Exercise 8 – Functions

Objective

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

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 my_func(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 values, defaulting the list each time. Can you explain the output?

   Can you suggest a solution to the problem?

2. 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(my_func)
   ```

3. 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 program 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 my_func1(val, lista=[]):
    lista.append(val)
    print("value of lista is:", lista)
    return

my_func1(42)
my_func1(37)
my_func1(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 list 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 my_func2(val, lista=None):
    if lista == None:
        lista = []
    lista.append(val)
    print("value of lista is:", lista)
    return

my_func2(42)
my_func2(37)
my_func2(99)
```

Output is:

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

Question 3

```
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", "r"):
    print(prep_row(row))
```

If time allows:

Lambda version:

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