Project 2 part(a)

This database will store all the 1 qubit gates that can be made from combinations (dot products) of the h, t and s gates. In addition, it will also store the combination that generates each of these strings in terms of h, t and s gates. This will be part (a) of the project. The aim is to then find the gates or its best approximation in the database (using the Solvay-Kitaev algorithm) and then find the h, tand s combination which can be used to make that gate. This will be part(b) of the project. The main program is qu_new2.py

qu_new2.py - C:\Users\stall\Documents\future-Learm\using_database\new_approach_new1\qu_new2.py (3.8.2)

File Edit Format Run Options Window Help

```
def main():
    import sqlite3 as lite
    import lib1.qu_table_create_1 as create1 #create_table_one(database,num)
    import lib1.qu_table_create_2 as create2
    import lib1.qu_table_delete_1 as delete1#delete_table1(database,num)
    import lib1.qu_table_delete_2 as delete2
    import lib1.three_matrices_data as three_matrix
    import lib1.reap1 as reap1 # function name is reap_data1_outcomes(data_base,num)
    import lib1.reap1o_m as reap1om # function name is reap_data1_outcomes(data_base,num)
    import lib1.reformat_array_m as rf
    import lib1.del_duplicates_t1 as dd  # del_dup1(data_base,num)
    import lib1.dot_prod_new as dp #dot_product("qd1",1)
    import lib1.del_intersction as di  # d_inter(data_base,num)

    import lib1.sow_data1 as sow   # sow_data1(data_base,my_matrix,num)
    import numpy as np
    import math
    import cmath


    d_base = str(input("enter name of database  "))
    ans1 = str(input("create new table?  "))
    if ans1 == "y":
        nm = int(input("enter number of your table  "))
        create1.create_table_one(d_base,nm)
        create2.create_table_two(d_base,nm)
    ans0 = str(input("create new master table?  "))
    if ans0 == "y":
        create1.create_table_one(d_base,'m')
        create2.create_table_two(d_base,'m')

    ans2 = input("delete table?  ")
    if ans2 == "y":
        delete1.delete_table1(d_base,nm)
        delete2.delete_table2(d_base,nm)
    ans3 = input("does data exist in the data base ? y/n  ")
    if ans3 == 'n':
        three_matrix.three_matrices(d_base)

    max_num = int(input("What is the final outcome number, this will be the final value"))
    start = int(input("Numberof last table with data, this will be the starting value?  "))
    for i in range(start,(max_num)):
        create1.create_table_one(d_base,i+1)
        create2.create_table_two(d_base,i+1)
        print("using table",i,"to generate table ",i+1)
        dp.dot_product(d_base,i) # the two arrays of these will need to be written to new tables
        di.d_inter(d_base,i+1) # removing any overlap in new table with master _list
        my_list= reap1.reap_data1_outcomes(d_base,i+1)

        #reap outcomes and then sow into master
        my_new = reap1om.reap_data1_outcomes(d_base,i+1)
        sow.sow_data1(d_base,my_new,"m")



main()
```

There is a new library lib1 with modified modules

The key module is the dot_prod_new module which instead of returning matrices to the main program now writes them directly to the database. This was necessary as there are now two sets of data to be stored, the coefficients of the matrix elements and also the h, t and s combinations. This module is analysed as a subprogram here rather than a module as it calls on many modules in the lib1 folder