Practical Python:

1: Introduction – from basics to NLP and ML

Dafydd Gibbon

First South African Workshop in Digital Humanities
North-Western University, Potchefstroom, SA
2015-04-04 to 2015-04-05

Practical Python: from Basics to NLP and ML

- The course is intermediate level, meaning that participants should have at least a little experience of programming.

- Why Linux? The main server application OS

- Why Python? The main back-end web server language

    1. Top 15 programming languages in 2015, in this order:
       JavaScript, Java, Python, CSS, PHP, Ruby, C++, C, Shell, C#, Objective C, R, VimL, Go, Perl

    1. Client-side processing is generally JavaScript
    2. Python is widely (but not exclusively) used for server-side processing by ...
        - Rdio, Instagram, and Pinterest, Django, Facebook, Google, YouTube, NASA, Yahoo, ..., me

- Features:
  1. all-purpose programming language
  2. high level: in general only a few lines of code are needed to formulate a function
  3. supports different programming styles:
     - linear scripting (to start with)
     - functional (for scientific computing)
     - logical (for modelling)
     - object-oriented
  4. highly extensible (extensive libraries for number-crunching, natural languge processing, imaging, speech processing, ...)
  5. dynamic types
  6. automatic memory management
  7. exception handling
  8. interpreted (on-the-fly compilation)
  9. both rapid prototyping and full software development
  10. supports web development
  11. considered easy for beginners to understand and learn

Practical Python: from Basics to NLP and ML

- Designer:
  1. Guido van Rossum
- Design philosophy:
  1. Beautiful is better than ugly
  2. Explicit is better than implicit
  3. Simple is better than complex
  4. Complex is better than complicated
  5. Readability counts

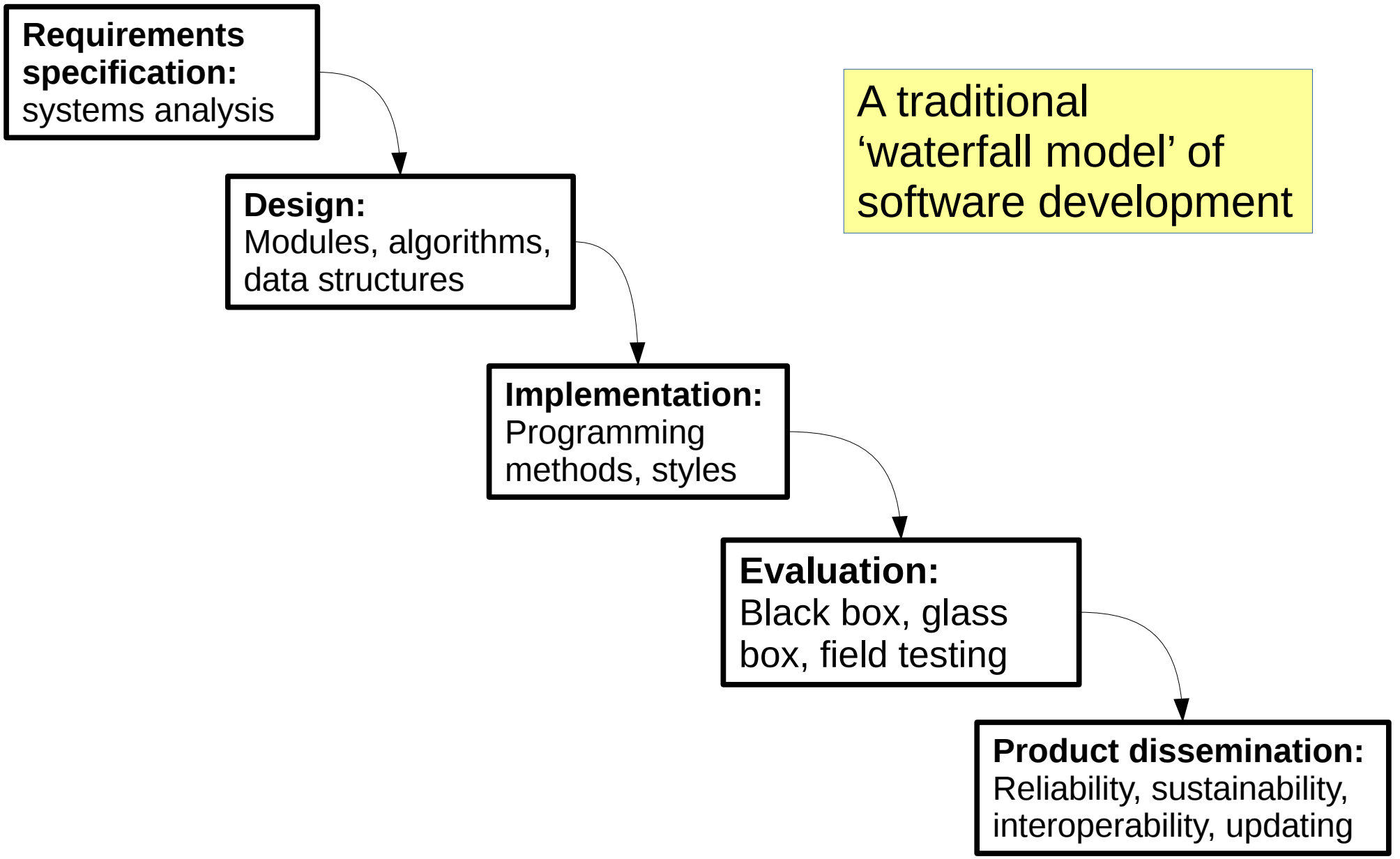
  and ...
  1. Programming is fun
     - Python is named after Monty Python
     - Idle (Integrated DeveLopment Environment) for Python is named after Eric Idle (of Monty Python)
- History:

  1989: first implementation

  2000: Python 2 (currently 2.7)

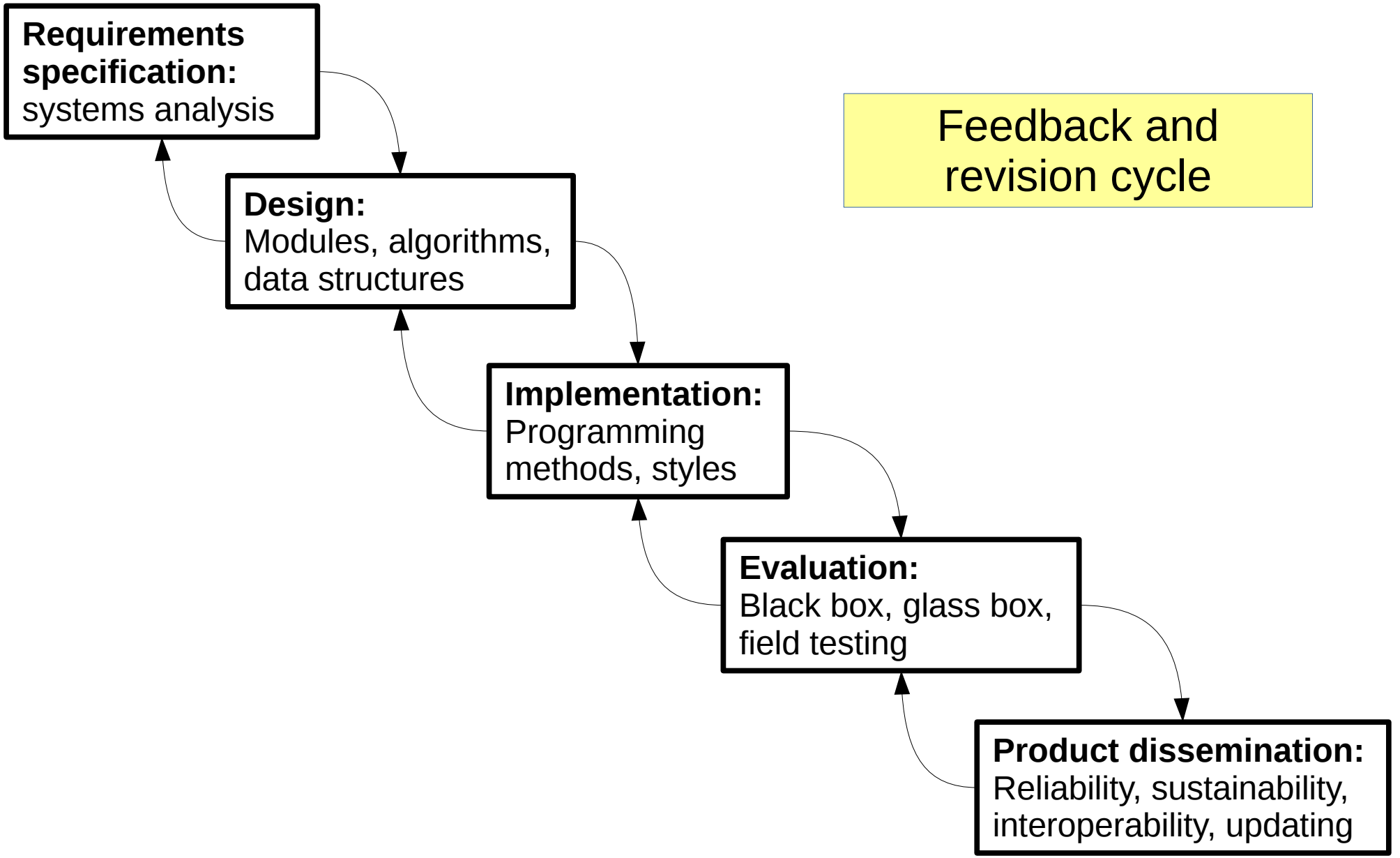
  2008: Python 3 (currently 3.5)
  - cleaner, but some backwards incompatibility

Software development vs. 'coding'!

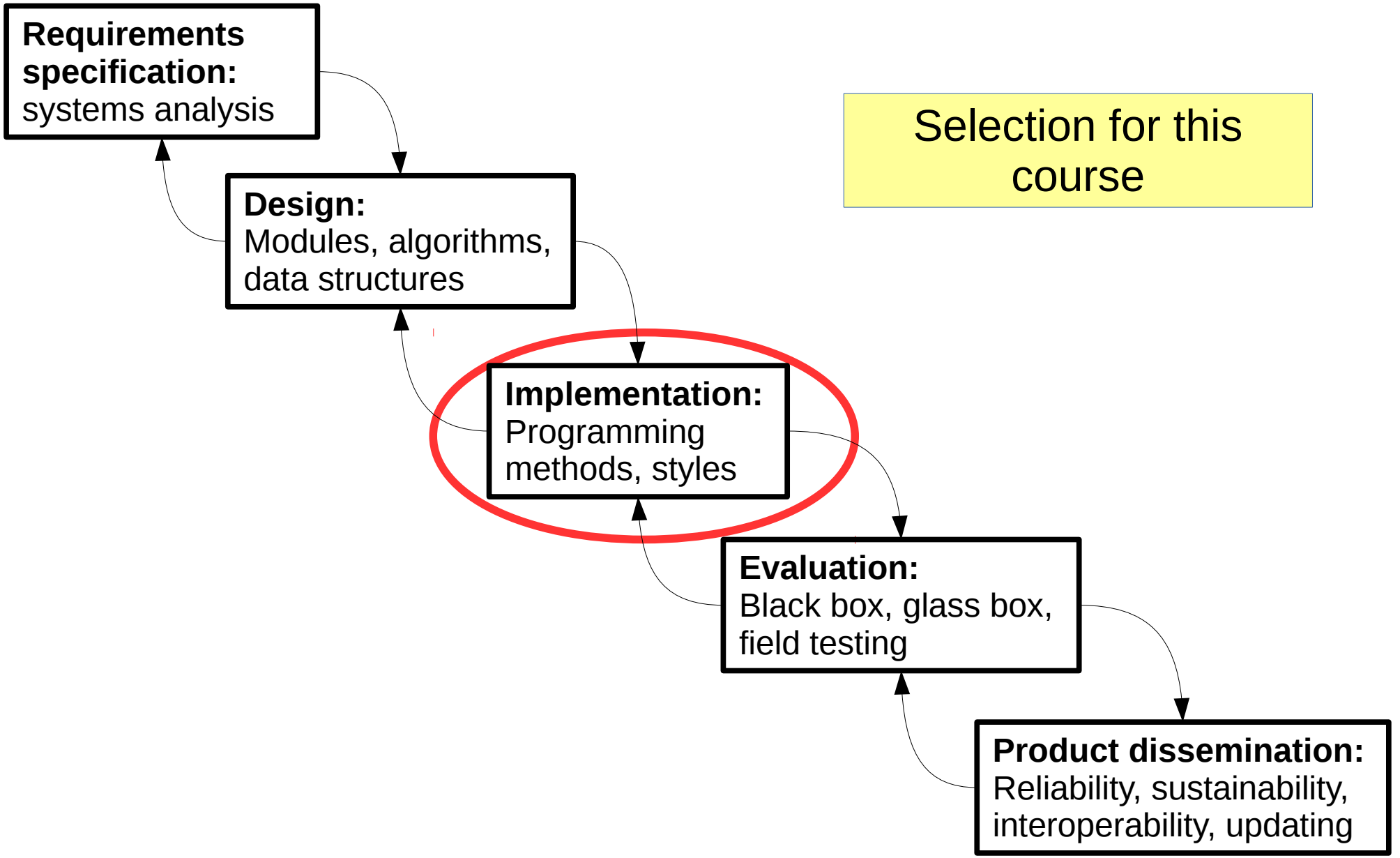


A traditional 'waterfall model' of software development

Software development vs. 'coding'!



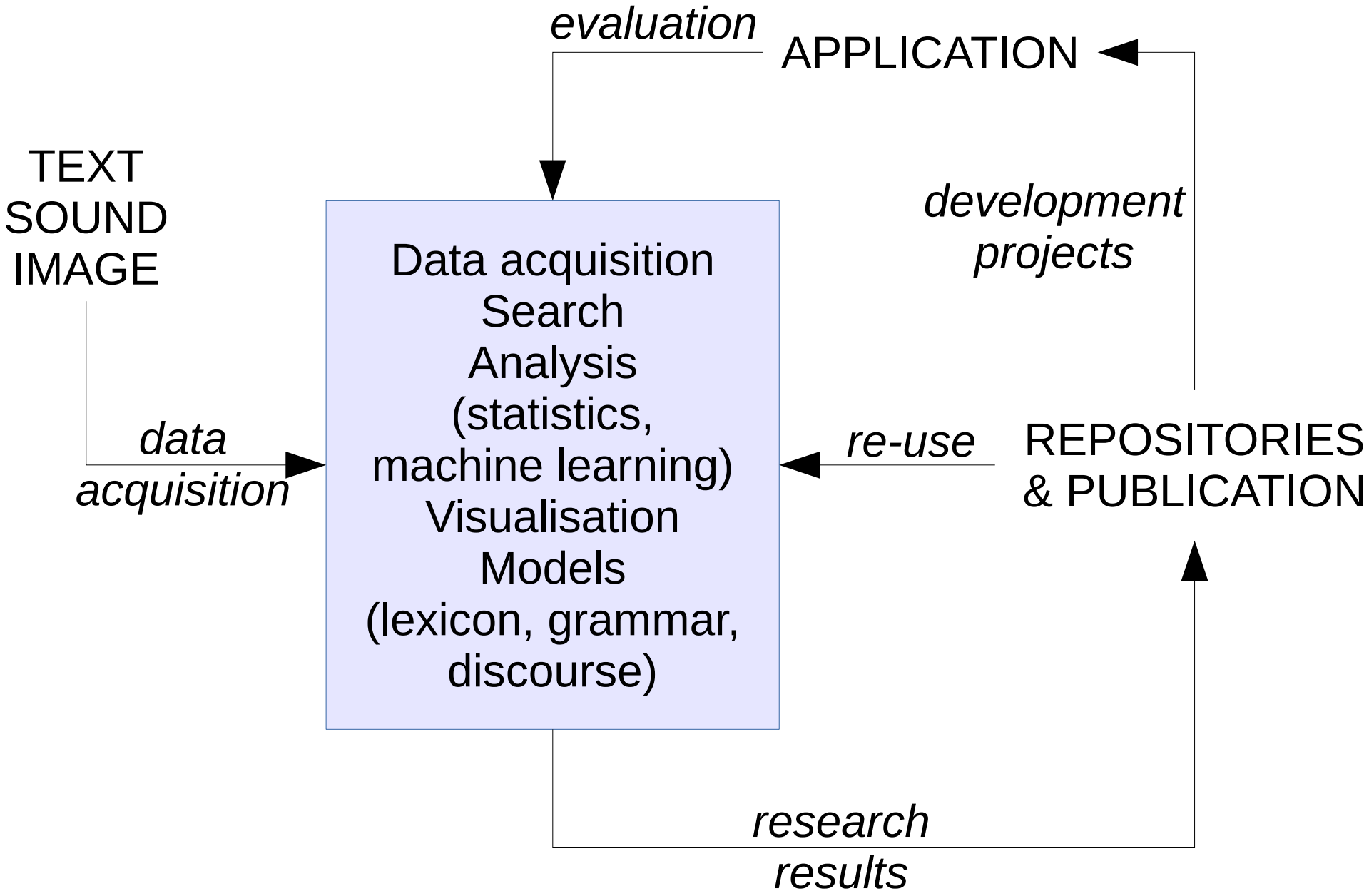
Software development vs. 'coding'!



Python based server-side web app demos

- DistGraph
  http://wwwhomes.uni-bielefeld.de/gibbon/DistGraph/
- TGA (Time Group Analyser)
  http://wwwhomes.uni-bielefeld.de/gibbon/TGA/
- FSA graph generator
  http://wwwhomes.uni-bielefeld.de/gibbon/Syllables/english-syllables-demo.html

Text-based research – resource – development cycle



Tutorial aims

The objectives of the tutorial are:

- to advance the Python knowledge and skills of participants who already have a basic understanding of programming
- to provide information and skills for problem solving in text analytics using Python, including
  - input and output of text data
  - word list and concordance creation from corpora
  - basic corpus statistics and visualisation with plot graphics
  - the Natural Language Toolkit Kit (NLTK) text analytics library
  - basics of unsupervised and supervised Machine Learning (ML)

Tutorial activities

1. Exercise environments
   - command line interface (CLI) with interactive Python shell
   - edited Python scripts run with in the command line interface (CLI)
   - edited Python modules called by Python scripts
2. Data input and output
   - reading and writing text files, string formatting
   - interaction via web interface (HTML→CGI→HTML)
3. Data types
   - strings, lists, tuples, dictionaries; plot graphics: sequence, scatter, box, histogram
4. Python operators and control structures
   - number, string, list and dictionry operators
   - conditional structures; for loops and list comprehensions
5. Python functions, modules and libraries
   - definition of functions, creation of modules, importing of modules and libraries
6. Example domains
   - text analytics, text-based machine learning

What you will need in this course

- A PC with Linux, Windows or Mac
  on a lab computer or laptop of your own

- Anaconda, recommended Python 2.7 distribution for Linux, Windows, Mac. Instructions here:
  `https://www.continuum.io/downloads`

  One of the following:
  1. Anaconda (nearly 500MB, Python with all the libraries needed for the tutorial, and more)
  2. Miniconda (25MB, Python without libraries); the **`numpy`**, **`matplotlib`**, and **`nltk`** libraries will first have to be installed with the conda installer in order to import into Python:
     `conda install numpy matplotlib nltk`

- Print or online version of “the NLTK book”

Notes on Python distributions

- Linux includes Python 'out of the box':
  1. Recommended distribution Ubuntu
  2. I use Xubuntu (Ubuntu with the XFCE user interface)
  3. Library installation using synaptic etc., or CLI:

     sudo apt-get install python-numpy python-matplotlib python-nltk
- Windows:
  1. It can be a bit of a hassle to install Python and the required libraries from different repositories
  2. Therefore the Anaconda (or Miniconda) distribution is recommended
- Anaconda and Miniconda:
  1. Anaconda contains all libraries needed and more (nearly 500MB)
  2. Miniconda requires downloading of required libraries (about 25MB)
- Python versions
  - Python 2.7: has been a long-term standard, used in the tutorial
  - Python 3.5: is more streamlined, but minor downward compatibility issues; some libraries have not been ported

(so I am sticking with 2.7 for the time being)

Python working environments

- After Python installation - basic Python environments:

  1. Python shells
     - plain CLI on Linux or Windows
     - Idle
     - IPython
     - Jupyter

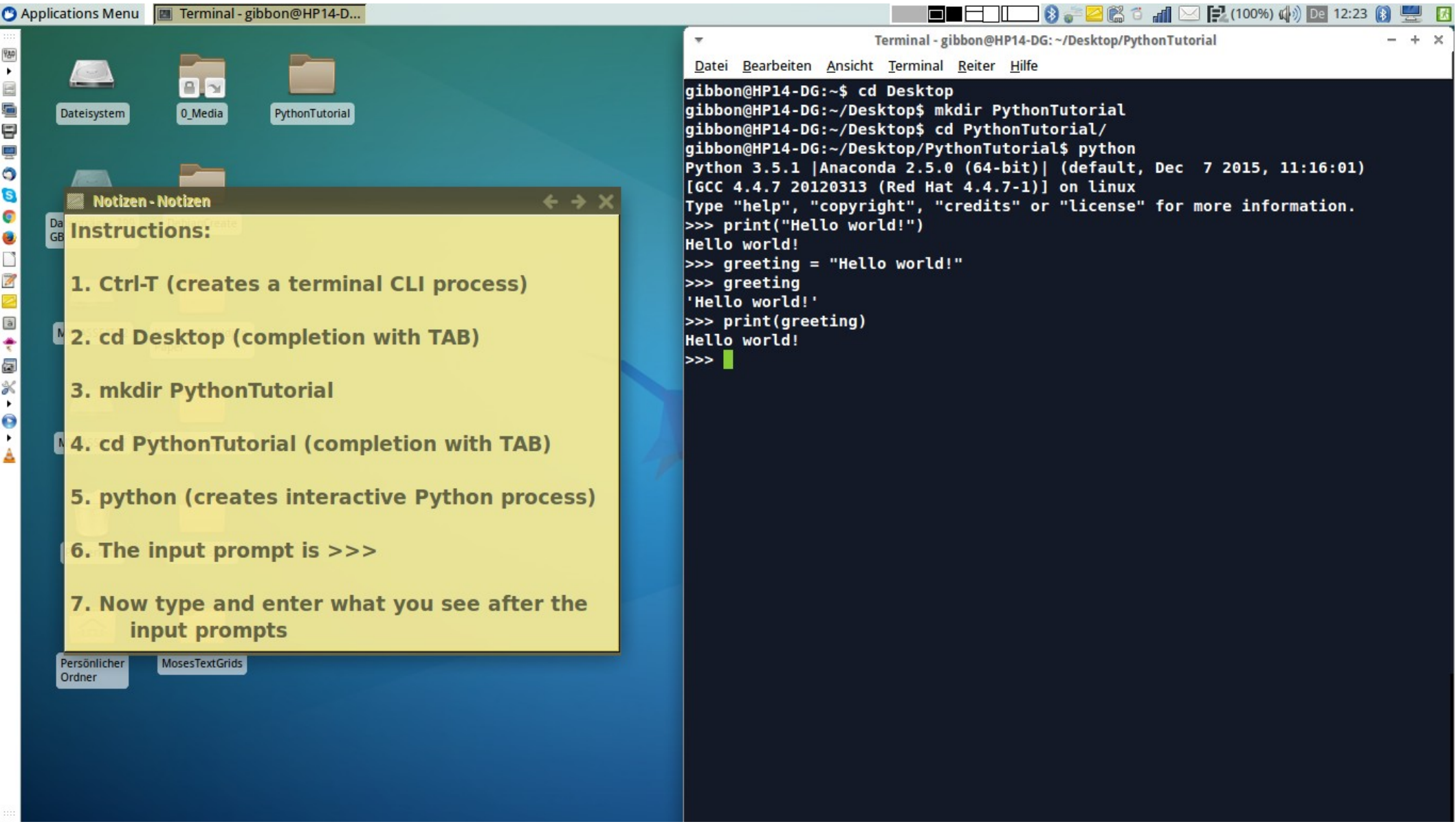
  2. Python + editor command line interface
     - Linux editors with syntax highlighting (vim, gedit, emacs, …)
     - my personal preference: via CGI with web input/output

- Python SDK – Software Development Kit:

  1. PyCharm
     - professional (paid), community (free), education (free)

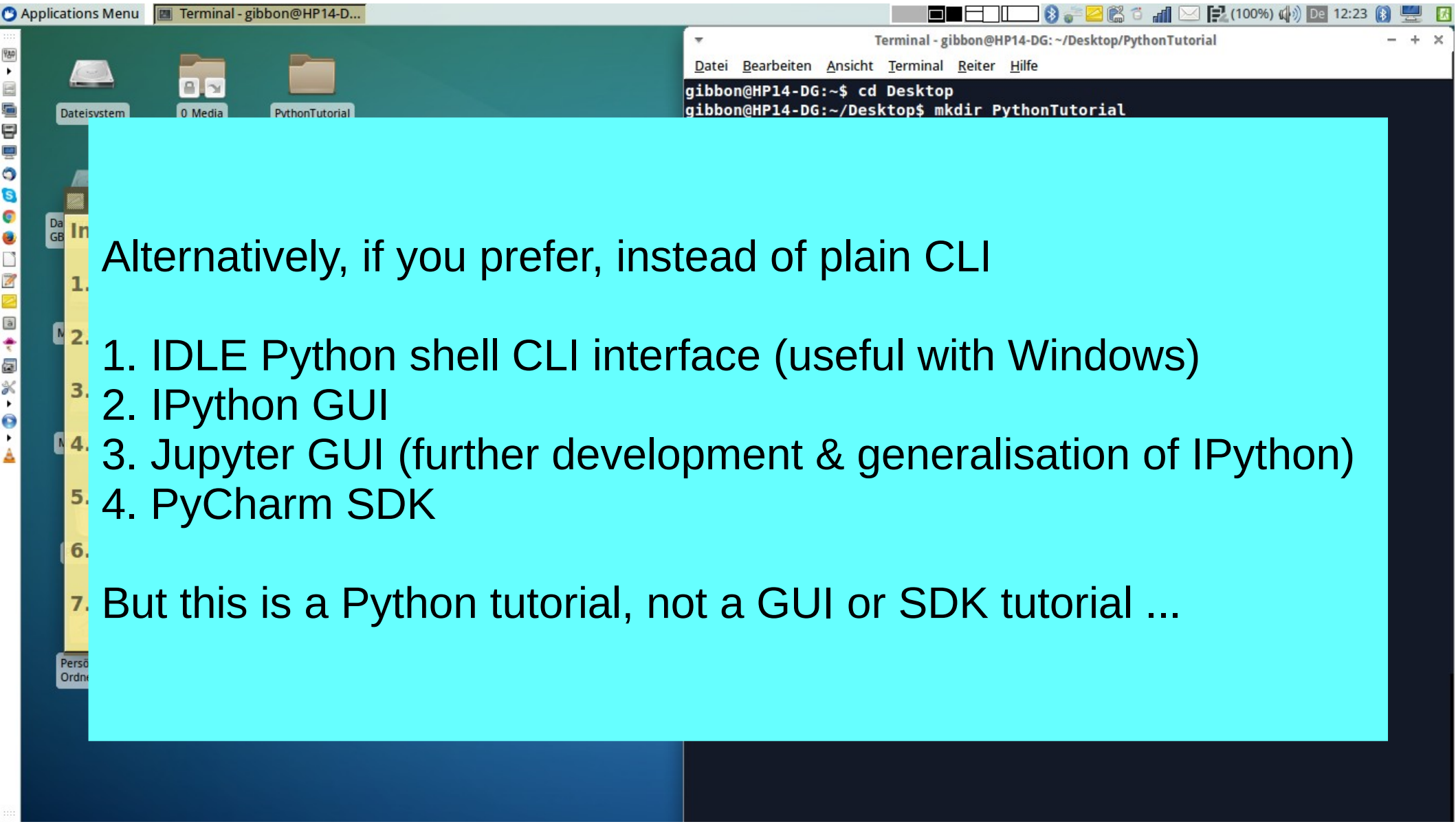
Your workspace for CLI interactive introduction

Interactive window



Your workspace for CLI interactive introduction

Interactive window



Alternatively, if you prefer, instead of plain CLI

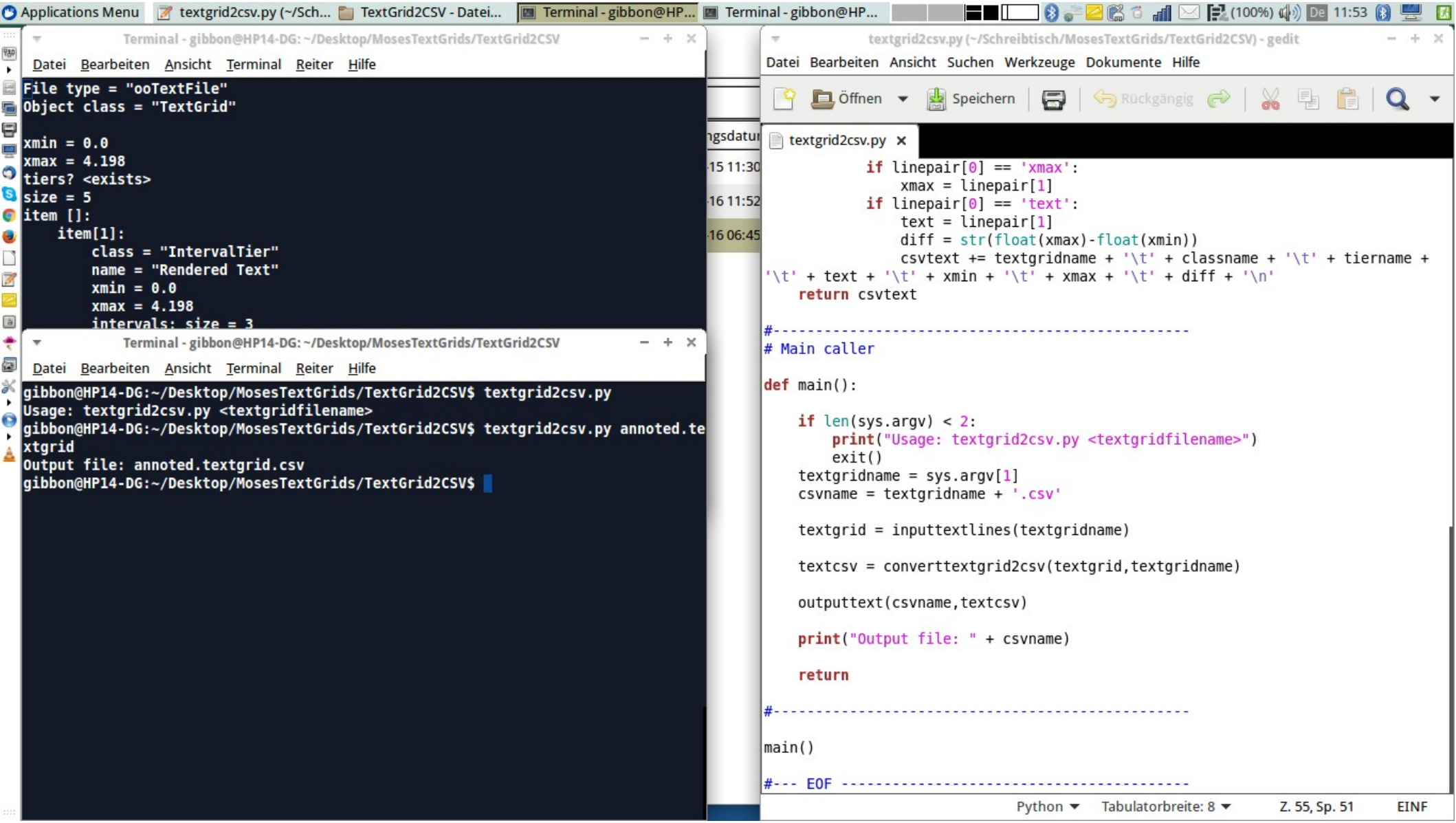
1. IDLE Python shell CLI interface (useful with Windows)
2. IPython GUI
3. Jupyter GUI (further development & generalisation of IPython)
4. PyCharm SDK

But this is a Python tutorial, not a GUI or SDK tutorial ...

A typical workspace for creating and running Python scripts

Data window

Editor window



Runtime window

Opening the Python shell with the prompt line

- Open a CLI terminal, **`mkdir`** a working directory, **`cd`** into it, and type:

```
python
```

- If you are using Anaconda, you should see:

```
Python 2.7.11 |Continuum Analytics, Inc.| (default, Dec
 6 2015, 18:08:32)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and
https://anaconda.org
>>>
```

- After the >>> prompt, type, enter and explain the lines:

```
"Hello world!"
"Hello world" + "!"
print "Hello world!"
exit()
```

Arithmetic in Python 2.7

- Note on division in Python 2.7 (not applicable to 3.5):

  If of the operands is an integer, the result is an integer with decimal places removed (not rounded). Therefore at least one integer must be a floating point number (use brackets if necessary to preserve application order).

  Another way to convert is to multiply an operand by `1.0`.

- Try the following:

```
2 / 3
3 / 3
5 / 3

float(5) / 3
5 / float(3)

1.0 * 5 / 3
5 / (1.0 * 3)
```

Arithmetic in Python 2.7

- Type, enter and explain the following:

```
100+5*2**3/4-10

100 + 5 * 2 ** 3 / 4 - 10

9**9
9 ** 9

81 ** 0.5
81 ** (½)

4 % 2
4.25 % 2
4.5 % 2f
```

Text analytics with NLTK: installing the NLTK book data

- First, the NLTK data collection for use with the NLTK book must be downloaded. Enter the **`python`** shell, and:

```
import nltk
nltk.download()
```

- The **`nltk.download()`** function creates a local database with 9 small text corpora of different kinds:
  1. a popup window appears
  2. select the row with ‘book ...’
  3. select ‘Download’
  4. be patient until the red bar is complete.
  5. finally exit Python with **`exit()`**

Text analytics with NLTK: importing the book data

- In the Python CLI environment, type:

```
from nltk.book import *
```

- You should see the following information:

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Text analytics with NLTK: using the book data

- Type, enter and explain the following lines:

```
sents()
sent1
len(sent1)
' '.join(sent1)
store = ' '.join(text1)
store
store.split()

texts()
text1
list(text1)
len(text1)
' '.join(text1)
```

- Compare `len(text1)` with `len(text2)`, ... etc.

Text analytics with NLTK: a quick KWIC

- Now type

  `text1.concordance("monstrous")`

- You should see a KeyWord In Context concordance:

```
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

- Try other words with the KWIC concordance function.

Text analytics with NLTK: a quick KWIC

- Try the previous inputs with all 9 texts

- Try the following with each text:

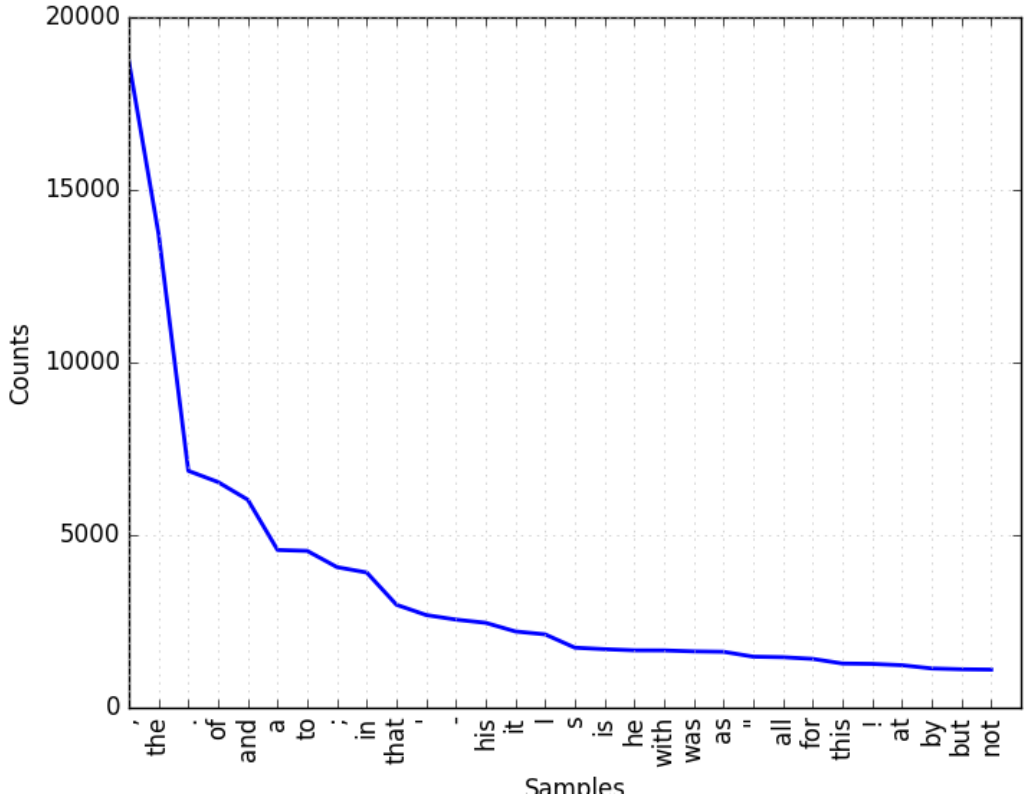
```
tokens = text1
types = set(text1)

sizetokens = float(len(tokens))
sizetypes = float(len(types))

type_token_ratio = sizetypes / sizetokens
type_token_ratio
```

Basic text analytics with NLTK: word frequency distribution

- Enter Python and type:

```
fdist1 = FreqDist(text1)
fdist1.plot(20)
```

- You should see this graph:
  (after viewing, close the window)

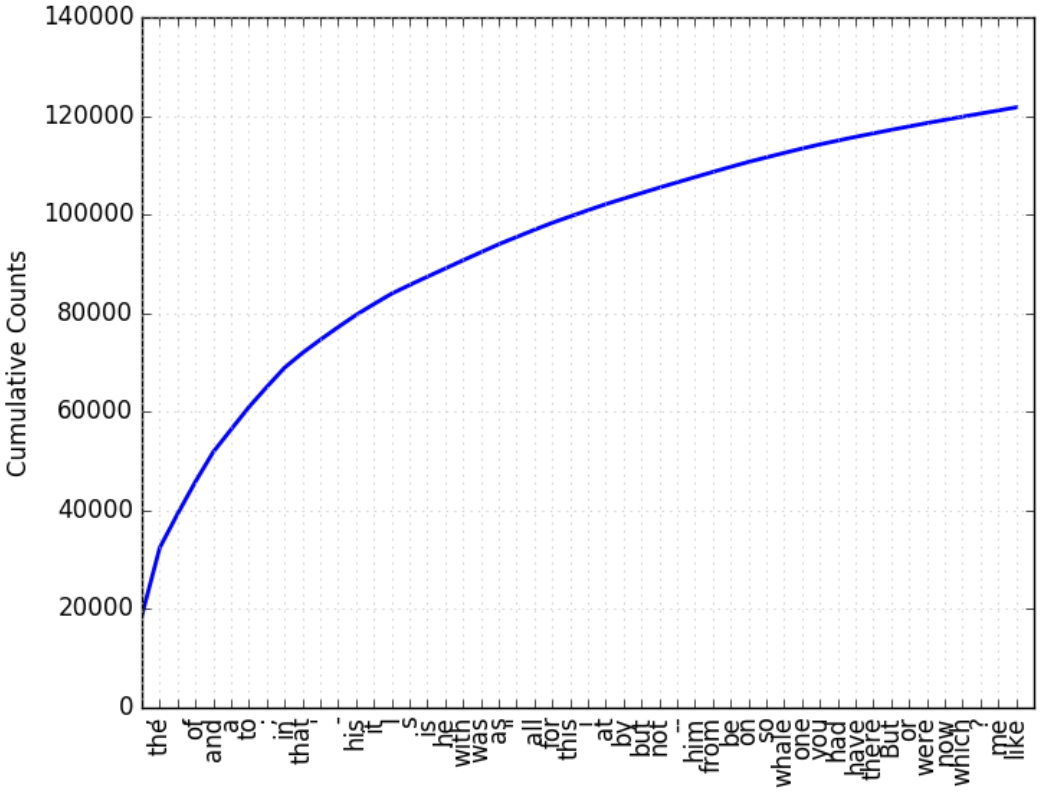


Basic text analytics with NLTK: word frequency distribution

- Type:

```
fdist1 = FreqDist(text1)
fdist1.plot(20,cumulative=True)
```

- You should see this graph:
  (after viewing, close the window)



Text analytics with NLTK: a Python function

- Type and save in a file **textstats.py** (with a text editor):

```
def printtextstats(textlist):
    eol = '\n'; cn = 4; cw = 8
    separator = '-' * cn * cw
    header = 'Title'.center(cw) + 'Types'.center(cw) +
 'Tokens'.center(cw) + 'TTR'.center(cw)
    print separator + eol + header + eol + separator
    for text in textlist:
        types = len(set(text)) ; tokens = len(text)
        typestr = str(types).rjust(cw)
        tokenstr = str(tokens).rjust(cw)
        ttr = float(types)/tokens
        ttrstr = str("{:8.4f}".format(ttr))
        title = str(text)[7:12] + '...'
        print title + typestr + tokenstr + ttrstr
    print separator
    return True
```

Text analytics with NLTK: using a Python module

- In the Python CLI environment, type:

```
import textstats as ts
from nltk.book import *
texts = [text1, text2, text3, text4, text5, text6, text7,
 text8, text9]
ts.printtextstats(texts)
```

- You should see:

```
-------------------------------
 Title   Types   Tokens   TTR
-------------------------------
Moby ...   19317  260819  0.0741
Sense...    6833  141576  0.0483
The B...    2789   44764  0.0623
Inaug...    9754  145735  0.0669
Chat ...    6066   45010  0.1348
Monty...    2166   16967  0.1277
Wall ...   12408  100676  0.1232
Perso...    1108    4867  0.2277
The M...    6807   69213  0.0983
-------------------------------
```

Text analytics with NLTK: a Python script

- Exit Python and create a second text file with the name **`textscript.py`**:

```
import textstats as ts
from nltk.book import *
texts = [text1, text2, text3, text4, text5, text6,
 text7, text8, text9]
ts.printtextstats(texts)
```

- In the CLI environment (without entering Python), type:

```
python textscript.py
```

- You should see the same table as before.
- The Python module file **`textstats.py`** defines Python objects.
- The Python script file **`textscript.py`** is actually does something with Python objects.

Typical Python idioms

Typical Python idioms

- Typical Python idioms (1):
  1. The if statement
     - conditionally executes a block of code, along with else and elif (contraction of else-if).
  2. The for and statements
     - iterates over an iterable object, capturing each element to a local variable for use by the attached block.
  3. The while statement
     - executes a block of code as long as its condition is true.
  4. The try ... except ... statement
     - allows graceful handling of exceptions raised in its attached code block to be by except clauses
     - ensures that clean-up code in a finally block will always be run regardless of how the block exits
  5. The class statement
     - executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
  6. The def statement
     - defines a function or method.

Typical Python idioms

- Typical Python idioms (2):
  1. The with statement
     - which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing RAII-like behavior.
  2. The pass statement
     - serves as a NOP. It is syntactically needed to create an empty code block.
  3. The assert statement
     - used during debugging to check for conditions that ought to apply.
  4. The yield statement
     - returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.
  5. The import statement
     - is used to import modules whose functions or variables can be used in the current program.
  6. The print statement
     - changed to the print() function in Python 3.

Typical Python idioms

- Typical Python idioms (3):

    1. List comprehensions:
        - set builder model

    2. Dictionaries:
        - (cf. association lists in languages like LISP)

    3. List slicing
        - (cf. languages like LISP and Prolog)

- Otherwise, syntax is somewhat like C, Java

    1. BUT:
        - blocks are indicated by indentation, not brackets!

Overview of Python objects for modelling DH domains

- Simple data types
  1. characters
     - 1-char strings
  2. numbers:
     - integers
     - floats
- Complex data types
  1. iterable:
     - strings
     - lists (mutable)
       - recursive
  2. non-iterable:
     - sets (mutable)
       - recursive
     - dictionaries (mutable)
       - recursive
- Functions
  1. built-in
     - pure
     - side-effects
  2. home-made
- Local modules
  1. components of applications
- Libraries
  1. *numpy*
  2. *matplotlib*
  3. *sklearn*

Official Python Tutorial:

https://docs.python.org/2/tutorial/

Further information

- You will find huge amounts of information on the internet about DH activities and tools in various disciplines.

- Check the internet for concepts introduced in the other lectures.

- For DH tools, this is an interesting page to start:
  http://dhresourcesforprojectbuilding.pbworks.com/w/page/69244314/Tutorials%20for%20DH%20Tools%20and%20Methods

- Rememember that the algorithms and data structures don't care whether they are dealing with speech, music, pictures or videos, so you will find many interesting ideas and applications concerned with solving problems from other domains.

- Some useful links:

https://www.python.org/
  - the standard reference site

https://wiki.python.org/moin/
  - comprehensive Python documentation

https://en.wikipedia.org/wiki/Python_(programming_language)
  - a compact and rather complete overview

https://www.continuum.io/why-anaconda
  - recommended comprehensive Python distribution

http://www.nltk.org/book/
  - Natural Language Toolkit: NLP library

http://scikit-learn.org/stable/
  - Machine Learning library

And there are many Python web tutorials.

Some useful references

Some further reading:

Bird, S., E. Klein, and Loper, E. 2009. Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit. O'Reilly Media, <http://www.nltk.org/book>

Church, K.W. UnixTM for Poets (check the internet).

Jurafsky, Daniel, and James H. Martin. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 2nd edition. Prentice-Hall.

Mitkov, Ruslan, ed. 2002. The Oxford Handbook of Computational Linguistics. Oxford University Press.

Also check for the ‘Python Cookbook’ and the ‘NLTK Cookbook’.

End of Introduction