Reminders

• Lab 9 and Exercise 5 were posted.
  • Due this Friday and this Sunday respectively.

  • Pay attention to the provided hints! Do NOT write unnecessarily complex code.

• If you have any questions on course material:
  • Come to office hours, ask in class, ask the lab TAs, post on Piazza, etc.
  • Do not wait till some coursework is due.
Next Lecture Topic

• (Relational) Databases
What is a Database? What is it used for?

• A database is used to store large, complex collections of data. It makes searching, storing and updating large amounts of data easy.

• Why use a database?
  • to search data quickly and in various ways
  • to find relationships among data
  • to allow multiple users to access/modify data.
Relational Databases

- Data is organized into multiple tables.
- Columns represent variables/attributes.
- Rows are records/instances.
  - The number of rows changes as we add/delete data.

Table: Students

<table>
<thead>
<tr>
<th>ID</th>
<th>Last name</th>
<th>First name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1765228</td>
<td>Smith</td>
<td>John</td>
</tr>
<tr>
<td>8735016</td>
<td>Brown</td>
<td>Alice</td>
</tr>
<tr>
<td>9002654</td>
<td>MacLeod</td>
<td>Alex</td>
</tr>
</tbody>
</table>

Table: VC Residents

<table>
<thead>
<tr>
<th>ID</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>1765448</td>
<td>120</td>
</tr>
<tr>
<td>9002654</td>
<td>121</td>
</tr>
</tbody>
</table>
Databases

• **SQL:** language for querying and maintaining relational databases
  • “Structured Query Language”
  • Pronounced “ess cue ell” or “sequel”
  • Can use in almost any database!

• Database Management System (DBMS): Software to allow users to interact with the database
  • We will use SQLite
SQLite in Python

• SQLite actually written in C
• But there is a builtin Python module that creates an interface for us to use it:

```
import sqlite3
```

• This module contains functions for connecting to the database, and passing SQL commands and queries to the DBMS.
Basic `sqlite3` usage

Connect to the database:

```python
con = sqlite3.connect('name.db')
```

Get a cursor which is our link to the database:

```python
cur = con.cursor()
```

After we edit our database, commit the changes:

```python
cur.commit()  # Typo fixed on Nov. 29th.
con.commit()
```

Close the cursor when we are done:

```python
cur.close()
```

Close the connection when we are done:

```python
con.close()
```
Running SQL Commands

• We run SQL commands in python with:

```python
cur.execute('SQL statement here')
```

• By convention, we write SQL keywords in uppercase and variables names in lowercase
Example: Precipitation table

<table>
<thead>
<tr>
<th>City</th>
<th>Snow</th>
<th>Total</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>St. John’s</td>
<td>322.1</td>
<td>1482</td>
<td>217</td>
</tr>
<tr>
<td>Charlottetown</td>
<td>338.7</td>
<td>1201</td>
<td>177</td>
</tr>
<tr>
<td>Halifax</td>
<td>261.4</td>
<td>1474</td>
<td>170</td>
</tr>
<tr>
<td>Fredericton</td>
<td>294.5</td>
<td>1131</td>
<td>156</td>
</tr>
<tr>
<td>Quebec</td>
<td>337.0</td>
<td>1208</td>
<td>178</td>
</tr>
<tr>
<td>Montreal</td>
<td>214.2</td>
<td>940</td>
<td>162</td>
</tr>
<tr>
<td>Ottawa</td>
<td>221.5</td>
<td>911</td>
<td>159</td>
</tr>
<tr>
<td>Toronto</td>
<td>135.0</td>
<td>819</td>
<td>139</td>
</tr>
<tr>
<td>Winnipeg</td>
<td>114.8</td>
<td>504</td>
<td>119</td>
</tr>
<tr>
<td>Regina</td>
<td>107.4</td>
<td>364</td>
<td>109</td>
</tr>
<tr>
<td>Edmonton</td>
<td>129.6</td>
<td>461</td>
<td>123</td>
</tr>
<tr>
<td>Calgary</td>
<td>135.4</td>
<td>399</td>
<td>111</td>
</tr>
<tr>
<td>Vancouver</td>
<td>54.9</td>
<td>1167</td>
<td>164</td>
</tr>
<tr>
<td>Victoria</td>
<td>46.9</td>
<td>858</td>
<td>153</td>
</tr>
<tr>
<td>Whitehorse</td>
<td>145.2</td>
<td>269</td>
<td>122</td>
</tr>
<tr>
<td>Yellowknife</td>
<td>143.9</td>
<td>267</td>
<td>118</td>
</tr>
</tbody>
</table>
cur.execute('CREATE TABLE Precipitation(City TEXT, Snow REAL, Total INTEGER, Days INTEGER)')

- Name of the table: **Precipitation**
- Names of columns: **City, Snow, Total & Days**
- Type of data in the columns:
  - **TEXT, REAL, INTEGER, INTEGER**
- Corresponding python types:
  - **str float int int**
SQL: Deleting a table

• Delete or *drop* a table:

```python
cur.execute(‘DROP TABLE Precipitation’)```
SQL: Inserting a row into a table

cur.execute('INSERT INTO Precipitation VALUES ("St. John\'s", 322.1, 1482, 217)')
or
cur.execute('INSERT INTO Precipitation VALUES(?,?,?,?)', ("St. John\'s", 322.1, 1482, 217))
or
cur.execute('INSERT INTO Precipitation VALUES(?,?,?,?)', (city_name, snow, total, days))

Where these are defined variables of the appropriate types
SQL: Selecting data from a table

Two-step process:

1. **SELECT** the columns we want:
   
   ```python
cur.execute('SELECT * FROM Precipitation')
   or

cur.execute('SELECT City, Snow FROM Precipitation')
```
SQL: Selecting data from a table

2. `fetchone()` retrieves the data one row at a time

   `fetchall()` retrieves all the data

We can then display or save the data, e.g.,

```
print(cur.fetchall())
```

```
data = cur.fetchall()
```

NOTE: data is retrieved as tuples e.g., ("St. John’s", 322.1)