sql modules:

1. three_matrices_data

This has a function *three_matrices(data_base)*. It imports the library module *my_database_sow_all2*, see next entry. It converts each of the h,t and s matrices into a data string (data = [(a,b,c,d,e,f,g,h)]) which is then written to an sql type database, as specificed in the function call, specifically into tables called outcomes1 and master_list.

2. my_database_sow_all2

This module adds data to a database through the function *sow_data(data_base,my_matrix,num)* where my_ matrix will be a 1x 8 list of [('00001', '00001', '00001', '1','00004', '2', '3', '2')] which are generated by another module (e.g. three_matrices_data) and num represents the outcomes number for the destination table in the sql database. The sql query that it executes is: *"INSERT INTO tablename VALUES(?,?,?,?,?,?,?,?);",my_matrix* . There is the capability of writing to 17 different tables all called outcomes and also the master_list which will store all unique(??) outcomes

3. qu_table_create_fn

This module has a function *create_table(data_base,num)* which runs an sql query to create a table in a database of the name specified in the function if it does not already exist. It will open or create an sqlite database of the name specified. The name of the table will be outcomes*num* . The sql query *is CREATE TABLE IF NOT EXISTS tablename (fieldnames)*. The table will have eight fields representing the real and imaginary coefficients of the elements in the 2 x 2 matrix:

Element1_r	REAL,
Element1_i	REAL,
Element2_r	REAL,
Element2_i	REAL,
Element3_r	REAL,
Element3_i	REAL,
Element4_r	REAL,
Element4_i	REAL

4. my_database_reap_all2

This module has a function *reap_data_outcomes(data_base,num)* which returns a complex number array (nx1x4). it runs the query *SELECT * FROM tablename.* The parameters of the functions have the usual meanings (see before)

5. query_table_delete_fnv1

This module has a function *delete_table(data_base,num)* which runs the query *DROP TABLE IF EXISTS tablename* .

6. q_db_table_u_records_fn

This module has a function *get_all(data_base,num)* which returns data as nx8 of numerical values from the specified outcomes table/ or master_list table in the database. It runs the sql query "SELECT DISTINCT * FROM tablename;"

7. q_db_table_records_fn

This module has the function *get_all(data_base,num)* which returns data as nx8 of numerical values. It runs the sql query "SELECT * FROM tablename;".

8. create_col

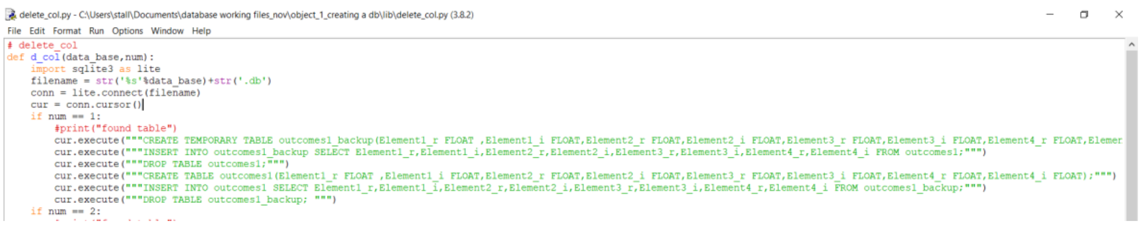
This module has the function *c_col(data_base,num)*. It runs the sql query """ALTER TABLE outcomes1 ADD strngy TEXT; """. This adds an empty column to the specified table which will contain a text string called strngy. This string will be used to identify duplicates in the table.

9. generate_strngy_data

This module has the function make_*strng(data_base,num)*. It runs the sql query "UPDATE outcomes1 SET strngy = Element1_i || Element1_r || Element2_i || Element2_r || Element3_i || Element3_r || Element4_i || Element4_r;". It fills the empty column created by module 13 with the data which will be then filtered for unique data.

10. delete_col

This module has the function *d_col(data_base,num)* and it stores the columns that are to remain in a temporary table it then deletes the original table which has all the columns and then rewrites the temporary table as the original table. There are six sql queries in this module:


delete_col.py - C:\Users\stall\Documents\database working files_nov\object_1_creating a db\lib\delete_col.py (3.8.2)
File Edit Format Run Options Window Help

```
# delete_col
def d_col(data_base,num):
    import sqlite3 as lite
    filename = str('%s'%data_base)+str('.db')
    conn = lite.connect(filename)
    cur = conn.cursor()
    if num == 1:
        #print("found table")
        cur.execute("""CREATE TEMPORARY TABLE outcomes1_backup(Element1_r FLOAT ,Element1_i FLOAT,Element2_r FLOAT,Element2_i FLOAT,Element3_r FLOAT,Element3_i FLOAT,Element4_r FLOAT,Elemen
        cur.execute("""INSERT INTO outcomes1_backup SELECT Element1_r,Element1_i,Element2_r,Element2_i,Element3_r,Element3_i,Element4_r,Element4_i FROM outcomes1;""")
        cur.execute("""DROP TABLE outcomes1;""")
        cur.execute("""CREATE TABLE outcomes1(Element1_r FLOAT ,Element1_i FLOAT,Element2_r FLOAT,Element2_i FLOAT,Element3_r FLOAT,Element3_i FLOAT,Element4_r FLOAT,Element4_i FLOAT);""")
        cur.execute("""INSERT INTO outcomes1 SELECT Element1_r,Element1_i,Element2_r,Element2_i,Element3_r,Element3_i,Element4_r,Element4_i FROM outcomes1_backup;""")
        cur.execute("""DROP TABLE outcomes1_backup; """)
    if num == 2:
```

11. del_intersction_t

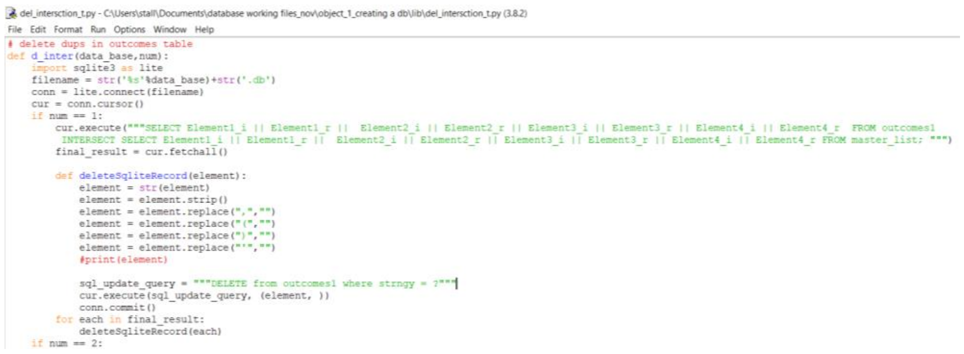
This module has the function *d_inter(data_base,num)* delete from the outcomes table the duplications between the master_list and outcomes table before the outcomes table is written to the master_list. This should keep the master_list free of duplicates.

It uses an sql query which finds the fields in the outcomes table which are also in the master_list:

SELECT Element1_i || Element1_r || Element2_i || Element2_r || Element3_i || Element3_r || Element4_i || Element4_r FROM outcomes1

INTERSECT SELECT Element1_i || Element1_r || Element2_i || Element2_r || Element3_i || Element3_r || Element4_i || Element4_r FROM master_list;

It then runs a function which deletes (via an sql query) those records from the outcomes table:


del_intersction_t.py - C:\Users\stall\Documents\database working files_nov\object_1_creating a db\lib\del_intersction_t.py (3.8.2)
File Edit Format Run Options Window Help

```
# delete dups in outcomes table
def d_inter(data_base,num):
    import sqlite3 as lite
    filename = str('%s'%data_base)+str('.db')
    conn = lite.connect(filename)
    cur = conn.cursor()
    if num == 1:
        cur.execute("""SELECT Element1_i || Element1_r ||  Element2_i || Element2_r || Element3_i || Element3_r || Element4_i || Element4_r  FROM outcomes1
         INTERSECT SELECT Element1_i || Element1_r ||  Element2_i || Element2_r || Element3_i || Element3_r || Element4_i || Element4_r FROM master_list; """)
        final_result = cur.fetchall()

        def deleteSqliteRecord(element):
            element = str(element)
            element = element.strip()
            element = element.replace(",","")
            element = element.replace("(","")
            element = element.replace(")","")
            element = element.replace("'","")
            #print(element)

            sql_update_query = """DELETE from outcomes1 where strngy = ?"""
            cur.execute(sql_update_query, (element, ))
            conn.commit()
        for each in final_result:
            deleteSqliteRecord(each)
    if num == 2:
```