GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Lecture 7: Files, databases, and pickling

Introduction for Linguists (LT2102)

Markus Forsberg
Språkbanken
University of Gothenburg

September 29, 2010

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Assignment 1

- I have now corrected all submissions. If you are missing your response, please resubmit.
- If you got return, do not be concerned, it is part of the learning process. Resubmit as soon as possible.
- If you don't understand why you got a return and/or my comments don't make sense to you, ask me as soon as possible.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Printing and returning

- There is some confusion about printing and returning in a function.

```
# ex.py
def procedure():
    print 'hello'

def function():
    return 'hello'
```

```
>>> import ex
>>> ex.procedure()
hello # not a string.
>>> ex.function()
'hello' # interpreter echos values.
>>> print ex.procedure()
hello
None # no return value
>>> print ex.function()
hello
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Some random comments

- Make sure that you know what value your name refers to, and use an appropriate name (endless source of confusion, otherwise).
- If you have a problem with a function, try explaining it, in detail, to someone. This usually clears things up.
- Reminder: work with the course literature in front of a computer. Try all examples, and experiment with them.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Solving a problem

- Solving a computational problem consists of three different tasks:
  - understanding how to solve the problem (in detail, since computers are stupid);
  - implement it in a programming language (the focus of this course);
  - convincing yourselves that the program actually solves the problem.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Assignment 2

- In assignment 2 we will work with a readability measure.
- You will only be presented with the top level function. It is your job to define the functions that you need to solve the problem.
- Hint: keep the functions short, do not try to do too much in a function.
- The assignment has two optional tasks. Solve the non-optional part first, then, if you have the time, go for the optional tasks.
- Deadline: 8 October, 23.59 CET

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Stand-alone program

- How to create a stand-alone program on a Unix-like system (still requires that Python is installed on the system where it is run).

```
#!/usr/bin/env python

import sys

if __name__ == '__main__':
    try:
        filename = sys.argv[1]
        with open(filename) as f:
            file_content = f.read()
            print file_content
    except IndexError:
        print ' usage: %s <filename>' % (sys.argv[0])
    except IOError:
        print " error: no file '%s'." % filename
```

```
$ chmod a+x standalone.py
$ standalone.py file.txt
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Default arguments in functions

```
>>> ts = 'a note on default arguments'.split()

>>> sorted(ts)
['a', 'arguments', 'default', 'note', 'on']

>>> sorted(ts,reverse=True) # reverse is a default argument.
['on', 'note', 'default', 'arguments', 'a']

def center(s,length,c=' '):
    if len(s) < length:
        n = (length-len(s))/2
        return n*c + s + n*c
    else:
        return s

>>> center('middle',20)
'       middle       '

>>> center('middle',20,'*')
'*******middle*******'
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Data persistence

- **Data persistence**: the data persists even though the Python interpreter has been terminated.
- We will now look at three different ways of achieving data persistence:
  - files
  - module anydbm
  - pickling

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Files

- Files are persistent data on a computer.
- Files modes: `r` (read), `w` (write), and `a` (append).

```
def create_notepad(name):
    with open(name,mode='w') as f:
        f.write('')

def add_note(name,note):
    with open(name,mode='a') as f:
        f.write(note+'\n')

def read_notes(name):
    with open(name,mode='r') as f:
        return f.read()

>>> filename = 'notepad.txt'
>>> create_notepad(filename)
>>> add_note(filename,'go shopping')
>>> add_note(filename,'walk the dog')
>>> print read_notes(filename)
go shopping
walk the dog
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

anydbm

- A **database** is a persistent, structured collection of data.
- The module `anydbm` allows us to create databases with a similar functionality to a dictionary. However, it only works on strings.

```
>>> import anydbm
>>> phonebook = anydbm.open('phonebook.db','c')
>>> phonebook['Alice'] = '555-1234'
>>> phonebook['Bob'] = '555-4321'
>>> phonebook.close()
[close down Python]

$ python
>>> import anydbm
>>> phonebook = anydbm.open('phonebook.db')
>>> phonebook['Bob']
'555-4321'
>>> phonebook.close()
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Pickling

- **Pickling**: translating almost any Python object to a string, and back again. Gives us the possiblity of persistent storage of Python objects.

```
>>> import pickle
>>> sentence = ['the', 'sun','is','shining']
>>> pickle.dumps(sentence)
"(lp0\nS'the'\np1\naS'sun'\np2\naS'is'\np3\
 aS'shining'\np4\na."
>>> data = pickle.dumps(sentence)
>>> sent = pickle.loads(data)
['the', 'sun', 'is', 'shining']
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Downloading content of URL:s

```
import urllib

def get_content_of_url(url):
    connection = urllib.urlopen(url)
    content = connection.fp.read()
    connection.close()
    return content

>>> get_content_of_url('http://thinkpython.com/secret.html')
'<title>Secret exercise</title>\n\n ... # HTML file
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Problem: Game show

- You are in a game show. The host presents you with three boxes, and tells you that one of the boxes contains one million dollars.
- You select a box.
- The game host opens all other boxes, except one, and he never opens the box with the million.
- There are now two boxes, the one you have selected, and another box. The host tells you that you may change box. Should you?
- The show ends before you get to choose, and you are asked to return next week.
- Write a Python program to simulate the situation to help you make the correct choice.

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Solution: Game show

```
import random

def create_boxes(num_of_boxes):
    return range(1,num_of_boxes+1)

def select_a_box(boxes):
    return random.choice(boxes)

def open_boxes(million_box, boxes):
    if million_box in boxes:
        return million_box
    else:
        return random.choice(boxes)

def play_once(num_of_boxes, change_box):
    boxes = create_boxes(num_of_boxes)
    million = select_a_box(boxes)
    player_choice = select_a_box(boxes)
    boxes.remove(player_choice)
    new_choice = open_boxes(million, boxes)
    if change_box:
        return new_choice == million
    else:
        return player_choice == million
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Solution: Game show (cont.)

```
def simulation(num_of_boxes, change_box, num_of_runs):
    winnings = 0
    for _ in range(num_of_runs):
        if play_once(num_of_boxes, change_box):
            winnings += 1
    return 100*float(winnings)/num_of_runs

same_box_result = simulation(3,False,1000)
print 'same box: %.1f %s' % (same_box_result, '%')
change_box_result = simulation(3,True, 1000)
print 'change box: %.1f %s' % (change_box_result, '%')
```

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Problem: Guess a number

- Implement the game ‘guess a number’. The game ask the player to guess a number between 1-1000, and the player’s goal is to guess the correct number with as few guesses as possible. If the player’s guess is wrong, then the game guides the player by responding "greater" or "smaller".

GÖTEBORGS UNIVERSITET

Språk-BANKEN

CLT

Solution: Guess a number

```
import random

def get_a_number(n):
    while True:
        s = raw_input('Guess a number (1-%i): ' % n)
        try:
            return int(s)
        except ValueError:
            print 'Not a number...'

def guess_a_number(n):
    number = random.randint(1,n)
    print number
    num_of_guesses = 0
    while True:
        guess = get_a_number(n)
        num_of_guesses += 1
        if guess == number:
            print 'Correct! (%i guesses)' % num_of_guesses
            break
        elif guess < number:
            print 'greater...'
        else:
            print 'smaller...'
```