GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Lecture 2: Functions and expressions

Introduction for Linguists (LT2102)

Markus Forsberg
Språkbanken
University of Gothenburg

September 13, 2010

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

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.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

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.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

FAQ: How do I install NLTK on my own computer?

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

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

FAQ: floating point division

- 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.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

FAQ: floating point division (cont.)

```
>>> 1.0/6
0.16666666666666666

>>> x = 1

>>> y = 6

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

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

FAQ: how do I print the whole Text?

- `Text` is NLTK-specific (Text is a class, more about that later).
- `Text` supports slicing:

```
>>> from nltk.book import *
...
>>> text1[0:len(text1)]
...
>>> text1[0:]
...
>>> text1[:]
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

FAQ: What is the ‘u’ in front of a string?

- u = Unicode
- `u'hello'` is a Unicode string.
- Unicode is needed to be able to represent non-English alphabets.
- More in lecture ‘Strings and documentation’.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Programming = debugging

- Since a programming language is extremely picky with details, you will more often than not get some error.
- Learning to program is, in many ways, to learn to understand what went wrong, and what needs to be done to fix it.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Strings

- Strings are used to represent text data.
- Example: `'Python programming'`, `"Python programming"`.
- Strings are *immutable*, i.e., they cannot be changed, instead must a new one be created.
- There are many convenient functions in Python for Strings: `split`, `join`, `replace`, `lower`, and more.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Lists

- Lists are used to represent sequences of data, e.g., a token list.
- Example: `['Python', 'programming']`
- Lists are *mutable*, i.e., they can be changed, .e.g., `x[0] = 'tree'`.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Procedure and Functions

```
def NAME(PARAMETERS):
    BODY
    (return EXPRESSION)
```

Procedure:

```
def print_twice(s):
    print s
    print s
>>> print_twice('hello')
hello
hello
```

Function:

```
def area(length, width):
    return length*width
>> print area(5,10)
50
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Conditionals

```
if CONDITION:
    CODE
elif CONDITION:
    CODE
else:
    CODE
```

```
def greater_than_hundred(n):
    if n > 100:
        print "n is greater than 100"
    elif n < 100:
        print "n is lower than 100"
    else:
        print "n is hundred"
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Numerical condition

<	less than
<=	less than or equal to
==	equal to (note this is two "=" signs, not one)
!=	not equal to
>	greater than
>=	greater than or equal to

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Combining conditions

- **not** *CONDITION*
- *CONDITION1* **and** *CONDITION2*
- *CONDITION1* **or** *CONDITION2*

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Some string condition

s.startswith(t)	test if s starts with t
s.endswith(t)	test if s ends with t
t in s	test if t is contained inside s
s.islower()	test if all cased characters in s are lowercase
s.isupper()	test if all cased characters in s are uppercase
s.isalpha()	test if all characters in s are alphabetic
s.isalnum()	test if all characters in s are alphanumeric
s.isdigit()	test if all characters in s are digits
s.istitle()	test if s is titlecased (all words in s have have initial capitals)

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Looping: for loop

```
for NAME in ITERATOR:
    BODY
```

- Almost everything in Python that you want to traverse is an iterator: lists, sets, strings, and more.

```
>>> for x in 'loop':
...     print x
...
l
o
o
p

>>> for x in ['the','dog','barked']:
...     print x
...
the
dog
barked
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Modules

- A module is a collection of related functions.
- We use `import module` to import a module.
- Functions (and more) is available through dot notation: `module.function`.

```
>>> import math
>>> math
<module 'math' (built-in)>
>>> math.cos(math.pi)
-1.0
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Modules cont.

- `from module import *` allows us to skip the dot notation, and just write `function`.
- In general, this is not a good idea, since:
  - the origin of `function` is not visible.
  - it increases the risk that we get accidental name collisions.

```
a1.py: a = 1
a2.py: a = 2
>>> from a1 import *
>>> from a2 import *
>>> a
2
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

FAQ: working with your own text in NLTK

```
"A module for importing text into a nltk.Text."
import nltk

def read_text(filename):
    "Convert a text file into a nltk.Text"
    with open(filename) as f:
        text = f.read()
    tokens = text.split()
    return nltk.Text(tokens)

def read_text_improved(filename):
    """Convert a text file into a nltk.Text with improved
    tokenization."""
    with open(filename) as f:
        text = f.read()
    tokens = []
    for s in nltk.sent_tokenize(text):
        s_tokens = nltk.word_tokenize(s)
        tokens.extend(s_tokens)
    return nltk.Text(tokens)

text_split = read_text('sherlock_holmes.txt')
text_imp = read_text_improved('sherlock_holmes.txt')
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Case study: interface design

- The rest of the lecture will be spent on Chapter 4 in Downey.
- http://www.greenteapress.com/thinkpython/html/book005.html