GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Lecture 1: Introduction

Introduction to programming (LT2111)

Markus Forsberg
Språkbanken
University of Gothenburg

2011-09-06

GÖTEBORGS UNIVERSITET

Språkbanken

CLT

Introduction & Administration

- The main goal of the course is that you will learn how to program using the programming language *Python*.
- Teachers:

Markus Forsberg
Course coordinator
markus.forsberg@gu.se

Johan Roxendal
Course assistant
johan.roxendal@gu.se

Schedule

- Course homepage:
  spraakbanken.gu.se/personal/markus/introduction_to_
  programming
- We will meet on Tuesdays and Fridays:
  - lecture
  - assignment supervision
  - exercise session
- 45 min + 15 min break + 45 min

Course literature

- Main course book: *Python for software design, how to think like a computer scientist, Allen B. Downey*
- *Natural Language Processing with Python, Steven Bird et al.* (we will only use the first chapters, but it is the main book in the ‘Programming in NLP’ course)
- The books are available online for free (linked from the homepage).
- Paperbacks cost around 25 euro each.
- Python documentation at the Python website: http://docs.python.org/.

Lectures

- Tuesdays, 10.15-12.00
- The main goal of the lectures is to help you grasp the theoretical content of the course.
- Please mail me about parts of the course that you find especially difficult, and I will try to include more material about it in the coming lectures.
- The slides are put on the course homepage after the lecture (as quickly as I can manage).

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Assignments

- Assignment supervision:
  Tuesdays, 13.15-15.00
  Fridays, 13.15-15.00
- 3 obligatory practical assignments, 1 optional, but recommended (this week).
- The assignments are done in groups of two.
- Do not make the mistake of being a passive member of a group! Switch control of the keyboard frequently!

Exercises

- Exercise sessions: Fridays, 10.15-12.00
- Paper-and-pen programming (except for the first week, which is practical)
- In the exercise session we solve the week's problems (that you should try to solve beforehand) and discuss alternative solutions.
- Why no computers?

Exam

- Date: week 43 (exact day yet to be decided)
- Grade: Pass with distinction, Pass, or Fail

Computer science crash course

- Computer science is the study of *computation*.
- computation = problem solving
- *Algorithm*: a detailed account of how to solve a problem.
- *Programming language*: a formal language to express computations.

Formal vs. natural languages

- *Natural languages*: what we normally mean by languages, i.e., what people speak.
- *Formal languages*: man-made languages designed for a specific purpose, such as programming languages.
- Differences:
  - *ambiguity*
  - *redundancy*
  - *literalness*

GÖTEBORGS UNIVERSITET

Språkbanken

CLT

Programming in a nutshell

- Input (keyboard, file, other devices)
- Output (screen, file, other device)
- Math (addition, multiplication)
- Conditional execution (select what to execute based on a condition)
- Repetition (usually combined with conditionals)
- That's it!
- Or is it?

GÖTEBORGS UNIVERSITET

Språkbanken

CLT

Low-level language: Assembler

```
section .data
        str:    db 'Hello world!', 0Ah
        strLen: equ $-str
   section .text
        global _start
   _start:
        mov eax,4
        mov ebx,1
        mov ecx,str
        mov edx,strLen
        int 80h
        mov eax,1
        mov ebx,0
        int 80h
```

High-level language: Python

```
print "Hello world!"
```

High-level language

- The difference is in the level of *abstraction* — details are hidden in a high-level language.
- A high-level language allows us to be much more productive.
- It also separates us from the machine, which makes our programs *portable*.
- However, for every new programming language you need to learn the abstraction of that language.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Standing on the shoulders of giants

- Programming is all about building on what others have done.
- Using a high-level programming language is exactly that.
- Instead of trying to reinvent the wheel, we often use code defined by others that helps us solve a particular problem.
- Python terminology: a *library* consists of *packages* that consist of *modules*. A module is a file containing code. (More about this later)
- *Python standard library* is always available with the Python program.
- However, we will actually many times reinvent the wheel just for the practice.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Some Python terminology

- *values*: basic things a program works with, like letters and numbers.
- *expression*: denotes a value, possibly after some computation (5+5 denotes 10).
- *types*: every value has a type, e.g., **2** is an *integer*, "**Hello world!**" is a *string*.
- *variables*: gives a name to a value. A variable has the same type as the value.
- *Statement*: performs an action, such as printing a string or assigning a name to a value.

Example

```
$ python
...
>>> type(12)
<type 'int'>

>>> type(12+12)
<type 'int'>

>>> name = 12+12

>>> name
24

>> type(name)
<type 'int'>
```

When things go wrong

Syntax errors
: are formal errors that may be lexical or syntactical.

Runtime errors
: are errors, also referred to as *exceptions*, occurring while running a program.

Semantic errors
: are errors where the program actually runs, but fails to do what we want.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Last year's FAQ: Typing some symbols on a Mac

```
shift     +8 = (
shift     +9 = )
      alt+8 = [
      alt+9 = ]
shift+alt+8 = {
shift+alt+9 = }
```

The logic: parentheses-like symbols on the same keys.

Last year's FAQ: What is NLTK?

- NLTK (Natural Language ToolKit) is not a part of standard Python, it is a Python package that requires separate installation.
- NLTK covers a wide range of Language Technology subjects and methods.
- NLTK also provides many Language Technology resources, e.g., WordNet that we will work with in assignment 1.

Last year's FAQ: How do I install NLTK on my own computer?

- Instructions are found here: http://www.nltk.org/download

Last year's FAQ: floating point division

```
>>> 4/10
0
>>> from __future__ import division
>>> 4/10
0.4
```

- What is `__future__`?
- First: changing how something such as 'division' works, even if it is a good idea, must be made conservatively, to avoid breaking existing code.
- But programmers are allowed to use the new division, if they explicitly declare that, hence: `from __future__ import division`
- Why the strange name `__future__`? Python's built-in things have names with surrounding double underscore to avoid that you be accident would use that name.

Last year's FAQ: floating point division (cont.)

```
>>> 1.0/6
0.16666666666666666

>>> x = 1

>>> y = 6

>>> float(x)/y
0.16666666666666666
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Assignment 0: description

- Not obligatory, but highly recommended.
- A hands-on assignment, where you will be familiarized with both programming in Python and Language Technology.
- Do not except to understand everything! Just work your way through the examples.
- Chapter 1 of the NLTK book.
- We will now spend the rest of the lecture on a live demo to get you started.