Python — SQLite

Thomas Schwarz, SJ

Basics

- Need to import sqlite3
  - Comes installed with Python
- Create a *connector* to a database
- Then use a *cursor* to interact with the database

Basics

```
import sqlite3
```

importing the latest version

```
with sqlite3.connect('my_database.db') as connector:
    crsr = connector.cursor()
    print('connected to the database')
```

Basics

```
import sqlite3

with sqlite3.connect('my_database.db') as connector:
    crsr = connector.cursor()
    print('connected to the database')
```

Connect to a database
This will create the database
is necessary

Basics

With the cursor, you will send commands and obtain results

```
import sqlite3

with sqlite3.connect('my_database.db') as connector:
    crsr = connector.cursor()
    print('connected to the database')
```

Basics

```
connector.commit( )
```

If you do not commit, then the database will not reflect your updates!

Creating SQL Commands

- Hint:
  - If you are developing:
    - Create your sql commands as strings
    - Print them out to check for syntax
    - Because error messages are not very informative

Creating SQL Commands

- Recall:
  - To embed python values into strings
    - Use blueprint.format( )
    - Use f-strings
    - f-strings have an initial f:
      - `f'an example string'`
    - To embed variable values, put the variable name in curly brackets
      - `f'these are the values of {x} and {y}'`

Creating SQL Commands

- Creating tables:
  -
```
        sql_cmd1 = """
    CREATE TABLE IF NOT EXISTS salesperson (
       name VARCHAR(30),
       telephone VARCHAR(10)
       );
    """
        sql_cmd2 = """
    CREATE TABLE IF NOT EXISTS customer (
       name VARCHAR(30),
       address VARCHAR(60),
       telephone VARCHAR(10)
       );
    """
```

Creating SQL Commands

- Creating Tables:

```
sql_cmd3 = """
CREATE TABLE IF NOT EXISTS sales (
    item VARCHAR(30),
    client VARCHAR(30),
    seller VARCHAR(30),
    date VARCHAR(9),
    price INT
    );
"""
```

Creating SQL Commands

- Notice that sqlite does not have a special class for date / time / datetime
    - There are a number of functions to translate to the various formats
    - Standard format is Int, containing the Unix time (Seconds since January 1, 1970)

Creating SQL Commands

- Executing with cursor.execute

```
crsr.execute(sql_cmd1)
crsr.execute(sql_cmd2)
crsr.execute(sql_cmd3)
```

Creating SQL Commands

- Retrieving results
  - Use the fetchall method to obtain a list like object

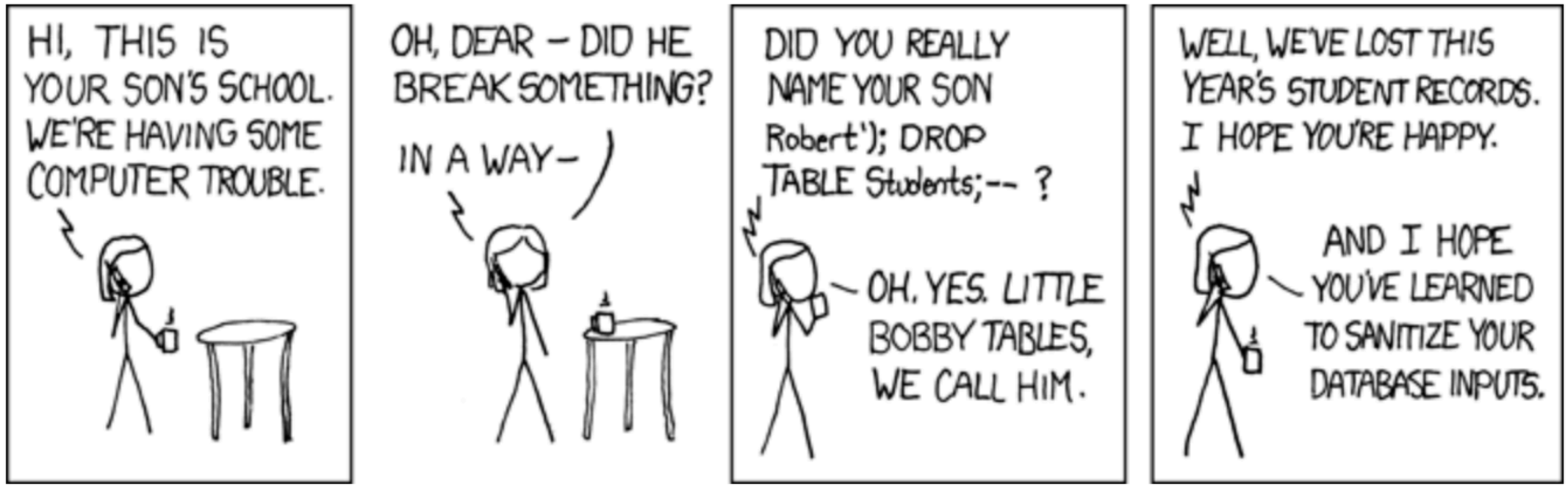
```
crsr.execute("SELECT * FROM customer")
print(crsr.fetchall( ))
```

  - Use the fetchone method to obtain a single row
  - Use as an iterator

```
crsr.execute("SELECT * FROM customer")
for item in crsr:
     print(item)
```

Preventing SQL Injection Attacks

- If you create sql statements from input provided by users, you can get into trouble



Preventing SQL Injection Attacks

- You can do this safer by using a placeholder, followed by a tuple of values
- `cur.execute("insert into lang values (?, ?)", ("C", 1972))`
-

```
lang_list = [
    ("Fortran", 1957),
    ("Python", 1991),
    ("Go", 2009),
]
cur.executemany("insert into lang values (?, ?)", lang_list)
```