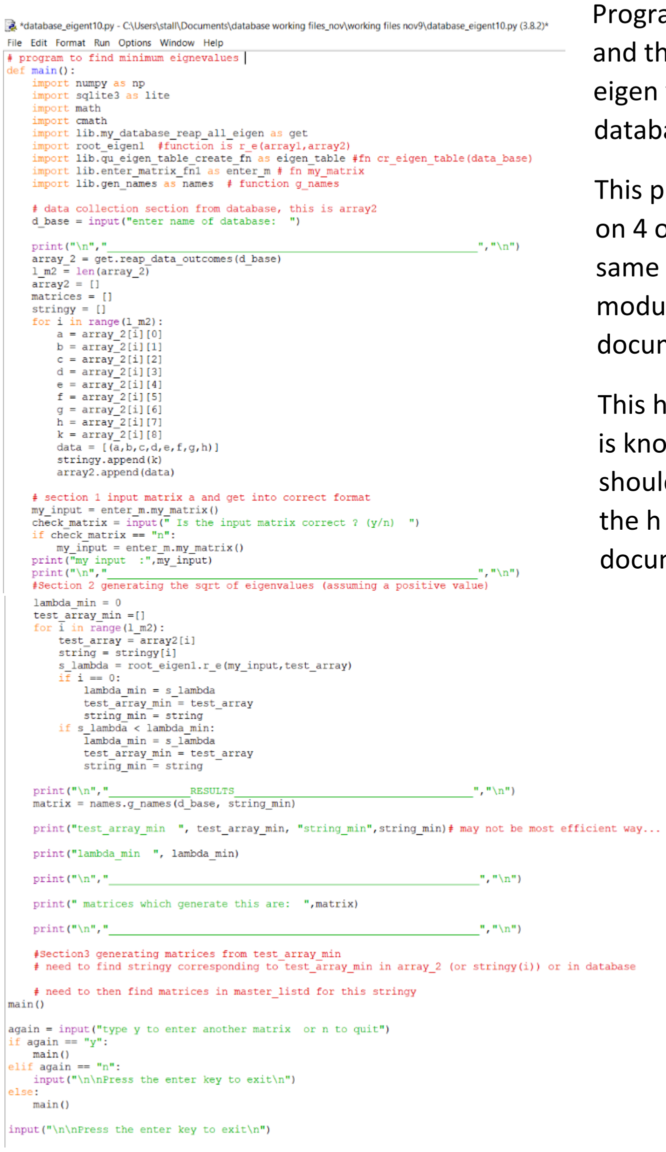
Project 2c programs

Contents

Program 1 **database_eigent10**



database_eigent10.py - C:\Users\stall\Documents\database working files_nov\working files nov9\database_eigent10.py (3.8.2)
File Edit Format Run Options Window Help

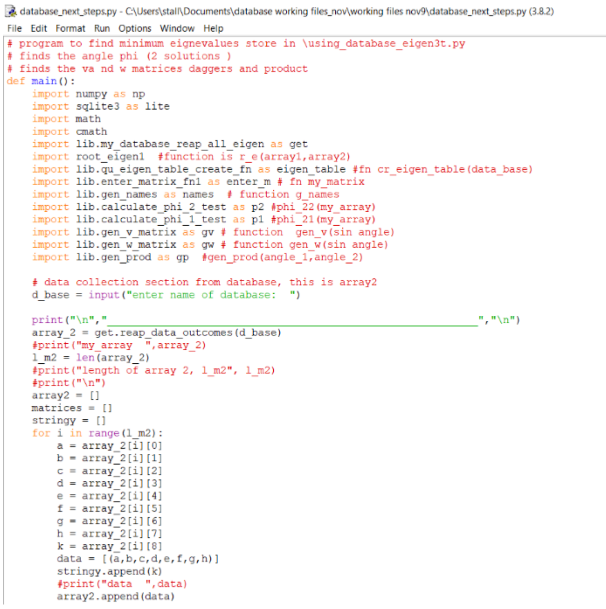
```
# program to find minimum eignevalues
def main():
    import numpy as np
    import sqlite3 as lite
    import math
    import cmath
    import lib.my_database_reap_all_eigen as get
    import root_eigen1  #function is r_e(array1,array2)
    import lib.qu_eigen_table_create_fn as eigen_table #fn cr_eigen_table(data_base)
    import lib.enter_matrix_fn1 as enter_m # fn my_matrix
    import lib.gen_names as names  # function g_names

    # data collection section from database, this is array2
    d_base = input("enter name of database:  ")

    print("\n","____________________________________________________________","\n")
    array_2 = get.reap_data_outcomes(d_base)
    l_m2 = len(array_2)
    array2 = []
    matrices = []
    stringy = []
    for i in range(l_m2):
        a = array_2[i][0]
        b = array_2[i][1]
        c = array_2[i][2]
        d = array_2[i][3]
        e = array_2[i][4]
        f = array_2[i][5]
        g = array_2[i][6]
        h = array_2[i][7]
        k = array_2[i][8]
        data = [(a,b,c,d,e,f,g,h)]
        stringy.append(k)
        array2.append(data)

    # section 1 input matrix a and get into correct format
    my_input = enter_m.my_matrix()
    check_matrix = input(" Is the input matrix correct ? (y/n)  ")
    if check_matrix == "n":
        my_input = enter_m.my_matrix()
    print("my input  :",my_input)
    print("\n","____________________________________________________________","\n")
    #Section 2 generating the sqrt of eigenvalues (assuming a positive value)
    lambda_min = 0
    test_array_min =[]
    for i in range(l_m2):
        test_array = array2[i]
        string = stringy[i]
        s_lambda = root_eigen1.r_e(my_input,test_array)
        if i == 0:
            lambda_min = s_lambda
            test_array_min = test_array
            string_min = string
        if s_lambda < lambda_min:
            lambda_min = s_lambda
            test_array_min = test_array
            string_min = string

    print("\n","_____________RESULTS_____________________________________","\n")
    matrix = names.g_names(d_base, string_min)

    print("test_array_min  ", test_array_min, "string_min",string_min)# may not be most efficient way...

    print("lambda_min  ", lambda_min)

    print("\n","____________________________________________________________","\n")

    print(" matrices which generate this are:  ",matrix)

    print("\n","____________________________________________________________","\n")

    #Section3 generating matrices from test_array_min
    # need to find stringy corresponding to test_array_min in array_2 (or stringy(i)) or in database

    # need to then find matrices in master_listd for this stringy
main()

again = input("type y to enter another matrix  or n to quit")
if again == "y":
    main()
elif again == "n":
    input("\n\nPress the enter key to exit\n")
else:
    main()

input("\n\nPress the enter key to exit\n")
```

Program 1 will allow a user to enter a 2x2 matrix and the matrix which corresponds to the minimum eigen value of the adjacent matrices in the database is returned.

This program is **database_eigent10.py** which calls on 4 other modules. Root_eigen1 is saved at the same level and this also call on an additional 2 modules, These are documented in another document.

This has been tested by entering in a matrix which is known to be in the database. The program should find that value. This was carried out with the h matrix and found to be true. This is documented in the test documents

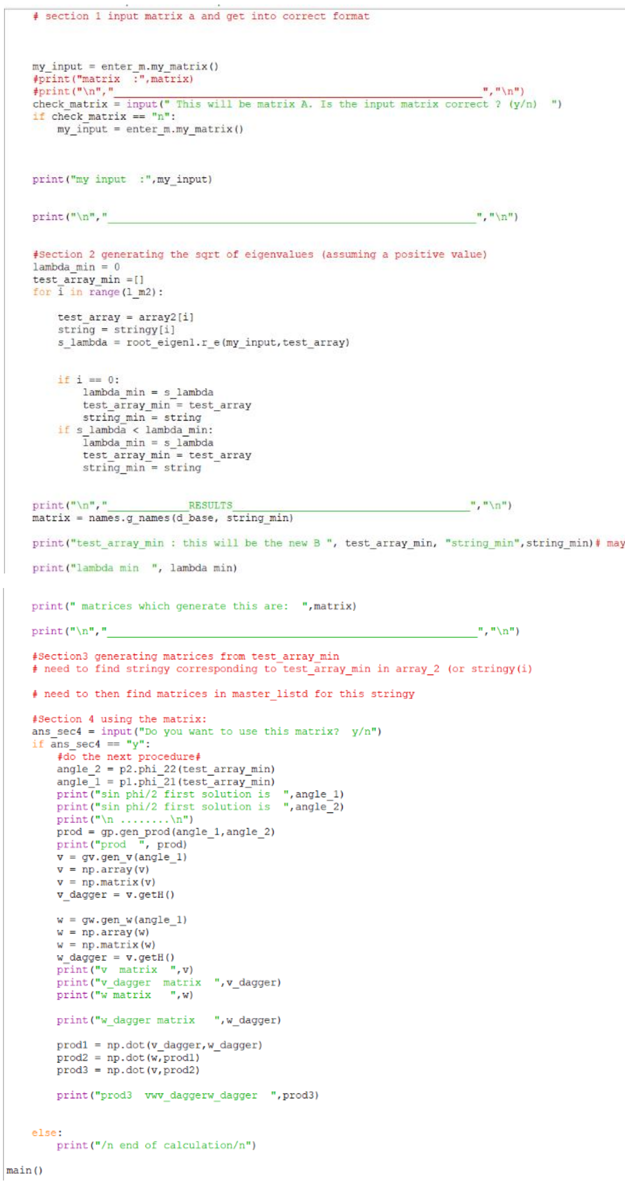
Program 2: database_next_steps.


database_next_steps.py - C:\Users\stall\Documents\database working files_nov\working files nov9\database_next_steps.py (3.8.2)

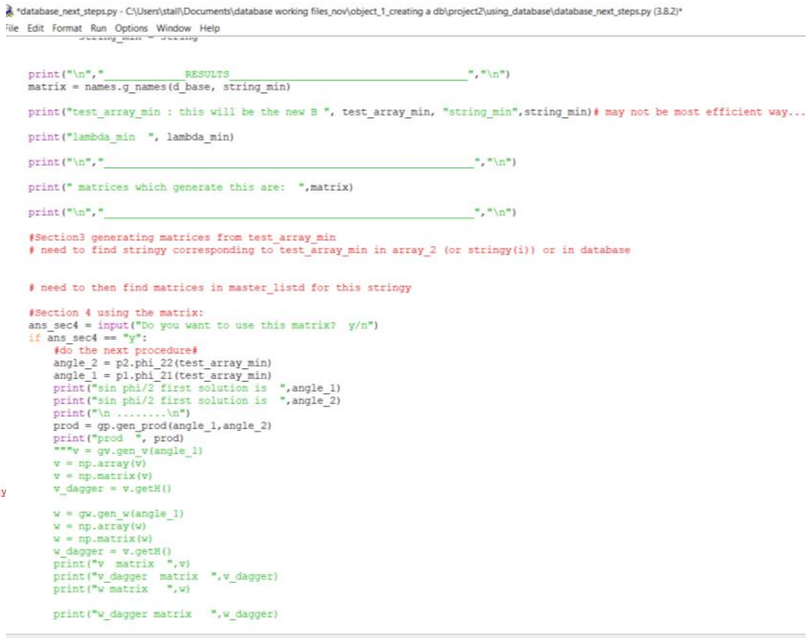
```
# program to find minimum eignevalues store in \using_database_eigen3t.py
# finds the angle phi (2 solutions )
# finds the va nd w matrices daggers and product
def main():
    import numpy as np
    import sqlite3 as lite
    import math
    import cmath
    import lib.my_database_reap_all_eigen as get
    import root_eigen1  #function is r_e(array1,array2)
    import lib.qu_eigen_table_create_fn as eigen_table #fn cr_eigen_table(data_base)
    import lib.enter_matrix_fn1 as enter_m # fn my_matrix
    import lib.gen_names as names  # function g_names
    import lib.calculate_phi_2_test as p2 #phi_22(my_array)
    import lib.calculate_phi_1_test as p1 #phi_21(my_array)
    import lib.gen_v_matrix as gv # function  gen_v(sin angle)
    import lib.gen_w_matrix as gw # function gen_w(sin angle)
    import lib.gen_prod as gp  #gen_prod(angle_1,angle_2)

    # data collection section from database, this is array2
    d_base = input("enter name of database:  ")

    print("\n","____________________________________________________________","\n")
    array_2 = get.reap_data_outcomes(d_base)
    #print("my_array  ",array_2)
    l_m2 = len(array_2)
    #print("length of array 2, l_m2", l_m2)
    #print("\n")
    array2 = []
    matrices = []
    stringy = []
    for i in range(l_m2):
        a = array_2[i][0]
        b = array_2[i][1]
        c = array_2[i][2]
        d = array_2[i][3]
        e = array_2[i][4]
        f = array_2[i][5]
        g = array_2[i][6]
        h = array_2[i][7]
        k = array_2[i][8]
        data = [(a,b,c,d,e,f,g,h)]
        stringy.append(k)
        #print("data  ",data)
        array2.append(data)
```

Program 2 is called **database_next_steps.py**.

This program finds the set of minimum eignevalues and then finds the angle phi . It then goes on to find the v and w matrices , their daggers of an input matrix and the dot product of all these calculated matrices

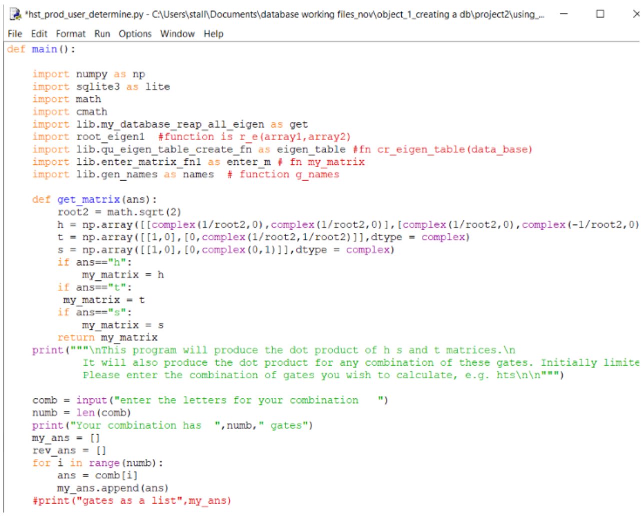


```
    # section 1 input matrix a and get into correct format


    my_input = enter_m.my_matrix()
    #print("matrix  :",matrix)
    #print("\n","____________________________________________________________","\n")
    check_matrix = input(" This will be matrix A. Is the input matrix correct ? (y/n)  ")
    if check_matrix == "n":
        my_input = enter_m.my_matrix()


    print("my input  :",my_input)


    print("\n","____________________________________________________________","\n")


    #Section 2 generating the sqrt of eigenvalues (assuming a positive value)
    lambda_min = 0
    test_array_min =[]
    for i in range(l_m2):

        test_array = array2[i]
        string = stringy[i]
        s_lambda = root_eigen1.r_e(my_input,test_array)


        if i == 0:
            lambda_min = s_lambda
            test_array_min = test_array
            string_min = string
        if s_lambda < lambda_min:
            lambda_min = s_lambda
            test_array_min = test_array
            string_min = string


    print("\n","_____________RESULTS______________________________________","\n")
    matrix = names.g_names(d_base, string_min)

    print("test_array_min : this will be the new B ", test_array_min, "string_min",string_min)# may

    print("lambda min  ", lambda min)


    print(" matrices which generate this are:  ",matrix)

    print("\n","____________________________________________________________","\n")

    #Section3 generating matrices from test_array_min
    # need to find stringy corresponding to test_array_min in array_2 (or stringy(i)

    # need to then find matrices in master_listd for this stringy

    #Section 4 using the matrix:
    ans_sec4 = input("Do you want to use this matrix?  y/n")
    if ans_sec4 == "y":
        #do the next procedure#
        angle_2 = p2.phi_22(test_array_min)
        angle_1 = p1.phi_21(test_array_min)
        print("sin phi/2 first solution is  ",angle_1)
        print("sin phi/2 first solution is  ",angle_2)
        print("\n ........\n")
        prod = gp.gen_prod(angle_1,angle_2)
        print("prod  ", prod)
        v = gv.gen_v(angle_1)
        v = np.array(v)
        v = np.matrix(v)
        v_dagger = v.getH()

        w = gw.gen_w(angle_1)
        w = np.array(w)
        w = np.matrix(w)
        w_dagger = v.getH()
        print("v  matrix  ",v)
        print("v_dagger  matrix  ",v_dagger)
        print("w matrix   ",w)

        print("w_dagger matrix   ",w_dagger)

        prod1 = np.dot(v_dagger,w_dagger)
        prod2 = np.dot(w,prod1)
        prod3 = np.dot(v,prod2)

        print("prod3  vwv_daggerw_dagger  ",prod3)


    else:
        print("/n end of calculation/n")

main()
```

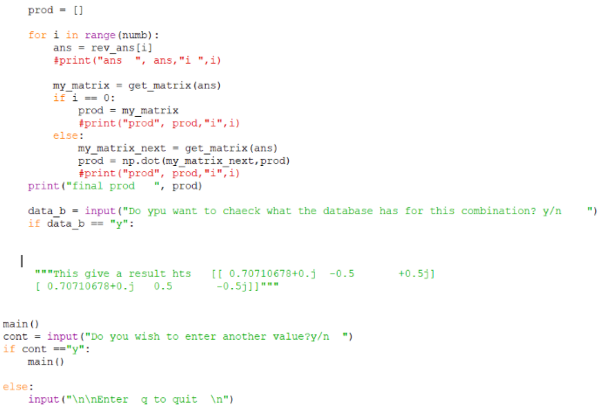

database_next_steps.py - C:\Users\stall\Documents\database working files_nov\object_1_creating a db\project2\using_database\database_next_steps.py (3.8.2)

```
            string_min = string


    print("\n","_____________RESULTS______________________________________","\n")
    matrix = names.g_names(d_base, string_min)

    print("test_array_min : this will be the new B ", test_array_min, "string_min",string_min)# may not be most efficient way...

    print("lambda_min  ", lambda_min)

    print("\n","____________________________________________________________","\n")

    print(" matrices which generate this are:  ",matrix)

    print("\n","____________________________________________________________","\n")

    #Section3 generating matrices from test_array_min
    # need to find stringy corresponding to test_array_min in array_2 (or stringy(i)) or in database


    # need to then find matrices in master_listd for this stringy

    #Section 4 using the matrix:
    ans_sec4 = input("Do you want to use this matrix?  y/n")
    if ans_sec4 == "y":
        #do the next procedure#
        angle_2 = p2.phi_22(test_array_min)
        angle_1 = p1.phi_21(test_array_min)
        print("sin phi/2 first solution is  ",angle_1)
        print("sin phi/2 first solution is  ",angle_2)
        print("\n ........\n")
        prod = gp.gen_prod(angle_1,angle_2)
        print("prod  ", prod)
        """v = gv.gen_v(angle_1)
        v = np.array(v)
        v = np.matrix(v)
        v_dagger = v.getH()

        w = gw.gen_w(angle_1)
        w = np.array(w)
        w = np.matrix(w)
        w_dagger = v.getH()
        print("v  matrix  ",v)
        print("v_dagger  matrix  ",v_dagger)
        print("w matrix   ",w)

        print("w_dagger matrix   ",w_dagger)
```

Program 3: hst_prod_user_determine

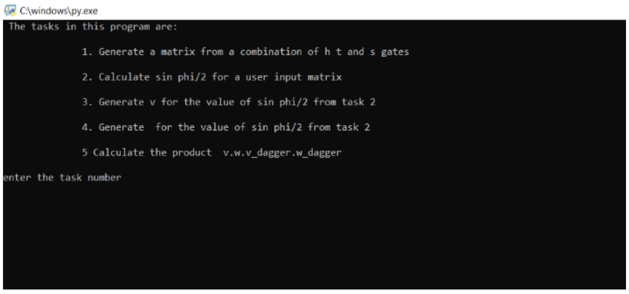

*hst_prod_user_determine.py - C:\Users\stall\Documents\database working files_nov\object_1_creating a db\project2\using_...
File Edit Format Run Options Window Help

```
def main():

    import numpy as np
    import sqlite3 as lite
    import math
    import cmath
    import lib.my_database_reap_all_eigen as get
    import root_eigen1  #function is r_e(array1,array2)
    import lib.qu_eigen_table_create_fn as eigen_table #fn cr_eigen_table(data_base)
    import lib.enter_matrix_fn1 as enter_m # fn my_matrix
    import lib.gen_names as names  # function g_names

    def get_matrix(ans):
        root2 = math.sqrt(2)
        h = np.array([[complex(1/root2,0),complex(1/root2,0)],[complex(1/root2,0),complex(-1/root2,0)
        t = np.array([[1,0],[0,complex(1/root2,1/root2)]],dtype = complex)
        s = np.array([[1,0],[0,complex(0,1)]],dtype = complex)
        if ans=="h":
            my_matrix = h
        if ans=="t":
         my_matrix = t
        if ans=="s":
            my_matrix = s
        return my_matrix
    print("""\nThis program will produce the dot product of h s and t matrices.\n
            It will also produce the dot product for any combination of these gates. Initially limite
            Please enter the combination of gates you wish to calculate, e.g. hts\n\n""")

    comb = input("enter the letters for your combination   ")
    numb = len(comb)
    print("Your combination has  ",numb," gates")
    my_ans = []
    rev_ans = []
    for i in range(numb):
        ans = comb[i]
        my_ans.append(ans)
    #print("gates as a list",my_ans)
```



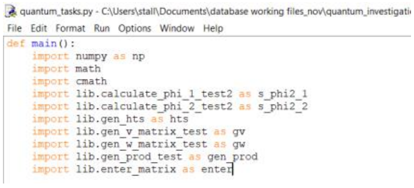
```
    prod = []

    for i in range(numb):
        ans = rev_ans[i]
        #print("ans  ", ans,"i ",i)

        my_matrix = get_matrix(ans)
        if i == 0:
            prod = my_matrix
            #print("prod", prod,"i",i)
        else:
            my_matrix_next = get_matrix(ans)
            prod = np.dot(my_matrix_next,prod)
            #print("prod", prod,"i",i)
    print("final prod   ", prod)

    data_b = input("Do ypu want to chaeck what the database has for this combination? y/n    ")
    if data_b == "y":


     """This give a result hts   [[ 0.70710678+0.j  -0.5       +0.5j]
     [ 0.70710678+0.j   0.5       -0.5j]]"""


main()
cont = input("Do you wish to enter another value?y/n  ")
if cont =="y":
    main()

else:
    input("\n\nEnter  q to quit  \n")
```

Program 4 Quantum investigations:



```
 The tasks in this program are:

             1. Generate a matrix from a combination of h t and s gates

             2. Calculate sin phi/2 for a user input matrix

             3. Generate v for the value of sin phi/2 from task 2

             4. Generate  for the value of sin phi/2 from task 2

             5 Calculate the product  v.w.v_dagger.w_dagger

enter the task number
```

This program is called quantum_tasks and gives a menu of tasks, as shown on the left. It incorporates parts of the earlier programs. It imports the following modules:


quantum_tasks.py - C:\Users\stall\Documents\database working files_nov\quantum_investigati
File Edit Format Run Options Window Help

```
def main():
    import numpy as np
    import math
    import cmath
    import lib.calculate_phi_1_test2 as s_phi2_1
    import lib.calculate_phi_2_test2 as s_phi2_2
    import lib.gen_hts as hts
    import lib.gen_v_matrix_test as gv
    import lib.gen_w_matrix_test as gw
    import lib.gen_prod_test as gen_prod
    import lib.enter_matrix as enter
```