologies. Creating a presentation layer.

Learning methods:

1. Lectures: multimedia presentation, presentation illustrated with examples presented on black board, solving tasks, multimedia showcase
2. Tutorials: solving tasks, practical exercises, performing experiments, discussion, teamwork, multimedia showcase, workshops, team games, case studies, tutorial

Basic bibliography:

1. Programming Python, Mark Lutz, O'Reilly Media 2011
2. Python Web Development with Django, Jeff Forcier, Paul Bissex, Wesley J Chun, New York, Addison-Wesley Professional 2008
3. Python Text Processing with NLTK 2.0 Cookbook, Jacob Perkins, Packt Publishing 2010
4. Architektura oprogramowania w praktyce, L. Bass, P. Clements, R. Kazman, WNT
5. SOA Design Pattern, Thomas Erl

Additional bibliography:

1. Foundations of GTK+ Development, Andrew Krause, Apress 2007
2. Introduction to Python Programming and Developing GUI Applications with PyQT, B. M. Harwani, Course Technology PTR 2011

Result of average student's workload

Activity	Time (working hours)
1. participating in laboratory classes / tutorials: 15 x 2 hours,	30
2. preparing to laboratory classes: 15 x 1 hours,	15
3. consulting issues related to the subject of the course; especially related to t laboratory classes and projects,	10
4. implementing, running and verifying software application(s) (in addition to laboratory classes)	22
5. participating in lectures	30
6. studying literature / learning aids (10 pages = 1 hour), 50 pages	5
7. discussing the results of the examination	1
8. preparing to and participating in final assessment tests related to lectures (10h + 2h)	12

Student’s workload

Source of workload	hours	ECTS
Total workload	125	5
Contact hours	73	3
Practical activities	77	3