ΓΛΩΣΣΙΚΗ ΤΕΧΝΟΛΟΓΙΑ

Python & NLTK: Εισαγωγή

Εισαγωγή

Γιατί Python?

Παρουσίαση NLTK

Πηγές και χρήσιμα εργαλεία

Φροντιστήριο σε Python

- Στο φροντιστήριο:
  - Εισαγωγή στην Python
  - Ζητήματα προγραμματισμού για επεξεργασία κειμένων
  - Παρουσίαση της NLTK πλατφόρμας
- Έκδοση Python:
  - Το φροντιστήριο θα γίνει σε Python 2.X.X
  - Η έκδοση που θα έχετε εσείς πρέπει να υποστηρίζει NLTK:

"NLTK requires Python versions 2.5-2.7."

Γιατί Python;

- Εύκολη! Θα τη μάθετε αμέσως.
- Χρειάζεται να γράψετε πολύ λιγότερο κώδικα. (Ο χρόνος development είναι 10 φορές μικρότερος)
- Είναι scripting, παρόλα αυτά αρκετά γρήγορη. (Implemented in C)
- Ο κώδικας σε Python είναι μικρότερος και πιο «καθαρός», εύκολος να διαβαστεί και να κατανοηθεί. (Τα blocks κώδικα ορίζονται από κενά)
- Cross-Platform: Μπορείτε να προγραμματίσετε σε Windows ή Linux
- Υπάρχουν πολλά και δωρεάν διαθέσιμα resources στο δίκτυο για να διαβάσετε.
- NLTK (Natural Language Toolkit): πλατφόρμα με έτοιμα εργαλεία επεξεργασίας φυσικής γλώσσας

Python

- High Level
- Scripting
- Elegant Syntax
- Interpreted
- Object Oriented
- Functional
- Dynamic Typing
- Automatic Memory Management

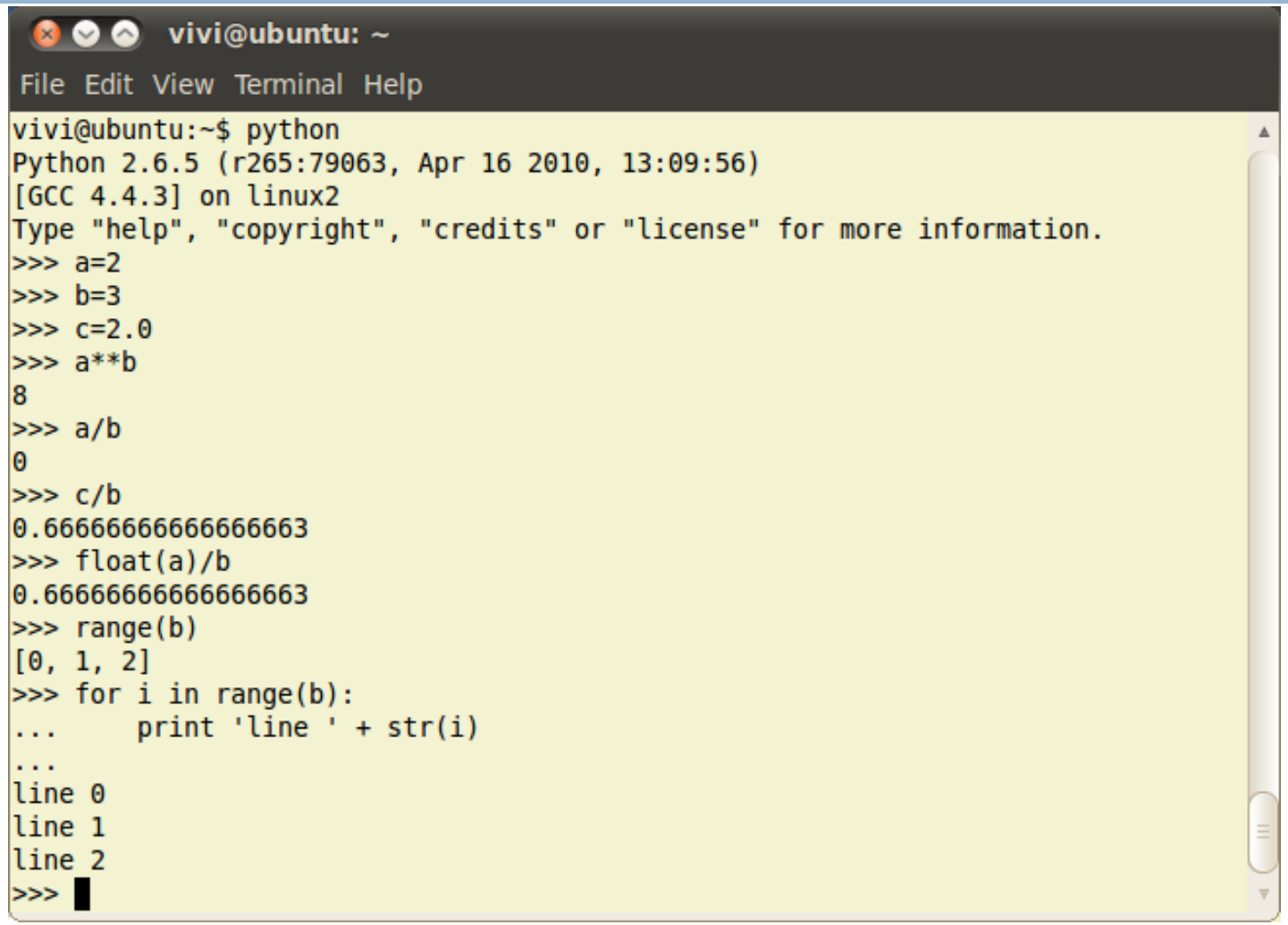
My First Program

print "Hello World!"

Αντί για:

```
#include <stdio.h>
int main(int argc, char** argv)
{
        printf("HelloWorld!\n");
}
```

Interactive Mode



Πηγές για Python

- Python Documentation
  - http://docs.python.org/tutorial/index.html
- Dive into Python
  - http://diveintopython.org/
- Ελληνική κοινότητα προγραμματιστών Python
  - http://python.org.gr
- effbot.org
  - http://effbot.org/
- Google
  - http://www.google.com

Editors

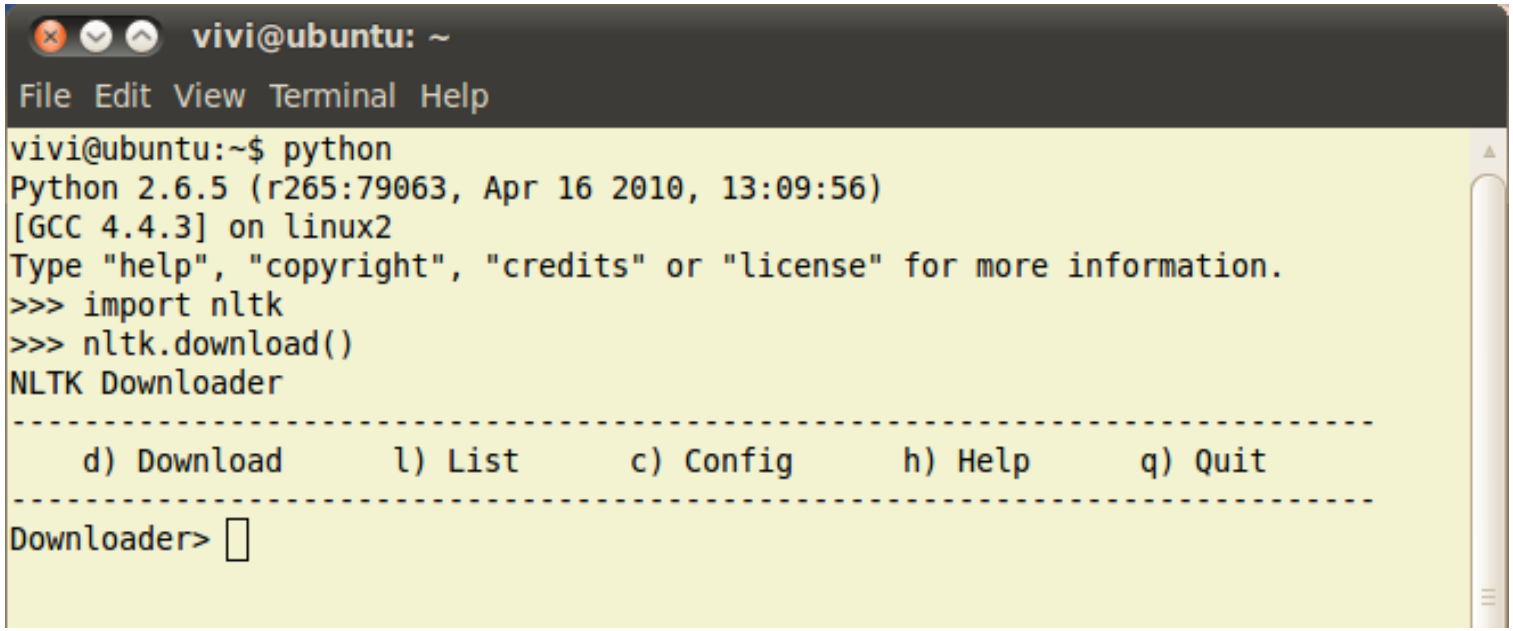
- Editors
  - Windows
    - Notepad etc.
  - Linux
    - Gedit etc.
- IDEs
  - Windows
    - Active Python
    - Netbeans
    - …
  - Linux
    - Eclipse
    - Netbeans
    - …

NLTK – Natural Language Toolkit

- Πακέτο βιβλιοθηκών και προγραμμάτων της Python για εφαρμογές Επεξεργασίας Φυσικής Γλώσσας.
- Χρησιμοποιείται ευρύτατα ως ερευνητικό εργαλείο στο πεδίο της υπολογιστικής γλωσσολογίας
- Περιλαμβάνει πολλά γνωστά corpora
- Πρέπει να το εγκαταστήσετε χωριστά
- http://www.nltk.org/
  - Download του NLTK & οδηγίες για εγκατάσταση
- http://www.nltk.org/book
  - το βιβλίο “Natural Language Processing with Python”
  - Περιλαμβάνει περιγραφή όλων των διαθέσιμων εργαλείων

Πρόσβαση στα resources

- Το NLTK με την εντολή download δίνει τη δυνατότητα εγκατάστασης διάφορων resources



Διαθέσιμα resources

- Μέρος της λίστας των διαθέσιμων:

```
Hit Enter to continue:
  [*] toolbox.............. Toolbox Sample Files
  [*] treebank............. Penn Treebank Sample
  [*] udhr................. Universal Declaration of Human Rights Corpus
  [*] unicode_samples..... Unicode Samples
  [*] webtext.............. Web Text Corpus
  [*] wordnet.............. WordNet
  [*] wordnet_ic........... WordNet-InfoContent
  [*] words................ Word Lists
  [*] book_grammars....... Grammars from NLTK Book
  [*] ycoe................. York-Toronto-Helsinki Parsed Corpus of Old
                            English Prose
  [*] basque_grammars..... Grammars for Basque
  [-] large_grammars...... Large context-free and feature-based grammars
                            for parser comparison
  [*] sample_grammars..... Sample Grammars
  [*] spanish_grammars.... Grammars for Spanish
  [*] tagsets.............. Help on Tagsets
  [*] maxent_treebank_pos_tagger Treebank Part of Speech Tagger (Maximum entropy
)
  [*] rslp................. RSLP Stemmer (Removedor de Sufixos da Lingua
                            Portuguesa)
  [*] hmm_treebank_pos_tagger Treebank Part of Speech Tagger (HMM)
Hit Enter to continue:
```

Βιβλίο

- “Natural Language Processing with Python”
- Μπορείτε να εγκαταστήσετε τις πηγές του βιβλίου:

```
Collections:
  [-] all-corpora..... All the corpora
  [-] all............. All packages
  [-] book............ Everything used in the NLTK Book

([*] marks installed packages; [-] marks out-of-date or corrupt packages)

---------------------------------------------------------------------------
    d) Download      l) List      c) Config      h) Help      q) Quit
---------------------------------------------------------------------------
Downloader> d

Download which package (l=list; x=cancel)?
  Identifier> book
```

Βασικά στοιχεία της γλώσσας

Επισκόπηση κύριων χαρακτηριστικών

Native Datatypes

Γενικά Χαρακτηριστικά Ι

- Data Types
  - boolean, integer, float, long, complex
  - Δεν ορίζονται ρητά. Καθορίζεται ο τύπος στον χρόνο εκτέλεσης.
    - Προσοχή!!
- Sequence Types
  - string, list, tuple
- Mapping Types
  - dictionary
- Set Types
  - set

Γενικά Χαρακτηριστικά II

- Control Flow
  - if, for, while, break, continue
- Functions
  - def foo():
- Classes
  - class foo():
- Modules
  - from *module* import *something*
  - import *module*

Native Datatypes – Lists #1

- Η λίστες είναι ο τύπος δεδομένων που χρησιμοποιείται περισσότερο στην Python
- Είναι arrays, των οποίων το μέγεθος αλλάζει δυναμικά όσο προστίθενται στοιχεία.
- Τα στοιχεία δεν είναι απαραίτητο να είναι του ίδιου τύπου δεδομένων!

Ορισμός Λίστας

```
>>>a = ['spam', 'eggs', 100, 1234]
>>>a
['spam', 'eggs', 100, 1234]
```

Native Datatypes – Lists #2

Αναφορά στα στοιχεία της λίστας

•Index
a[0],a[1],…,a[len-2], a[len-1]
•Negative index
a[-len],a[-(len-1)],…, a[-1]

Προσθήκη στοιχείων

```
>>>a.append(333)
>>>a
['spam', 'eggs', 100, 1234, 333]
>>>a.insert(2,-1)
>>>a
['spam', 'eggs', -1, 100, 1234, 333]
>>>a.extend([1,2])
>>>a
['spam', 'eggs', -1, 100, 1234, 333, 1, 2]
```

Native Datatypes – Lists #3

Διαγραφή στοιχείων

```
>>>a.pop()
>>>a
['spam', 'eggs', -1, 100, 1234]
>>>a.remove(-1)
>>>a
['spam', 'eggs', 100, 1234]
>>>del a[1]
>>a
['spam', 100, 1234]
```

Αναζήτηση

```
>>>'spam' in a:
True
>>>'cat' in a:
False
```

Native Datatypes – Lists #4

Απαρίθμηση

```
>>>for i in a:
...       print i
...
'spam'
100
1234
>>>for i in range(len(a)):
...        print i, a[i]
…
0,'spam'
1,100
2,1234
```

List Comprehensions

```
>>> a = [1,2,3,4,5]
>>> b = [6,7,8]
>>> [x*2 for x in a]
[2, 4, 6, 8, 10]
>>> [x*3 for x in a if x > 2]
[9, 12, 15]
>>> [x+y for x in a for y in b]
[7, 8, 9, 8, 9, 10, 9, 10, 11, 10, 11, 12, 11, 12, 13]
>>> [x for x in a if x in b]
[]
>>> [x+y for x in a for y in b if x+y<10]
[7, 8, 9, 8, 9, 9]
>>>
```

Native Datatypes - Dictionaries #1

- Τα dictionaries ορίζουν σχέσεις μεταξύ κλειδιών και τιμών.
- Πρόκειται για συλλογές εγγραφών που αποτελούνται από ένα κλειδί και την τιμή που αντιστοιχεί σε αυτό.

Ορισμός Dictionary

```
>>>tel = {'jack': 4098, 'sape': 4139}
>>>tel
{'jack': 4098, 'sape': 4139}
```

Αναφορά στα στοιχεία

```
>>>tel['jack']
4098
```

Native Datatypes - Dictionaries #2

Προσθήκη στοιχείων

```
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

Διαγραφή στοιχείων

```
>>> del tel['sape']
>>>tel
{'guido': 4127, 'jack': 4098}
```

Native Datatypes - Dictionaries #3

Αναζήτηση

```
>>>'guido' in tel
True
>>>'sape' in tel
False
```

Απαρίθμηση

```
>>>for k,v in tel.items():
...             print k,v
...
'guido', 4127
'jack', 4098
```

Native Datatypes – Tuples #1

- Μια tuple είναι μια λίστα που δεν αλλάζει.
- Η τιμές που περιέχει μια tuple δεν μπορούν να αλλάξουν μετά τη δημιουργία της.

Ορισμός Tuple

```
>>> t = ("Mon", "Tue", "Thu", "Wed", "Fri", "Sat", "Sun")
>>> t
('Mon', 'Tue', 'Thu', 'Wed', 'Fri', 'Sat', 'Sun‘)
```

Αναφορά στα στοιχεία της tuple

•Index

t[0],t[1],…,t[len-2], t[len-1]

•Negative index

t[-len],t[-(len-1)],…, t[-1]

Native Datatypes – Sets #1

- Τα sets είναι μια αταξινόμητη συλλογή μοναδικών αντικειμένων
- Χρησιμοποιούνται συχνά για αφαίρεση διπλοτύπων και για μαθηματικές πράξεις συνόλων

Ορισμός set

```
>>> aset=set([2,4,5,6,7,12,23,45])
>>> aset
set([2, 4, 5, 6, 7, 12, 45, 23])
>>> aset=set([2,4,5,6,7,12,23,45,2])
>>> aset
set([2, 4, 5, 6, 7, 12, 45, 23])
```

Native Datatypes – Sets #2

Προσθήκη στοιχείων

```
>>> aset
set([2, 4, 5, 6, 7, 12, 45, 23])
>>> aset.add(1)
>>> aset
set([1, 2, 4, 5, 6, 7, 12, 45, 23])
>>> aset.add(4)
>>> aset
set([1, 2, 4, 5, 6, 7, 12, 45, 23])
```

Διαγραφή στοιχείων

```
>>> aset
set([1, 2, 4, 5, 6, 7, 12, 45, 23])
>>> aset.remove(1)
>>> aset
set([2, 4, 5, 6, 7, 12, 45, 23])
```

Native Datatypes – Sets #3

Αναζήτηση

```
>>> aset
set([2, 4, 5, 6, 7, 12, 45, 23])
>>> 2 in aset
True
>>> 9 in aset
False
```

Απαρίθμηση

```
>>> aset
set([2, 4, 5, 6, 7, 12, 45, 23])
>>> for i in aset:
...     print i
...
2
4
5
κ.ο.κ.
```

Native Datatypes – Sets #4

Πράξεις Συνόλων

```
>>> aset=set([5,3,45,16])
>>> bset=set([9,2,16,3,25])
>>> aset.union(bset)
set([2, 3, 5, 9, 45, 16, 25])
>>> aset.intersection(bset)
set([16, 3])
>>>
```

Ζητήματα σχεδιασμού

Δομές Δεδομένων

Βέλτιση χρήση των διαθέσιμων τύπων

Επιλογή δομής δεδομένων

- Κριτήρια:
  - Μέγεθος του προβλήματος
  - Πως θα χρησιμοποιηθεί
- Ενέργειες που καθορίζουν το κόστος:
  - Lookup: αναζήτηση/έλεγχος ύπαρξης δεδομένων στη δομή.
  - Insert: εισαγωγή δεδομένων στη δομή.

Συνήθεις κλάσεις

- Οι νεότερες γλώσσες προγραμματισμού προσφέρουν συνήθως δύο βασικές κλάσεις:
  - Dictionary
    - Δυναμικού μεγέθους.
    - Κάθε εγγραφή είναι του τύπου <key, value>.
    - Το κλειδί εισάγεται σε hashtable.
    - Οι εγγραφές δεν ταξινομούνται.
  - List:
    - Δυναμικού μεγέθους.
    - Κάθε εγγραφή είναι μόνο value.
    - Δεν υπάρχει οργάνωση.
    - Υποστηρίζει κλήσεις ταξινόμησης.

Υπέρ-Κατά

- Dictionaries:
  - Πολύ γρήγορα lookups.
  - Πιο αργή προσθήκη εγγραφών (hashing process).
  - Για να ταξινομηθεί πρέπει να εξαχθεί η λίστα των κλειδιών.
- Lists:
  - Αργά lookups. Συνήθως γίνεται σειριακή αναζήτηση.
  - Γρήγορη προσθήκη εγγραφών.
  - Μπορεί να ταξινομηθεί.

Τι χρησιμοποιούμε?

- Κριτήρια:
  - Πόσες θα είναι οι εγγραφές? Πχ για < 10 δεν αξίζει το κόστος του hashing.
  - Πόσο συχνά θα ψάχνουμε στη δομή? Πχ αν σε κάθε insert αντιστοιχεί και ένα lookup ή περισσότερο το hashing συμφέρει.
  - Θα ταξινομήσουμε? Αν και οι περισσότερες γλώσσες υποστηρίζουν απευθείας μετατροπή των κλειδιών του Dictionary σε λίστα για να γίνει ταξινόμηση.
- Πως επιλέγουμε?
  - Μαντεύοντας. Ο γενικός κανόνας είναι “rule of thumb”. Κάθε πρόβλημα είναι διαφορετικό και μέχρι να το δεις να δουλεύει δεν ξέρεις με σιγουριά.