Week 4 - Python Exercises

IDBS - Autumn 2023 - Omar Shahbaz Khan

This exercise will practice the following:

1. Creating and Dropping tables (create_peron_table)
2. Inserting, Selecting, Updating (insert_person, print_people, transfer_money)
3. Implementing business logic (get_balance_unsafe/safe, set_balance, transfer_money)
4. Using transactions (transfer_money_with_transactions)

Python

These exercises requires Python, so if you haven't already installed it please do so.

There are many ways to install Python, some operating systems have it pre-installed.

Test by running the command `python --version` in a terminal/command prompt.

If you don't have a python installation you can get it from: https://www.python.org

Another option is Anaconda: https://www.anaconda.com

For Windows users you can also get it from the Microsoft Store

Choose whichever you prefer

After installing python or if you already have it, install psycopg

```
python -m pip install psycopg
```

in mac: pip3 install psycopg //once

Running the exercise

1. Open terminal/command prompt in the folder the python script is
2. Run `python 04-exercises.py <task_number>`
3. Example for running task 1: `python 04-exercises.py 1` in mac: python3 script.py 1

Create Database

Start by creating an empty database 'pydb'.

```
createdb -U postgres pydb
```

默认用户：createdb pydb
指定用户：-U postgres

This assumes you are using the user postgres, change as you see fit.

In the database.ini file specify the user and password.

Task 1 - Creating a Table

Create a table in the database with the following:

- Drop the table if it exists.
- Create a table `person` with attributes: `name VARCHAR(100)` and `balance INT`

Add your code to the function `create_person_table(conn)`

Task 2 - Inserting a person

Insert a new person with the given parameters

Add your code to the function `insert_person(conn, name, balance)`

Task 3 - Print all the rows in person

Print the name and balance of every person. The print format should be `name=<name>, balance=<balance>`

Add your code to the function `print_persons(conn)`

Task 4 - Get balance (Bad)

Implement the function `get_balance_unsafe(conn, name)`, in a way that is susceptible to SQL Injection. The function works as follows:

1. Select the balance where name is equal to the parameter name
2. Return the balance

The idea is to demonstrate that a caller could maliciously inject SQL into the parameter.

Instead of calling get_balance_unsafe("'Batman'"), the caller could write get_balance_unsafe("'; SOME SQL STATEMENT HERE; --"), and make the application perform actions that were not intended.

For example deleting or modifying data.

Task 5 - Get balance (Good)

Implement the function `get_balance_safe(conn, name)` in a secure manner by separating the query string and parameters in execute().

Task 6 - Set balance

Set the balance of a given account.

Add your code to the function `set_balance(conn, name, balance)`

Task 7 - Transfer Money

In this task transfer an amount from person 1 to person 2.

Feel free to add more logic to the transfer, such as a check to see if the balance of the account the money is transferred from becomes negative, and deny the transfer in such a case.

Add your code to `transfer_money(conn, acc1, acc2, amount)`

Task 8 - Transfer with Transaction

The implementation of the transaction logic is easy to get wrong, and might be confusing to write without an example. For this reason, the solution is provided. Instead, try to understand what each line does, why it is necessary, and discuss your thoughts with a classmate or a teaching assistant.