GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Lecture 3: Control and recursion

Introduction for Linguists (LT2102)

Markus Forsberg
Språkbanken
University of Gothenburg

September 14, 2010

Question: NLTK, chapter 1, question 23

- *Review the discussion of looping with conditions in Section 1.4. Use a combination of for and if statements to loop over the words of the movie script for Monty Python and the Holy Grail (text6) and print all the uppercase words, one per line.*

```
for word in text6:
    if word.isupper():
        print word
```

Question: NLTK, chapter 1, question 28

- *Define a function percent(word, text) that calculates how often a given word occurs in a text, and expresses the result as a percentage.*

```
def percent(word,text):
    word_count  = text.count(word)
    text_length = len(text)
    return 100*float(word_count)/text_length
```

Conditionals: pitfall

```
>>> if 'tree'.isupper:
...     print "strange, 'tree' is not uppercase."
...
strange, 'tree' is not uppercase.

>>> 'tree'.isupper
<built-in method isupper of str object at 0x7fc4bd90de10>

>>> 'tree'.isupper()
False
```

Local variables

- Variables defined in a function, is *local* to that function.

```
name = 'global'

def function():
     name = 'local'

>>> function()
>>> name
'global'
```

- Calling `function()` does not change the value of the *global* `name`.

Local variables cont.

- However, we can change the content of mutable types, such as lists.

```
name = ['global','global']

def function():
    name[0] = 'local'

>>> function()
>>> name
['local', 'global']
```

- But we are not able to change what `name` refers to.

tuples

- An immutable data structure used to group values.
- Works like a list except that we have no way of changes the values.

```
>>> name = (1,2)

>>> type(name)
<type 'tuple'>

>>> (m,n) = name

>>> m
1
>>> n
2
```

Control and Recursion

- **Control** is about what is available to control the flow of execution, e.g., conditionals and loops.
- **Recursion** is a way of defining a function where its definition includes a call to itself.
- Recursion can be thought of as a kind of loop.

While loop

- We use `while` when we want to loop based on a condition.

```
while CONDITION:
    CODE

>>> x = 0
>>> while x < 5:
...     print x
...     x = x + 1
...
0
1
2
3
4
```

- If you are not careful with the condition, you end up with an infinite loop.

break and continue

- We use `break` to exit a loop, and `continue` to skip a step.

```
>>> x = 10
>>> while True:
...     x = x-1
...     if x == 0:
...         break
...
>>> x
0

>>> for x in range(10):
...     if x%2 == 0:
...         continue
...     print x
...
1
3
5
7
9
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Calling a function in another function

- Every time a function is called, a *function frame* is created containing local variables and parameters.
- When we call another function, we put the current function frame on a `stack`.
- Think of it as a stack of notes, where the top note describe the function we are currently at.

GÖTEBORGS UNIVERSITET

Språkbanken

CLT

Recursion

- A recursive function is a function that calls itself.
- A recursive function needs a *termination condition*, a *base case*, or we get infinite recursion.

```
def countdown(n):
    if n <= 0:
        print 'Blastoff!'
    else:
        print n
        countdown(n-1)
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Recursion and mathematics

- Mathematical formulas are typically defined recursively, e.g., factorial, `n!`:
  $0! = 1$
  $n! = n * (n - 1)!$

```
def factorial(n):
    if n==0:
        return 1
    else:
        return n*factorial(n-1)
```

Python and Recursion

- Recursion is not well supported by Python, the limit of the number of function frames on the stack is by default 1000.

```
>>> factorial(999)
...
RuntimeError: maximum recursion depth exceeded
```

- Instead, we mostly use loops in Python.
- (for the interested: the limit can be changed with `sys.setrecursionlimit`).

GÖTEBORGS UNIVERSITET

Språkbanken

CLT

Factorial as a loop

```
def factorial(number):
    result = 1
    for n in range(1,number+1):
        result = n*result
    return result

>>> factorial(1000)
HUGE NUMBER
```

String formatting

- String formatting is an elegant way of creating new strings. We avoid building complex strings with concatenation.
- `format % value(s)`
- `format` is a string with holes in it and `value(s)` are hole fillers.

```
>>> s = 'this %s string %s' % ('is','formatting')
>>> s
'this is string formatting'
```

- More about this in the next lecture.

GÖTEBORGS UNIVERSITET

Språkbanken

CLT

List comprehension

- List comprehension is a convenient way of defining new lists.
- We use looping and conditions to define the values of the list.

```
>>> range(1,10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> [n*2 for n in range(1,10)]
[2, 4, 6, 8, 10, 12, 14, 16, 18]

>>> [n*2 for n in range(1,10) if n % 3 == 0]
[6, 12, 18]

>>> [(x,y) for x in range(1,5) for y in range(1,5)]
[(1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2),
 (2, 3), (2, 4), (3, 1), (3, 2), (3, 3), (3, 4),
 (4, 1), (4, 2), (4, 3), (4, 4)]
```

Module with test code

```
# test_code.py

def print_hello():
    print 'hello'

if __name__ == '__main__':
    print_hello()
```

```
>>> import test_code
>>> test_code.print_hello()
hello
```

```
$ python test_code.py
hello
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Rest of the lecture

- Presentation of Assignment 1: Princeton WordNet.
- Walk-through of NLTK Chapter 2.5.