Programming in Python V: Accessing a Database

Computer Science 105
Boston University
David G. Sullivan, Ph.D.

Accessing a Database from Python

• Import the necessary Python module:
  ```python
  import sqlite3
  ```

• Connect to the database and create an object known as a database handle, assigning it to a variable named db.
  ```python
  db = sqlite3.connect('<name of database file>')
  ```

• To connect to the database that we used for PS 2 and 3:
  • put the database file (movie.db) in the same folder as the Python program that will access it
  • use the following command in the program:
    ```python
    db = sqlite3.connect('movie.db')
    ```
Performing a Query

• Given a database handle, we can perform a query by:
  • using the database handle to create a cursor object:
    \texttt{cursor = db.cursor()}
  • using the cursor to execute the query:
    \texttt{cursor.execute("<SELECT command>" )}

• Once the query has been executed, we use the cursor to access the results.

• To get one tuple from the result, we use the \texttt{fetchone} function:
  \texttt{row = cursor.fetchone()}

  • if there's nothing to get, this function returns \texttt{None}

Performing a Query (cont.)

• Example for our movie database:
  >>> cursor = db.cursor()
  >>> cursor.execute("SELECT name, year FROM Movie;")
  >>> cursor.fetchone()
  ('Avatar', 2009)
Performing a Query (cont.)

- Another example:
  ```python
  >>> cursor.execute('''SELECT name, rating
             FROM Movie
             WHERE year = 1976;''')
  >>> cursor.fetchone()
  ('Rocky', 'PG')
  >>> cursor.fetchone()
  ('Network', 'R')
  >>> cursor.fetchone()
  ('All the President’s Men', 'PG')
  ```

Using a for Loop to Display Query Results

- Python allows us to use a for loop to process the results of a query one tuple at a time.

- Syntax: ```python
  for tuple in cursor:
  # code to process tuple goes here
  ```

- Example:
  ```python
cursor = db.cursor()
cursor.execute('''SELECT name, rating
             FROM Movie
             WHERE year = 1976;''')

  for tuple in cursor:
  print(tuple[0], tuple[1])
  ```

- The above code would give the following output:
  ```
  Rocky PG
  Network R
  All the President's Men PG
  ```
Executing a Query Based on User Input

- How can we execute a query that's based on user inputs?
  - example interaction:
    
    ```
    year to search for? 1976
    Rocky PG
    Network R
    All the President's Men PG
    ```

  - In theory, we could construct the query using string concatenation:
    ```
    searchYear = input("year to search for? ")
    command = '''SELECT name, rating FROM Movie
    WHERE year = '''
    command = command + searchYear + ''''
    cursor.execute(command)
    for tuple in cursor:
        print(tuple[0], tuple[1])
    ```

SQL Injection Vulnerability

- Problem: constructing a query using string concatenation can lead to a serious security breaches.
  - known as a SQL injection.

- Example: let's say that in addition to the movie tables, there's a table called Secret containing sensitive data.

- The user of the program shown on the previous slide can see data from Secret by entering something like this:
  ```
  Year to search for? 1976; SELECT * FROM Secret
  ```

- The string concatenation will produce the following:
  ```
  SELECT name, rating FROM Movie WHERE year = 1976;
  SELECT * FROM Secret;
  ```

- The program will display the entire first two columns of Secret!
SQL Injection Vulnerability (cont.)

• Here's another problematic input!

  Year to search for? 1976; DROP TABLE Secret

Parameterized Queries

• To avoid SQL injections, we use a parameterized query.

  • example:

    ```python
    command = '''SELECT name, rating FROM Movie
                   WHERE year = ?;'''
    ```

    • ? is a placeholder

  • We execute the parameterized query as follows:

    ```python
    cursor.execute(command, [searchYear])
    ```

    command string with placeholders

    a list containing the values that should replace the placeholders (the parameters)

  • The `execute` function replaces the placeholders with the specified values – taking whatever steps are needed to treat each parameter as a single literal value.
Parameterized Queries (cont.)

- Here's a query that takes three parameters:

```sql
command = '''SELECT M.name, M.year
FROM Movie M, Person P, Director D
WHERE M.id = D.movie_id
AND P.id = D.director_id
AND P.name = ?
AND M.year BETWEEN ? AND ?;'''
```

- Here's an example of using it:

```python
dirName = input("director's name: ")
start = input("start of year range: ")
end = input("end of year range: ")

cursor.execute(command, [dirName, start, end])
for tuple in cursor:
    print(tuple[0], tuple[1])
```

The Full Program (getMoviesByDirector.py)

```python
import sqlite3

filename = input("name of database file: ")
db = sqlite3.connect(filename)
cursor = db.cursor()

dirName = input("director's name: ")
start = input("start of year range: ")
end = input("end of year range: ")

command = '''SELECT M.name, M.year
FROM Movie M, Person P, Director D
WHERE M.id = D.movie_id
AND P.id = D.director_id
AND P.name = ?
AND M.year BETWEEN ? AND ?;'''

cursor.execute(command, [dirName, start, end])
for tuple in cursor:
    print(tuple[0], tuple[1])
```
Handling Queries with No Results

- What if the user enters a director who isn't in the database, or a range of years with no movies for the director?
- We'd like our program to print a message when this happens.
- One way of doing this is to maintain a count of the number of tuples that the program processes:

```python
...,
cursor.execute(command, [dirName, start, end])
count = 0
for tuple in cursor:
    print(tuple[0], tuple[1])
count = count + 1
# print a message if there were no results
# what should go here?
```

Concluding a Database Session

- At the end of a program that accesses a database, you should always do the following:
  - commit the transaction in which the commands were executed:
    ```python
db.commit()
```
  - close the database handle:
    ```python
db.close()
```
A Front-End for Our Movie Database

- Let's write a Python program that serves as a front-end to our movie database.

- For now, it will do the following:
  - get the name of a person from the user
  - use a parameterized SELECT command to retrieve the appropriate record from the Person table
  - if the specified person is in the database, print his/her information in the following format:
    
    <name> was born on <dob> in <pob>.
  
  - otherwise, print an appropriate error message

- Extension: change the format of the dob from yyyy-mm-dd to mm/dd/yyyy
Converting the Date Format

• To change the format of the dob from yyyy-mm-dd to mm/dd/yyyy, what should we do?