Approved by the Academic council
of the Education programme
«Computational linguistics»
Protocol No. 13 from 23.08. 2019

Syllabus

Introduction into scientific programming
(3 ECTS)

Nick Howell
nhowell@hse.ru,
https://www.hse.ru/org/persons/209638461?fbclid=IwAR0lUTrqbPmvXXZLgZlJSyUbH0jU8sQdUZ2FMhE6G3jPW4hPeTBu2F5XJ78

1. Course Description

a) Pre-requisites

This course does not require any specific backgroung. There are no pre-requisites for it.

b) Abstract

The course is designed to further the students' knowledge of natural language processing and to polish their programming skills. The course aims to provide the students with the programming and natural language processing knowledge and competencies necessary to plan and conduct research projects of their own leading to the M.Sc. dissertation and scientific publications.

2. Learning Objectives

The course aims:

- to further the students' programming skills;
- to provide them with the necessary skills to write programs for experiments and corpus studies;
- to teach them how to re-format data;
- to teach them how to retrieve data from the Internet;
- to teach the students how to write their code so that it is readable by other linguists;
- to teach them how to present their research that involves coding in the written and in the oral form;
- to provide an overview of some of the most exciting current computational projects;
- to teach the students how to read and to assess critically linguistic research that uses computational methods;
- to teach them how to formulate linguistic questions in a way that can be addressed computationally;
- to teach them to conduct independent computational studies.

3. Learning Outcomes

On completion of the course, the students will be able:

- to write programs (code) for experiments and corpus studies;
- to re-format data;
- to retrieve data from the Internet;
- to write their code so that it is readable by other linguists and programmers;
- to present their research that involves coding in the written and in the oral form;
- to read and to assess critically linguistic research that uses computational methods;
- to formulate linguistic questions in a way that can be addressed computationally;
- to conduct independent natural language processing studies.

4. Course Plan

1 Computer architecture I. Ideas of electronics, physical components of the computer. Software, and software layering in modern multi-user computers (kernel, system, user software). Contrast against unikernel (less layering) and against highly virtualised (more layering) systems. Networking of various reliability/bandwidth/latency levels.

2 Data representation I. Text encodings, various unicodes. Numerical representations. Simple data structures: lists, trees, graphs. Elements of graph theory. Cryptographic ideas: hashing, symmetric and asymmetric ciphers.

3 Interfaces I. Command-line and text interfaces. Historical perspective. Teletypes, shells, Read-Evaluate-Print Loops. Basic *nix-style conventions for interaction. Basics of shell scripting. Processes and job control. Quick intro to SSH, text editors, GnuPG, Git.

4 Collaboration I. Communications channels from latency, flexibility, control, trust perspectives. Version control. Quality control: bug reporting, continuous integration, documentation.

5 Licensing. Clarity, conformity. Interactions between business, academia, government, and community. Copy-left, open, proprietary.

6 Software design patterns. Monolithic vs modular vs microservice design. Examples in kernel, system, and user.

7 exams

8 Software creation I. Compilers and assembly. Libraries and portability, linking and build. Interpreters.

9 Software packaging I. User-local / system-global installation. Managed vs unmanaged installs. Package management: distributions, languages, user. Image-based: VMs, containers.

10 Collaboration II. Patch review. Rebasing, merging, reverting. Filters and hooks.

11 Graphical and web interfaces. Accessibility, scriptability, portability.

12 Computer architecture II. Heterogeneous computing, cluster computing, distributed computing. Hard vs easy parallelism.

13 Software creation II. JIT compilation. Cross-compile and heterogeneous computing. Optimisation.

15 Data representation/Interfaces II. Distributed data structures. Databases. Schedulers and cluster management.

16 exams

5. Reading List

Required

No required reading is assigned in this course.

Optional

i. Course handouts.

ii. Python official documentation (Open access: https://www.python.org/doc/).

iii. Mystem official documentation (Open access: https://tech.yandex.ru/mystem/doc/index-docpage/).

iv. Hammond, Michael. *Python for Linguists*. Unpublished draft. University of Arizona. August 22, 2017.

(Available on the author's official webpage: https://faculty.sbs.arizona.edu/hammond/ling508-f17/pyling.pdf).

6. Grading System

The course spans over two modules. Throughout both modules students will be given homework as a type of intermediate assessment. Students will be asked once to deliver a 15 min. oral presentation on an NLP topic of their choosing. Students will be asked to submit a final project by the end of the second module.

Homework assignments: students are required to submit two homework assignments; they are given a week to complete each assignment, e.g., the students will be asked to create a corpus of on-line reviews with a web interface; to build a tokenizer for language X etc.

In-class presentation: students will be asked to deliver a 15 min. oral presentation on an NLP topic of their choosing.

Final project (exam): students will be asked to submit a piece of code that addresses some linguistic problem relevant for their research. At least three topics discussed in the class should be used in the code.

Final grade calculator:

30% -- final project (exam)

20% -- homework assignment 1

25% -- homework assignment 2

25% -- in-class presentation

Final grade is rounded in favor of the student. There are no blocking elements of the grade. The exam can be repassed according to the rules approved by the Academic council of the Education programme. The other assignments are not to be repassed. Students are assessed based on a 10-point grading scale: https://www.hse.ru/studyspravka/Scale/.

Table of Grade Accordance

10-point grading scale	**5-point grading scale**	
1 - very bad 2 – bad 3 – no pass	no pass – 2	**FAIL**
4 – pass 5 – highly pass	pass – 3	**PASS**
6 – good 7 – very good	good – 4	
8 – almost excellent 9 – excellent 10 – perfect	excellent – 5	

7. Examination type

Examination takes place in a form of a written final project described in the previous section. Examination grade is not a blocking grade.

8. Methods of Instruction

The class is a mixture of:

- lectures on the natural language processing;
- seminars on programming with training problem solving and the discussion of the solutions;
- oral presentations;
- individual instruction and feedback on homework assignments and the final project.

9. Special Equipment and Software Support (if required)

The course is delivered in lectures and classes where students are required to write programming code in order to solve training problems. The course requires computer rooms with an access to the Internet and a projector.