Functions

Objectives

To write and call our own user-written functions, and to continue practising Python.

Reference Material

Chapter 08 Functions. You might also wish to look at `os.times` in the online Python documentation.

Questions

1. Create a function which takes two arguments: a value and a list. The list should have a default of an empty list, for example:

   ```
   def myfunc(value, alist=[]):
   ```

   The function should append the value to the list then print the contents of the supplied list.

   Call this function several times with various different values, defaulting the list each time. Can you explain the output?

   Can you suggest a solution to the problem?

2. You may recall from a previous exercise that we had a couple of functions to time an operation. In this exercise you get to write those functions.

   The basis of the timing is to use **os.times()**. This returns a tuple describing the processor time used since the script started, and other items. We are only interested in the first two items in this tuple, the system time and user time. The total elapsed CPU time is the sum of these two items.

   Since the time given by **os.times()** is from some arbitrary moment, to start the timer we need to take the CPU time used so far and store it in a global variable. Do that in a function called **start_timer()**.

   In a function called **end_timer()** get the CPU time used, and subtract the time stored in the global variable to get the CPU time used between the two function calls. Display that CPU time, rounded to 3 decimal places, and a text message passed by the user as a parameter. The text should have a default value of "End time", and be a minimum width of 12

characters.

Test your functions by having a long operation between the two calls. We suggest counting the number of lines in the 'words' file. For example:

```
start_timer()
lines = 0
for row in open("words"):
    lines += 1

end_timer()
print "Number of lines:",lines
```

This code is already supplied in **Ex8.2.py**

3. Ah! We almost forgot. You did document your functions in the previous exercise, didn't you? Add docstrings to your functions, if you didn't already do it.

   To test: start IDLE and load your script (File/Open) then run it (<F5>). Then in the "Python Shell" windows type:
   ```
   >>> help(start_timer)
   >>> help(end_timer)
   ```

4. By now you should have seen the `country.txt` file in another exercise. It consists of lines of comma-separated values. If we wish to input these to a SQLite database using SQL, then these values (also called fields) must be slightly modified:
   a. Trailing white-space (including new-lines) must be removed.

   b. Any embedded single quote characters must be doubled. For example, `Cote d'Ivorie` becomes `Cote d''Ivorie`.

   c. All values must be enclosed in single quotes. For example, `Belgium,10445852,Brussels,737966,Europe` becomes:
   `'Belgium','10445852','Brussels','737966','Europe'`

   Write a Python script with a function to change a line into the correct format for insertion into an SQL statement, using the guidelines above. Call the function for each line in country.txt, and display the reformatted line.

   Hints:
   ```
   a.  rstrip()
   b.  re.sub()
   c.  lrow = []
       for field in row.split(','):
           lrow.append("'" + field + "'")
       Then use join()
   ```

If time allows:

   If you used the suggested `'for'` loop in the Hint, rewrite the code to use `map()` with a `lambda` function instead.

   .

Solutions

Question 1

Our function:

```
def myfunc1(val, lista = []):
    lista.append(val)
    print "value of lista is:",lista

myfunc1(42)
myfunc1(37)
myfunc1(99)
```

Output is:

```
value of lista is: [42]
value of lista is: [42, 37]
value of lista is: [42, 37, 99]
```

The problem is that the empty function is declared at the time of the function definition, which is at run-time. The default parameter is a reference to the empty list declared at that time. Subsequent default calls do not create a new list, they use the same one each time.

The normal Python idiom to solve this is to default to None instead:

```
def myfunc2(val, lista = None):
    if lista == None: lista = []
    lista.append(val)
    print "value of lista is:",lista

myfunc2(42)
myfunc2(37)
myfunc2(99)
```

Output is:

```
value of lista is: [42]
value of lista is: [37]
value of lista is: [99]
```

Questions 2 & 3

```
import os

start_time = 0;


#######################################################
# TIMER FUNCTIONS
def start_timer():
    """
    The start_timer() function marks the start of
    a timed interval, to be completed by end_timer().
    This function requires no parameters.
    """
```

```
        global start_time
        (utime,stime) = os.times()[0:2]
        start_time = utime+stime


    def end_timer(txt='End time'):
        """
        The end_timer() function completes a timed interval
        started by start_timer.  It prints an optional text
        message (default 'End time') followed by the CPU time
        used in seconds.
        This function has one optional parameter, the text to
        be displayed.
        """
        (utime,stime) = os.times()[0:2]
        end_time = utime+stime
        print "%-12s: %05.3f seconds" % \
                (txt,end_time-start_time)


    start_timer()
    lines = 0
    for row in open ("words"):
        lines += 1

    end_timer()
    print "Number of lines:",lines
```

Question 4

```
import re


def prep_row (row):
    row = row.rstrip()
    row = re.sub(r"'",r"''",row)

    # Add quotes around each field
    lrow = []
    for field in row.split(','):
        lrow.append("'" + field + "'")


    return ",".join(lrow)



for row in open ("country.txt"):
    print prep_row(row)
```

If time allows:

Lambda version:

```
lrow = list((map (lambda f: "'"+f+"'", row.split(','))))
```