Generating the "team" function of Parkrun

* IDE Used: Sublime Text 3 (ST3)
* All code executed from .py files
* All files in /parkrun directory
* All file names to be in lower case using pot\_hole case
* All variable names to use camelCase

# Create the database

↘ In ST3, create create\_parkrun\_db.py

As this is only a few lines and a one-off process, this could be done from the cmd line

#import the sqlite3 module

import sqlite3

#create a connection (a connection object) to a non-existing db #creates that db

myConn = sqlite3.connect("parkrun\_db.db")

↘ Save and execute/build create\_parkrun\_db.py (Ctrl+S, Ctrl+B in ST3)

If you do not get an error upon executing the file, you can assume parkrun\_db.db has been created

If you encounter an error, investigate.

Potential errors would be:

* spelling mistake of function name connect()
* absence of quotes around parameter of connect()
* trying to write to read-only volume/dir

Note that omitting the .db file extension will mean that your db may not be recognised as such.

↘ Close create\_parkrun\_db.py

# Create Tables

↘ In ST3, create test\_create\_parkrun\_tables.py

The objective is to create all the tables in a single file that is run once. However, it is good practice to breakdown the process to catch potential errors and make debugging easier.

As the process of creating a table is effectively repeated as many times as you have tables to create, it would be good practice to develop a single function and call it every time we create a table.

The function would simply need to return the SQL statement which would be run by the execute() function of the cursor.

An SQL CREATE statement has the following structure

CREATE TABLE "TableName" ( "Field1" PARAMETERS OF FIELD 1, "Field2" PARAMETERS OF FIELD, "Fieldn" PARAMETERS OF FIELD n)

We can break it down into two parts:

* the table name part
* the fields names part. We can break each field into two parts
  + the field name part
  + the field parameters part

The variable types best suited for the structure described above are:

* a string for the table name part
* a dictionary for the fields names part. In that dictionary, the field name is the key and the field parameters are the value for that key.

The IPO table corresponding to that process is as follows

|  |  |  |
| --- | --- | --- |
| INPUT | PROCESS | OUTPUT |
| Table name as string  Table fields and corresponding parameters as dictionary | **CREATE** an SQLStatement variable of String type and **ASSIGN** the initial constant part of the SQL CREATE Statement.  **APPEND** the string passed to the end of the SQLStatement variable including SQL syntax  **LOOP** through the keys of the dictionary passed and corresponding keys  **APPEND** each key, value pair to the SQLStatement variable including SQL syntax  **RETURN** the SQL-ready SQLStatement | SQL-ready CREATE TABLE Statement |

↘ Create the function on a test DB

Consider the following code[[1]](#footnote-1) as help towards creating your own function

Looping through the keys of a dictionary

For x in thisdict:

print(x)

Looping through the values of a dictionary

For x in thisdict:

print(thisdict[x])

Right-trimming a string

>>>a = “a string”

>>>print a[:-1]

a strin

import sqlite3

#create a connection (a connection object) the db

myConn = sqlite3.connect("testDB.db")

# create a cursor

myCursor = myConn.cursor()

#list the fields in a dictionary. the key is the field name

#the value is the list of parameters for that field

Chapterfields = {

"ChapterID": "INTEGER PRIMARY KEY AUTOINCREMENT",

"ChapterName": "TEXT",

"ChapterCourse": "TEXT",

"ChapterCreationDate": "DATE"

}

# build the table name part of the SQL CREATE TABLE statement

def createTable(tablename, fields):

sql\_statement = "CREATE TABLE \""

# the last (below is the opening ( of the fields part of

# the SQL CREATE TABLE statement

sql\_statement = sql\_statement + tablename + "\" ("

# build the fields names part of the SQL CREATE TABLE statement

for x in fields:

sql\_statement = sql\_statement+("\""+x+"\" "+fields[x]+",")

#Trim the trailing "'" and add a closing parenthesis

return sql\_statement[:-1]+")"

#Execute the SQL Statement

myCursor.execute(createTable("Chapter",Chapterfields))

↘ In ST3, create create\_parkrun\_tables.py to create the tables identified in the explore phase

# Populate (insert data into) the database

The least you handle/manipulate/edit data the better (less risk of error)

Create a .csv per table using Excel

Use real data when possible (Chapters, Events, RecentExerciseFrequency, RunnningClubs)

chapters.csv

Data cut and pasted into Excel from Chapters Web pages

Events.csv generated by hand (15 events, 3 per chapter, consecutives weeks 08/06 to 22/06 except Southbank, 01/06 to 15/05, 22/06 having been canceled)

1. All code examples from [https://www.w3schools.com](https://www.w3schools.com/) [↑](#footnote-ref-1)