UNIVERSITY OF TORONTO
FACULTY OF INFORMATION

Faculty of Information
School of Graduate Studies
University of Toronto – St. George
Semester: Fall 2018
INF1340H – Programming for Information Systems

preliminary, 2018-11-24

Assignment 3 -- 18%
due: 2018 December 07, before midnight

Write a Python program that delivers the requested results. Include appropriate documentation: comments and docstrings.

1. assignment overview

Assume that the user is a bioinformatician who wants to query a genetic database and retrieve some data from it.

a) Ask the user to specify the query he/she wants to execute (i.e., which data should be retrieved).

b) Connect to the database and retrieve the requested data.

c) Print a summary of the retrieved data. The retrieved data might be big and/or complicated. Tell the user the size of the data, and display the first ten rows that were retrieved.

2. Asking for the query

Use Tkinter features to prompt the user to enter an SQL query, and then make this query string available to the calling program.

`show tables;` should be offered as a default query string and should be presented to the user as a time-saving option. If the user wants to run a different query, he/she can type it in.

You may use a radio button approach, or a simple text entry approach.

Global variables are permissible in the GUI section of your program. Not permissible elsewhere. You should be able to complete this task without using `lambda`, but if you want to use it, please feel free to do so.

steps:
- Create a basic Tk object (a window with nothing in it).

- Add a label, some sort of text entry box, and an OK button to the Tk object.
Optionally, add a radio button feature.
- Wait for the user to click on the OK button.
- Get the query string (new or default).
- Close (destroy) the Tk object.

3. Submitting and running the query

Connect to the Ensembl database and submit the query to it. The server hostname is `ensembldb.ensemble.org`, and the database name is `gorilla_gorilla_core_52_1`.
Save the data (rows) that are retrieved as a result. "Save" here means to keep a copy of the data in your Python program until your program no longer needs it. It is not required that you store the data in a file.

The query might not run successfully. For instance, your program might not be able to connect to the database server. Failures (inability to return the requested rows) should be reported to the user. Your program should use Python exception handling to ensure graceful termination.

steps:
- Prepare for possible failure of the operation.
- Create a connection with the database server and its gorilla database.
- Create a cursor to use with this connection.
- Submit the query.
- Collect the results of the query. Save for the summary task.
- Close the cursor, and sever the connection.

4. Analyzing summarizing the results

Tell the user how many rows and columns were retrieved. Tell her/him the column names. Characterize the columns (e.g., what the types are: numeric, text, boolean, etc.) Display the first ten rows of the retrieved data set. (Display all rows if fewer than ten.)

5. Some requirements

- Submit your work in a single file. The file name should begin with "a3" and should include the partners' names, e.g., a3-george-and-gracie.py

- Identify the starting point by using `if __name__ == "__main__":`

- Implement the program's basic tasks by calling `get_query()`, `retrieve_data()`, and `summarize_data()` from your main program.

- Use Tkinter for your graphical user interface.

- Use a MySQL connector, and if you cannot use the regular connector, please consult with the instructor.

- Use Python exception handling (`try:` and `except:`) as necessary to ensure that a MySQL connection/retrieval problem won't cause your program to terminate early with Python run-time messages.

- If you create any temporary files, please delete them from within your program. Do not ask the user to clean up for you.

- Please check the a3 notes file periodically for clarifications.

6. Optional nice features that you could include -- but not required

a) Tell the user that he/she might have to wait a while for the query to run.
b) Tell the user how long the retrieval took.
c) Save the retrieved data in a file and/or in the clipboard.
d) Use a hanging indent when presenting the first ten rows.
e) Prepare a graphical profile (image file) of the retrieved data.
f) Allow the user to query a database other than gorilla_gorilla_core_52_1.

7. tasks along the critical path

Please make sure (as soon as possible) that you are able to
- connect to and query the Ensembl database
-- install (or otherwise gain access to) an appropriate MySQL connector.

Your study and preparation for this assignment should include
- GUI basics
-- global variables
- database basics
- exception handling

8. MySQL Queries:

You do not need to concern yourself with query validity. Submit whatever the user tells you to.

Sample queries:

```
show tables;
select * from coord_system;
select * from gene;
select * from gene where analysis_id = 34;
select * from gene where biotype='snRNA';
```

9. to facilitate marking

- Use the Quercus assignment dropbox, and if you worked as a team, submit only one copy per pair.

- Provide your name and your partner's name in a comment at the top of the program. Put your name on line 4, and your partner's name on line 5 (lines numbering from 1). Also, provide your section number (L101 for Tuesday, L102 for Thursday).

- Give your program a title (also in a comment at the top), and indicate your revision number and the date of the revision. Your program should be compatible with Python 3.5, and it is suggested that you indicate version compatibility in a comment, too -- i.e., which version(s) of Python will your program run on?

- The names of global variables should begin with '`g_`'. Python code within exception-handling blocks (`try:, except:, finally:`) should be tagged with '`##`' to the right of each line.

- Try to avoid confusing the markers. Creativity is encouraged, but please keep in mind that graders are expecting the program to behave as described earlier in this document.
If you add extra features to your program, please ensure that they do not prevent smooth, simple operation. Any extra features should be "opt-in", not "opt-out". I.e., the marker should not need to answer any extra questions or click on anything extra if he does not want to use the features.

- If you know, at submission time, that your program does not run successfully, please make a note of the known deficiencies in a comment at the top.

- Describe appropriate test cases in your docstrings.

- Use prescribed names for constants, variables, and functions. Please draw upon this list as needed:

```
HOST
USER
PASSWORD
DATABASE
DEFAULT_QUERY_STRING
```

```
get_query()
retrieve_data()
```

```
summarize_data()

user_query_string
```

- If your program is not easy to read, add comments, whitespace, etc., as appropriate. If your functions are long, rewrite them so that they are shorter, or split the code out into additional functions. (Note that if you use programming tricks to shorten the code, you should add comments to explain the tricks.)

example of first lines in your .py file:

```
#
# inf1340, section L101
# assignment 3 - due 2018-12-07
# Burns, George                     << Names on lines 4 and 5
# Allen, Gracie                     << Blank line if no partner
#
# MySQL client program
# v1.0.3 - 2018-12-04
# compatible with Python versions 3.4 - 3.7
# source file:  a3-george-and-gracie.py
#
```

scoring: (out of 100 possible)

headers, docstrings, and bodies of functions:	
to obtain the query string from the user	25 marks
to connect to the database server and retrieve data	20 marks
exception handling for the connection and data retrieval	5 marks
to summarize the retrieved data	20 marks
other code, including __main__	5 marks
coding style and comments	10 marks
Program produces correct output.	10 marks
facilitation of marking (as described earlier)	5 marks

Don't be late! There is a 15% per day (or part of day) penalty for late submission.