Introduction to programming
Lecture 1: Introduction

UNIVERSITY OF GOTHENBURG

Richard Johansson

September 1, 2015

Introduction & Administration

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

Richard Johansson
Course coordinator
richard.johansson@gu.se

Luis Nieto Piña
Course assistant
luis.nieto.pina@gu.se

Ildikó Pilán
Course assistant
ildiko.pilan@gu.se

Schedule

- Course homepage:

  `http://spraakbanken.gu.se/personal/richard/itp2015`
- We will meet on Tuesdays and Fridays:
  - lecture
  - assignment supervision

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 – make sure you get the latest version.
- 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).

Assignments

- Assignment supervision:
  Tuesdays, 13.15-15.00
  Fridays, 10.15-12.00 (**except two Fridays**)
- 3 mandatory 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!

Assignment 0: description

- Not mandatory, but highly recommended.
- A hands-on assignment, where you will be familiarized with both programming in Python and basic natural language processing.
- Do not except to understand everything! Just work your way through the examples.
- Chapter 1 of the NLTK book.

Exam

- Date: one of October 19–30 (exact day yet to be decided)
- Grade: Pass with distinction, Pass, or Fail

Some general notes about this course

- Learning to program is like learning to play an instrument or ride a bicycle: you need to practice!
- If you have experience of some other programming language, this course will be easy
- . . . but if this is the first time you program, please work hard and practice **continuously**
- Please also note that everything else in the program depends on that you properly learn the contents of this course.

Computer science crash course

- Computers are machines that solve computational problems "mechanically"
  - To do anything useful, they need detailed instructions.
- *Algorithm*: a detailed step-by-step account of how to solve a problem.
- *Programming language*: a formal language to express algorithms.

Example of an algorithm

Serves 4

- 100g slightly stale crusty white bread, soaked in cold water for 20 mins
- 1kg very ripe tomatoes, diced
- 1 ripe red pepper and 1 green pepper, deseeded and diced
- 1 medium cucumber, peeled and diced
- 2 cloves of garlic, peeled and crushed
- 150ml extra virgin olive oil
- 2tbsp sherry vinegar
- Salt, to taste
- Garnishes – see below

1. Mix the diced tomatoes, peppers and cucumber with the crushed garlic and olive oil in the bowl of a food processor or blender. Squeeze out the bread, tear it roughly into chunks, and add to the mixture.
2. Blend until smooth, then add the salt and vinegar to taste and stir well.
3. Pass the mixture through a fine sieve, then cover and refrigerate until well chilled.
4. Serve with garnishes of your choice: I liked diced black olives, hard-boiled egg and small pieces of cucumber and pepper; mint or parsley also works well, and many people add spring onion, cubes of Spanish ham and so on.

Example: long division

$$
\begin{array}{cccccc}
0&2&3&5&,4\ldots& \\ \hline
1&6&4&8& &\multicolumn{1}{|c}{7}\\ \cline{6-6}
\underline{0}&&&&&\\
1&6&&&&\\
\underline{1}&\underline{4}&&&&\\
&2&4&&&\\
&\underline{2}&\underline{1}&&&\\
&&3&8&&\\
&&\underline{3}&\underline{5}&&\\
&&&3&0&\\
&&&\underline{2}&\underline{8}&\\
&&&&\ldots&
\end{array}
$$

Example: Swedish personal identity check digit

For instance: 640823-323

1. Multiply every second digit by 2, leave the rest unchanged

$$12\ 4\ 0\ 8\ 4\ 3\ 6\ 2\ 6$$

2. Sum all the digits (note that 12 becomes 1 + 2)

$$1 + 2 + 4 + 0 + 8 + 4 + 3 + 6 + 2 + 6 = 36$$

3. The check digit is equal to 10 - d, where d is the last digit of the sum. (If this is 10, then 0).

640823-323**4**

Example: finding a name in the telephone book

- How do we find a name in the telephone book?

Example: finding a name in the telephone book

1. Open the book roughly where you think the name would be (or in the middle)
2. Found the name?
   - If yes, then we're done
   - If the name would be on this page but isn't, then it is not listed
3. Select a new place to open the book: before or after the current position
4. Repeat from 2.

Binary search more efficient than linear search!

(logarithmic complexity faster than linear)

Discussion

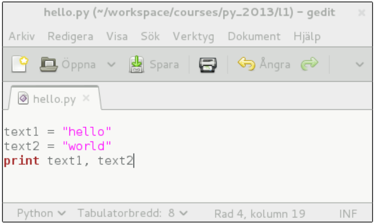
How do we find the most frequent word in today's *Göteborgsposten*?

An example of a very short program

```
print(’Hello!’)
```

Editing Python program code

- Write the Python program in a text file, e.g. `hello.py`
  - Specialized editors assist you (e.g. coloring)
    - In the lab room, the TextMate tool is installed



hello.py (~/workspace/courses/py_2013/l1) - gedit
Arkiv Redigera Visa Sök Verktyg Dokument Hjälp
Öppna
Spara
Ångra
hello.py
Python
Tabulatorbredd: 8
Rad 4, kolumn 19
INF

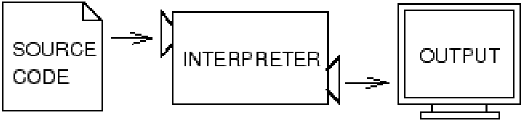
```
text1 = "hello"
text2 = "world"
print text1, text2
```

- The text that you write is called the **source code**
- The text file (conventionally ending in `.py`) is called a **source file** or a **script**

Running a Python program

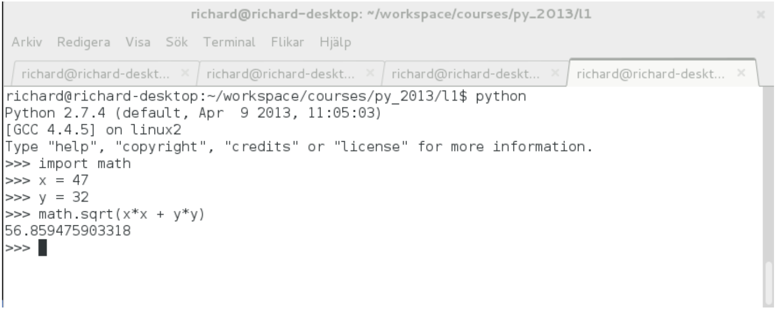
- Your code is **executed** (run) by a tool called the Python **interpreter**:

```
python hello.py
```



The Python interpreter interactively

- The interpreter can also be run *interactively*:


richard@richard-desktop: ~/workspace/courses/py_2013/l1
Arkiv Redigera Visa Sök Terminal Flikar Hjälp
richard@richard-deskt...
richard@richard-deskt...
richard@richard-deskt...
richard@richard-deskt...

```
richard@richard-desktop:~/workspace/courses/py_2013/l1$ python
Python 2.7.4 (default, Apr  9 2013, 11:05:03)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> x = 47
>>> y = 32
>>> math.sqrt(x*x + y*y)
56.859475903318
>>> 
```

Python interpreters online

- http://www.pythontutor.com/visualize.html
  - also shows state of execution
- http://www.tutorialspoint.com/execute_python3_online.php

Installing Python on your own machine

- `https://www.python.org/downloads/`
- Follow the directions for your system (e.g. Windows, mac)
- Make sure that you install a recent version of Python (e.g. 3.4.3)

Versions of Python

- The **Python 3** version was introduced in 2008
- This version introduced a number of improvements to the Python language, mainly in how you deal with text
- However, due to these changes, the community has been slow to adopt Python 3

The anatomy of another short program

```
x = 5
y = x*x + 1
print(x + y)

text = ’This is a piece of text.’
print(len(text))
```

Some Python terminology – see Downey, chapter 2

- **statement**: performs an action, such as printing a string or assigning a name to a value.
    - statements are **executed** line by line
- **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., **5** is an **integer**, **'This is a piece of text'** is a **string**. (text)
- **variables**: gives a name to a value.

When things go wrong: “bugs”

Syntax errors are when the code is broken or incomplete.

- `y = x*x +`

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

- . . . such as trying to open a file that does not exist

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

Modular programming

- To be productive, we often need to build on what others have done.
- Instead of trying to reinvent the wheel, we often use code defined by others that helps us solve a particular problem.
- **Modularity**: breaking down a program into reusable parts.
- *Python standard library* is always available with the Python program.
  - `https://docs.python.org/3/library/index.html`
- However, we will actually many times reinvent the wheel just for the practice.

The NLTK library for natural language processing

- 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.

Installing NLTK on your own computer

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

Example: most frequent word in GP

```
with open(’gp.txt’) as f:
    table = {}
    for line in f:
        for word in line.split():
            if word in table:
                table[word] += 1
            else:
                table[word] = 1
print(max(table, key=table.get))
```

Next lecture

Writing and running simple programs.

- Values, expressions, variables, types.
- Numbers, strings, lists.
- Conditions: `if` ...
- Functions.