Project 2b and 2c modules

1. my_database_reap_all_eigen

This module has a function *reap_data_outcomes(data_base)* and it returns n x 1x8 number array. It runs the sql query:

"SELECT * FROM master_list;"

2. generate_matrix_from_datafn

It uses the function *c_data_gen*(myarray) which takes the list of nx 1x8 lists and reconstruct the complex numbers to output a list of 1x4 matrices which can be used in dot product module,e.g.:

input data [[(1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 2.0)], [(1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 1.0, 2.0)]] .

example of output data: array([[[1.+0.j, 1.+1.j],

[0.+0.j, 1.+2.j]],

[[1.+1.j, 1.+2.j],

[1.+1.j, 1.+2.j]]])

3. generate_data_filec_r

It uses the function *data_gen(myarray)* function will take a list of 1x4 matrices of complex numbers and return a list of lists where each element has 8 numbers r,and im coefficients of the numbers in the 2x2 matrix, e.g. :

my_list = [[1.0, (1+1j),0, (1+2j)], [(1+1j), (1+2j), (1+1j), (1+2j)]]

will produce an output which has the form

[[(1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 2.0)],[(1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 1.0, 2.0)]].

4. root_eigen1

This has a function r_e(my_array1,my_array2) which uses gen_matrix_from_datafn as make_matrix and generate_data_filec_r as make_data. The first of these is outlined above, the second is outlined in project 1 (and above).

It reformats my_array1 and my_array2 as lists of matrices (in this case each list will have 1 matrix) called first (this is the input matrix) and next (this is a matrix taken from the database) . It takes the difference first – next and assigns the result to the variable c . It then reformats this as an numpy matrix and calls it x. It uses the numpy library to take the Hermitian conjugate (z).

the dot product of w and z is performed and the eigenvalues calculated using the numpy library

it then returns the square root of the maximum eigenvalue.

root_eigen1.py - C:\Users\stall\Documents\future-Learm\using_database\root_eigen1.py (3.8.2)

File Edit Format Run Options Window Help

```
#this takes 2 arrays of 1x8 shape representing
#two complex 2x2 matrices

def r_e(my_array1,my_array2):

    import numpy as np
    import cmath
    import math
    import lib.gen_matrix_from_datafn as make_matrix
    import lib.generate_data_filec_r as make_data
    #


    first = make_matrix.c_data_gen(my_array1) # this will be input matrix

    next_ = make_matrix.c_data_gen(my_array2) # this will be turn[i]
    #print("first  ",first)
    #print("next_  ",next_)
    c = first - next_
    #print("c  ",c)
    x = np.matrix(c)
    #print("x  ",x)
    z = x.getH()
    #print("z  ",z)
    w = np.dot(x,z)
    #print("w  ",w)

    eigenvalues, eigenvectors = np.linalg.eigh(w)
    #print("eigenvalues  ",eigenvalues)
    my_eigenvalue = max(eigenvalues)

    my_eigen = math.sqrt(my_eigenvalue)
    return my_eigen
```

Testing of modules is included in

5. enter_matrix_fn1

This has a procedure called my_matrix()

enter_matrix_fn1.py - C:\Users\stall\Documents\database working files_nov\working files nov9\lib\enter_matrix_fn1.py (3.8.2)

File Edit Format Run Options Window Help

```
    for n in range(4):
        print(n+1,"value   ")
        q1 = input("is the number in complex form? if so Enter as complex , e.g 1.0+1.0j, enter y or  n\n")
        if q1 == "y":

            a = complex(input("enter your value \n "))
            print("a  ",a)
            number.append(a)
        elif q1 == "n":
            q3 = input("is your value in exponetial (exp (i*n*pi))  form, n can be fractional and negative ?  y/n \n ")
            if q3 =="y":
                nn = float(input("enter your value of n \n "))
                ag = nn*(math.pi)
                a = complex(math.cos(ag),math.sin(ag))
                print("a  ",a)
                number.append(a)
        else:
            print(" you have entered an invalid character\n")



    #entries = list(map(complex,number))   #input().split()
    #print("entries  ",entries)

    # For printing the matrix
    matrix = np.array(number).reshape(R, C)



    a = matrix[0][0].real
    b = matrix[0][0].imag
    c = matrix[0][1].real
    d = matrix[0][1].imag
    e = matrix[1][0].real
    f = matrix[1][0].imag
    g = matrix[1][1].real
    h = matrix[1][1].imag


    data = [a,b,c,d,e,f,g,h]
    print("matrix  :",matrix)
    print("\n","_____________________________________________________","\n")

    return data
```

6. gen_names

This has a function *g_names(data_base,string*) which runs SELECT matrices from master_listd where stringy = ? using the command cur.execute(sql_update_query, (string,))

7. calculate_phi_2_test

   This module has *phi_22(my_array)* which return phi (solution 2)

8. calculate_phi_1_test

   This module has *phi_21(my_array)* which return phi (solution 1)

9. gen_v_matrix 1

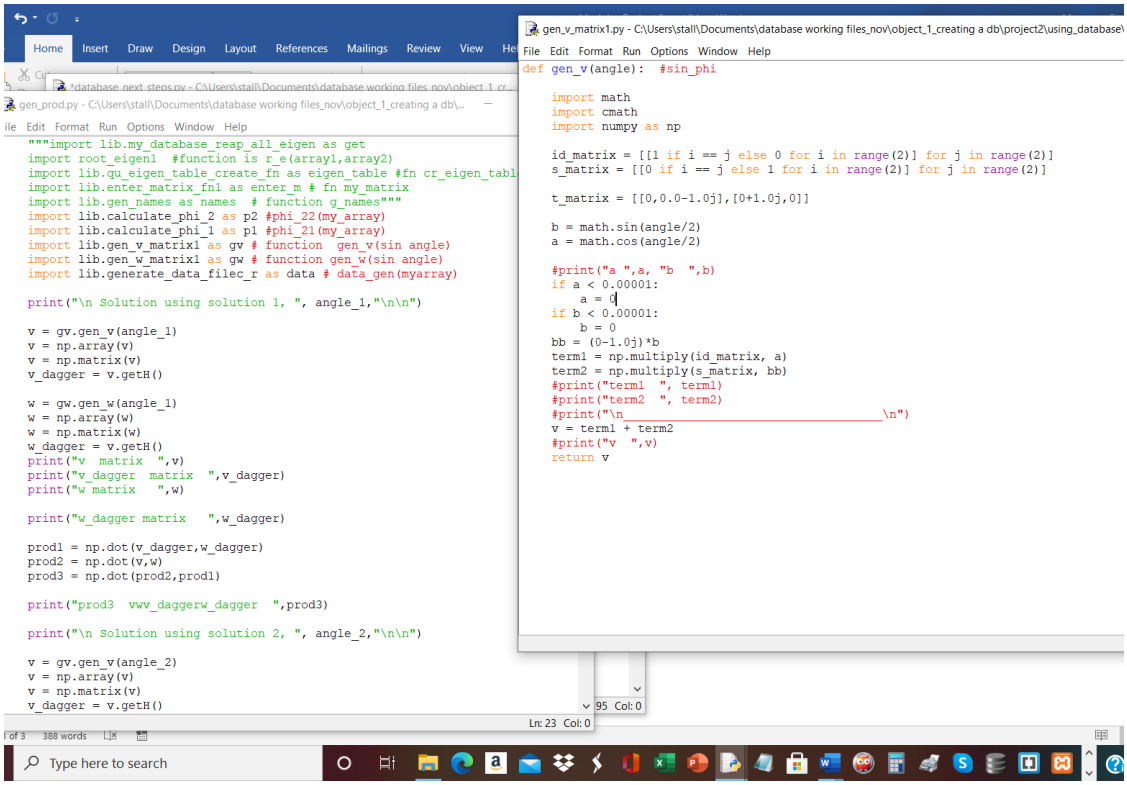
   This module has function gen_v(angle)

10. gen_w_matrix 1

    This module has function gen_w(angle)

11. gen_prod

    This module has gen_prod(angle_1,angle_2)



Not used

1. qu_eigen_table_create_fn #fn cr_eigen_table(data_base)