Syllabus for Programming Basics

1. Course Description

a. Title of the Course

Programming Basics is an adaptation course for the first year MA students of the program Linguistic theory and language description.

b. Pre-requisites

There are no prerequisites for this course; the course is aimed at those who have no or almost no prior experience with computer programming. A level of English proficiency sufficient for the participation in an MA program is assumed.

c. Course Type (compulsory, elective, optional)

This is an optional course.

d. Abstract

The course is an introduction to data processing and programming for theoretical linguists. It assumes little to no background in programming. The course aims to make the students comfortable with programming and with using programming in their own linguistic research. The first goal of the course is to introduce the basics of programming; the second goal of the course is to show the students that they can solve linguistic problems that they are interested in and working on using the tools they were given. The course provides an opportunity for the students to work on their own research problem as their final project.

2. Learning Objectives

The course aims:

- to teach the students the basics of Python;
- to make them comfortable using programming in their linguistic research;
- to familiarize them with some of the advances in data processing and natural language processing;
- to teach them how to create a linguistic corpus, how to retrieve data from the Internet, how to parse it, how to analyze it computationally;
- to teach the students how to present their computational work;
- to teach the students how to read each others' code and how to ethically use open access code written by somebody else.

3. Learning Outcomes

The intended outcomes are:

- students will be comfortable with the Unix shell and simple commands related to processing text;
- students will know basic constructions and functions of Python, as well as the most frequently used functions and modules used for text processing;

- students will be able to use regular expressions for information extraction;
- students will be able to retrieve data from the Internet, parse it and use it for linguistic processing;
- students will be able to write simple scripts in Python in order to use them in their own linguistic research;
- students will understand how common file forms for linguistic data work;
- students will learn how to structure their code, how to make it readable and reusable, and how to present it to other linguists.

4. Course Plan

Week 1: Interacting with Python. Data types and variables.

Installing and using Python. The interactive environment. Edit and run. Doing basic math in Python. Assignment of variables. Basic data types: numbers, Booleans, strings, lists, tuples, dictionaries. Mutability. Homework 1 is distributed.

Week 2: Control structures.

Grouping and indentation. *If*, *for*, *while*, *break*, *continue* operators. Homework 2 is distributed.

Week 3: Input and output. Modules.

Command-line input. Keyboard input. File input. File output. Simple functions. Functions that return values. Functions that take arguments. Modules. Writing your own modules. Homework 3 is distributed.

Week 4: Regular expressions.

Definition. Matching. Patterns. Homework 4 is distributed.

Week 5: Text manipulation. Choosing a final project.

Manipulating text. Morphological parsing. An example of a morphological parser: Mystem.

Week 6: Internet data.

Retrieving data. HTML. HTML parsing.

Week 7: Visualization. Code presentation. Final project consultation.

Drawing graphs in Python. Markdown. Jupiter notebook.

Week 8: Context-free grammars in Python. Final project presentations.

5. Reading List

a. Required

No required reading is assigned in this course.

b. 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

Active participation: students are required to attend the class, to ask questions, to participate in the discussions, to solve the training problems provided in class.

Homework assignments: students are required to submit four homework assignments; they are given a week to complete each assignment.

Final project: students are asked to submit a piece of code that addresses some linguistic problem. The code should be roughly equivalent to one-two homework assignments. It should be working and it should contain at least three independently defined functions. Along with the code, the students are asked to submit a write-up. The write-up could be done as a text part of the .ipynb file or as a text document separate from the file with the code. It should not be very long, but it should: a) state the goal of the project; b) state why the project is interesting; c) reformulate the problem in terms of functions that the student plans to code; d) describe what the program does and how it does it; e) state what the code accomplishes; f) state the program limitations and possible future improvements. Students are also asked to deliver a 7 min. oral presentation of their work and answer their peers' questions afterwards.

7. Guidelines for Knowledge Assessment

Final grade calculator:

40% -- final project

50% -- homework assignments

10% -- active participation

Final grade is rounded in favor of the student. Students are assessed based on the 10-point grading scale.

Table of Grade Accordance

<table>
<tr><th>10-point grading scale</th><th>5-point grading scale</th><th></th></tr>
<tr><td>1 - very bad
2 – bad
3 – no pass</td><td>no pass – 2</td><td>FAIL</td></tr>
<tr><td>4 – pass
5 – highly pass</td><td>pass – 3</td><td rowspan="3">PASS</td></tr>
<tr><td>6 – good
7 – very good</td><td>good – 4</td></tr>
<tr><td>8 – almost excellent
9 – excellent</td><td>excellent – 5</td></tr>
</table>

10 – perfect		

8. Methods of Instruction

Each class the instructor introduces new material and presents students with training problems that target the introduced material. Students are then given some time, five to twenty minutes, depending on the problem, to write code that solves the problem. Throughout this time, the instructor is available for individual instruction. At the end of the allotted time, the instructor provides their solution to the problem and discusses it with the students.

Homework assignments consist of one to five problems. Students are either required to supplement the missing parts of the code or to write the whole program that solves the problem. All the assignments target some linguistic problem. An example problem: write a piece of code that extracts all the word-initial consonant clusters from a given text file and sorts the clusters from the most frequent one to the least frequent one. Students receive individual feedback on each assignment via e-mail and are provided with the instructor's solution code for the assignment for reference.

Final projects challenge the students to write an independent piece of code that is related to their linguistic interests. Along with the piece of code, students are asked to submit a write-up of the project and to deliver a short oral presentation. Students receive individual feedback on their final project from the instructor and discuss their projects with each other after the oral presentations.

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.