Lab 15: Case Study: Basic Bioinformatics in Python

ITE315: Scripting Languages and System Administration

Contents

1 Overview

There is no native support with Python for building applications that use a graphical user interface rather than a text-based interface. In this lab, you will use the Tk toolkit to build a simple GUI-based application.

This application will use the SQLite database manager to read a set of DNA strings from a database table and display the items read in a selectable list box. Selection of an item will result in that item being echo'ed back to you in a dialog box.

2 Background

Our lecture on how to extract data from a database using Python was based on the MySQL database. That's a very heavy database that isn't suited to every problem. So for this lab we will be using the SQLite database. SQLite

is to Linux like Microsoft Access is to Windows: it's a lightweight file-based database manager.

The power of Python's database module is that it works the same way regardless of which database manager you use. One works by opening a connection to a database, using the `execute()` function to execute SQL statements, and then processing the resulting list of rows returned from a query.

3 Instructions

You will need to make certain that you have the correct software installed. We're assuming in this lab that you're doing the lab using Linux. SQLite is available for other operating system... speak with your Glorious Instructor if you want to do this lab in Windows or macOS.

3.1 Setup

You need to make certain that you have SQLite v.3 installed on your lab machine. Check to see if it is available by running the `sqlite3` command in the Bash shell.

If sqlite3 is not available, you should install it using your distribution's preferred package manager. For example, the install command for Ubuntu would be:

```
sudo apt-get install sqlite3
```

Download and unzip the contents of the data zipfile attached to the assignment on Blackboard into your working folder for this lab. You will see a file named `seqdata.db` in your folder.

3.2 Testing the DB connection by dumping the Data

Let's write a small Python script that will open the database, read the table containing the DNA strings and dump them to standard output:

```
#!/usr/bin/python
import sqlite3
dbname = 'seqdata.db'
try:
    conn = sqlite3.connect(dbname)
except:
```

```
    print "Unable to open database"

query = 'SELECT * FROM Sequence'
try:
    rowSet = conn.execute(query)
    for row in rowSet:
        seqNum = row[0]
        sequence = row[1]
        print "Sequence %d : %s" % (seqNum, sequence)
except:
    print "Error: Unable to access the database"

conn.close()
```

Some things to note about this code:

- See how we load the `sqlite3` module rather than the `mysql` module used in the lectures,
- But the sequence of things you do to access the database remains the same: (1) connect, (2) execute, and (3) process.
- Executing SQL statements returns an instance of a Python list data structure. Processing the rows in the returned data is just a matter of iterating over the list using a `for`-loop
- And note how each column in the SQL table maps to the indices of each row

Try this code, making certain that you run the script in the folder where you placed the SQLite database file during the Setup phase of the lab.

4 Combining the database with a GUI

Let's start a new Python script that builds the GUI application we want to build. We'll do this using Python's `tkinter` module that works with the Linux TCL framework.

Add the following to start your new script:

```
from Tkinter import *
import tkMessageBox

import sqlite3
```

This imports the modules we are going to need to build the GUI and connect to the database.

The next step is define and build the window of our new application. We do that by constructing a Tcl/Tk `Frame` widget:

```
top = Tk()
frame = Frame(top)
frame.pack()
label = Label(frame,text="Sequence Mutator", relief=RAISED)
label.pack(side=TOP)
```

The creates the window frame, packs it into the UI, and adds and packs a label that says what this window is doing.

The next step is add a Tcl/Tk `Listbox` to the `Frame` and fill it with data from the database:

```
seqListBox = Listbox(frame)
dbname = 'seqdata.db'
try:
    conn = sqlite3.connect(dbname)
except:
    print "Unable to open database"
    tkMessageBox.showinfo("Database error","Unable to open
        database")

query = 'SELECT * FROM Sequence'

rowSet = conn.execute(query)
for row in rowSet:
     seqNum = row[0]
     sequence = row[1]
     lbcontent =  "Sequence %d : %s" % (seqNum, sequence)
     seqListBox.insert(seqNum, sequence)
     print "Sequence %d: %s" % (seqNum,sequence)

seqListBox.pack(side = LEFT)
# We're through with the database
conn.close()
```

This is effectively the same as we did in our first program. The difference is that instead of printing the sequence, we the `ListBox` method for inserting the data as entries in the `ListBox`.

Now we need to do some setup to get input doing what we want it to do. We'll do this by calling the methods for listboxes that initializes the listbox

and sets the cursor on the first item in the list:

```
seqListBox.select_clear(0,"end")
seqListBox.selection_set(0)
seqListBox.see(0)
seqListBox.activate(0)
seqListBox.selection_anchor(0)
```

And finally, we will add a Tcl/Tk `Button` widget that will display the selected text in a Tcl/Tk `MessageBox`:

```

mutateButton = Button(frame, text="Mutate", fg="black",command=
    mutate)
mutateButton.pack( side = RIGHT)
# And hand control to the Tk event loop
top.mainloop()
```

One more step: we need to implement the callback command function `mutate` that goes at the top of your file after you defined `seqListBox`:

```
def mutate():
    selectedSeq = map(int,seqListBox.curselection())
    if not selectedSeq:
        sequenceString = ''
    else:
        sequenceString = seqListBox.get(selectedSeq[0])
    print "String to mutate is: "+sequenceString
    tkMessageBox.showinfo("Selected Sequence", sequenceString)
```

For this lab, the function grabs the data from the list box and displays it. In your immediate future, you will get to implement the actual mutation of the string.

5 Submission instructions

Combine the source code of both programs and screenshots of your program executing into a single PDF file and attach this PDF file to your submission in Blackboard.