Python/Database Interaction

Relational Databases

- Examples: SQLite, PostgreSQL, mySQL, MS SQL Server, Oracle, DB2
- Store data in tables with rows and columns (like spreadsheet or Pandas df)
- Structured Query Language (SQL) is used to retrieve and manipulate data

SQLite Example:

1. import SQLite3
2. Establish a connection to a new or existing database:

```
dbconn = sqlite3.connect('SQLite_Python.db')
```

   A database can be created in RAM instead of a physical file with:

```
dbconn = sqlite3.connect(':memory:')
```

3. Obtain a cursor object:

```
cursor = dbconn.cursor()
```

4. Execute a SQL statement with the execute method of the cursor object:

```
cursor.execute('SELECT * FROM users')      #or .executemany
```

5. For SELECT queries, fetch and process record set:

```
rows = cursor.fetchall() #or fetchone, fetchmany(nrow)
for row in rows:
   print('{0}, {1}, {2}, {3}'.format(row[0], row[1], row[2], row[3]))
```

6. Commit any changes (e.g., for CREATE TABLE, INSERT, UPDATE, DELETE statements) to the database:

```
dbconn.commit()
```

7. Very important: close the cursor and database connection:

```
cursor.close()
dbconn.close()
```

Best practice: Put all database operations in a try/except block. On error, rollback transactions to the last commit.

Extension Exercise:

Lookup your local extended forecast (e.g., at forecast.weather.gov). Create a database (your choice of type). Add a table called, "Weather," and define columns, "Id" (unique integer primary key), "Date (text)", "Day" (text), "High" (integer), and "Low" (integer). Insert records for each daily forecast into the table in the appropriate fields. Select the data in the Weather table and process its records to print the records and calculate the average high and low temperatures for the rows in the table. Optionally use Pandas for this purpose.