



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

## Lecture 9: GUI and debugging

*Introduction for Linguists (LT2102)*

Markus Forsberg  
Språkbanken  
University of Gothenburg

October 12, 2010



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Assignment 2

- ▶ I have now corrected all assignments.
- ▶ If you have not received a response, please resubmit.
- ▶ If you don't understand my comments, please ask.
- ▶ Some comments: `word_tokenize` requires a sentence; just read and process the data once; avoid using `print` in functions (unless you are debugging).



GÖTEBORGS  
UNIVERSITET

Språk  
BANKEN

CLT

# Assignment 3: Language Identification

- ▶ In assignment 3 you will work on the problem of *language identification*: given a text in a unknown language, find the most probable language.
- ▶ We solve this problem by creating *language models*, or *profiles*, for a set of languages, which the unknown texts are matched against.
- ▶ A profile is simply a frequency table of bits of text, called *N-grams* (the N refers to the length of the bit).
- ▶ For example, *ing* is a frequent suffix in English, so if it is also frequent in the text, then we are leaning toward English.



# The exam

GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

- ▶ Many have started to think about the exam.
- ▶ It will be problem solving with paper and pen.
- ▶ The exam will not be like an assignment (you have two weeks to do the assignments).
- ▶ A question is typically: *define a function  $f$  that (SOMETHING NOT THAT COMPLICATED).*
- ▶ There will be an reexam this semester. The date is not yet set.



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Debugging

- ▶ Debugging: to find the bugs of a program.
- ▶ Syntactic errors: formal errors
- ▶ Runtime errors: errors occurring while running the program
- ▶ Semantic errors: the program does not have the intended meaning



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Common syntactic errors

- ▶ Common syntactic errors:
  - ▶ keyword as variable name;
  - ▶ missing colon after compound statement (e.g., for and def);
  - ▶ string missing matching quotation marks;
  - ▶ unclosed parenthesis - (, {, (;
  - ▶ = instead of == in a condition;
  - ▶ indentation problems (especially while mixing tabs and whitespaces).



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Runtime error

- ▶ Runtime errors are exceptions in Python.
- ▶ Some examples are:

**NameError:** a name in use does not exist in the current environment.

**TypeError:** we are trying to do something that is not defined for the current type (e.g., string assignment).

**ValueError:** the type is correct, but the value is unexpected (e.g., converting a non-number string to an integer).

**AttributeError:** same as NameError, but when using the dot notation.

**IndexError:** trying to index outside a sequence.



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# A common mistake

- ▶ Do not do too much in a try block.
- ▶ Specify the exception you are catching.

NO:

```
try:  
    BIG BLOCK OF CODE  
except:  
    pass
```

YES:

```
try:  
    SMALL BLOCK OF CODE  
except ValueError:  
    DO SOMETHING SENSIBLE
```



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Semantic errors

- ▶ The most difficult errors are *semantic errors*.
- ▶ The program is working, it just solves the wrong problem.
- ▶ You need to analyze the connection between the erroneous behavior of the program and the source code.



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# How do we know that a program is correct?

- ▶ We can try to prove it. Requires a strong mathematical background and a mountain of patience, since there are so many details to take care of.
- ▶ Moreover, proving is not the complete answer, since there could be a bug in the interpreter/compiler...
- ▶ Testing: a poor man's proof.
- ▶ *Unit testing*: functions or programs are viewed as black boxes. We specify the input and the expected output.
- ▶ *Regression testing*: making sure that things that worked before, after some modification, still works.



# Testing

GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

## CLT

- ▶ Python has good support for testing.
- ▶ module doctest: tests are added to the documentation strings.
- ▶ module unittest: a more powerful framework for testing.



# module doctest

GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

- ▶ Documentation and testing in one.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import doctest

def average(values):
    """Computes the arithmetic mean of a list of numbers.

    >>> average([20,30,40])
    30.0

    >>> average([10,10,10])
    42.0
    """
    return sum(values,0.0)/len(values)

if __name__=='__main__':
    doctest.testmod()
```



# Unit testing

GÖTEBORGS  
UNIVERSITET

Språk  
BANKEN

CLT

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import unittest

def average(values):
    return sum(values, 0.0)/len(values)

class Test(unittest.TestCase):
    def test_average1(self):
        self.assertEqual(average([20, 30, 40]), 30.0)

    def test_average2(self):
        self.assertEqual(average([10, 10, 10]), 15.0)

if __name__ == '__main__':
    unittest.main()
```



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Performance logging

module 'timeit' is used to time a function call.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import timeit

if __name__=='__main__':
    t = timeit.Timer("primes.primes(1000)",
                    "import primes")
    print 'primes: %.2f seconds' % t.timeit(number=1000)

    t = timeit.Timer("primes.primes_improved(1000)",
                    "import primes")
    print 'p. improved: %.2f seconds' % t.timeit(number=1000)

$ python time_test.py
primes: 24.75 seconds
p. improved: 2.51 seconds
```



GÖTEBORGS  
UNIVERSITET

Språk  
BANKEN

CLT

# GUI

- ▶ *GUI*: Graphical User Interface
- ▶ There are many GUI libraries for Python (wxPython, Tkinter, Qt).
- ▶ We will have a look at Downey's module *Gui.py*, which is a simplified interface of Tkinter.
- ▶ A GUI consists of a collection of widgets, such as buttons and text fields.
- ▶ *Event-driven program flow*: the program acts on what the user does.
- ▶ When the user interacts with a widget, then the widget's *callback* function is called.



# An GUI example

GÖTEBORGS  
UNIVERSITET

Språk  
BANKEN

CLT

```
import Gui
import random

color = 'white'

def callback(canvas):
    global color
    colors = ['white', 'black', 'red', 'green']
    colors.remove(color)
    color = random.choice(colors)
    canvas.config(bg=color)

def change_color_demo():
    g = Gui.Gui()
    g.title('Color demo')
    canvas = g.ca(width=250,height=250,bg=color)
    button = g.bu(text='Change color',
                  command=Gui.Callable(callback, canvas))
    g.mainloop()

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



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Problem: Implementing a GUI

- ▶ Define a GUI with a circle controlled by four buttons: up, down, left, and right.



GÖTEBORGS  
UNIVERSITET

Språk  
BANKEN

CLT

# Solution: GUI

```
import Gui

circle = None

def up():
    circle.move(0,-50)

def down():
    circle.move(0,50)

def left():
    circle.move(-50,0)

def right():
    circle.move(50,0)
```



# Solution: GUI (cont.)

GÖTEBORGS  
UNIVERSITET

Språk  
BANKEN

CLT

```
def circle_game():  
    g = Gui.Gui()  
    canvas = g.ca(width=500, height=500, bg='white')  
    global circle  
    circle = canvas.circle([0,0], 50, fill='red')  
    g.title('circle game')  
    g.bu(text='up', command=up)  
    g.gr(cols=2)  
    g.bu(text='left', command=left)  
    g.bu(text='right', command=right)  
    g.endgr()  
    g.bu(text='down', command=down)  
    g.mainloop()
```



GÖTEBORGS  
UNIVERSITET

**Språk**  
BANKEN

CLT

# Problem: Testing a program

- ▶ The function below sorts a list (copied from <http://www.daniweb.com/code/snippet216689.html>).
- ▶ Or does it? Write some unit tests to ensure that it actually sorts a list.

```
def bubble_sort(list2):
    for i in range(0, len(list2) - 1):
        swap_test = False
        for j in range(0, len(list2) - i - 1):
            if list2[j] > list2[j + 1]:
                list2[j], list2[j + 1] = (list2[j + 1],
                                           list2[j])

                swap_test = True
        if swap_test == False:
            break
    return list2
```



# Solution: Testing

GÖTEBORGS  
UNIVERSITET

Språk  
BANKEN

CLT

```
import unittest
import random

def bubble_sort(list2):
    ... # removed to fit page

class Test(unittest.TestCase):
    def test_bubble_sort_1(self):
        self.assertEqual(bubble_sort([]), [])

    def test_bubble_sort_2(self):
        self.assertEqual(bubble_sort([1]), [1,2])

    def test_bubble_sort_3(self):
        self.assertEqual(bubble_sort(['b','a','c','d']),
                        ['a','b','c','d'])

    def test_bubble_sort_4(self):
        lst = [random.randint(-200,200) for _ in range(10)]
        result = sorted(lst, reverse=True)
        self.assertEqual(bubble_sort(lst), result)

if __name__ == '__main__':
    unittest.main()
```