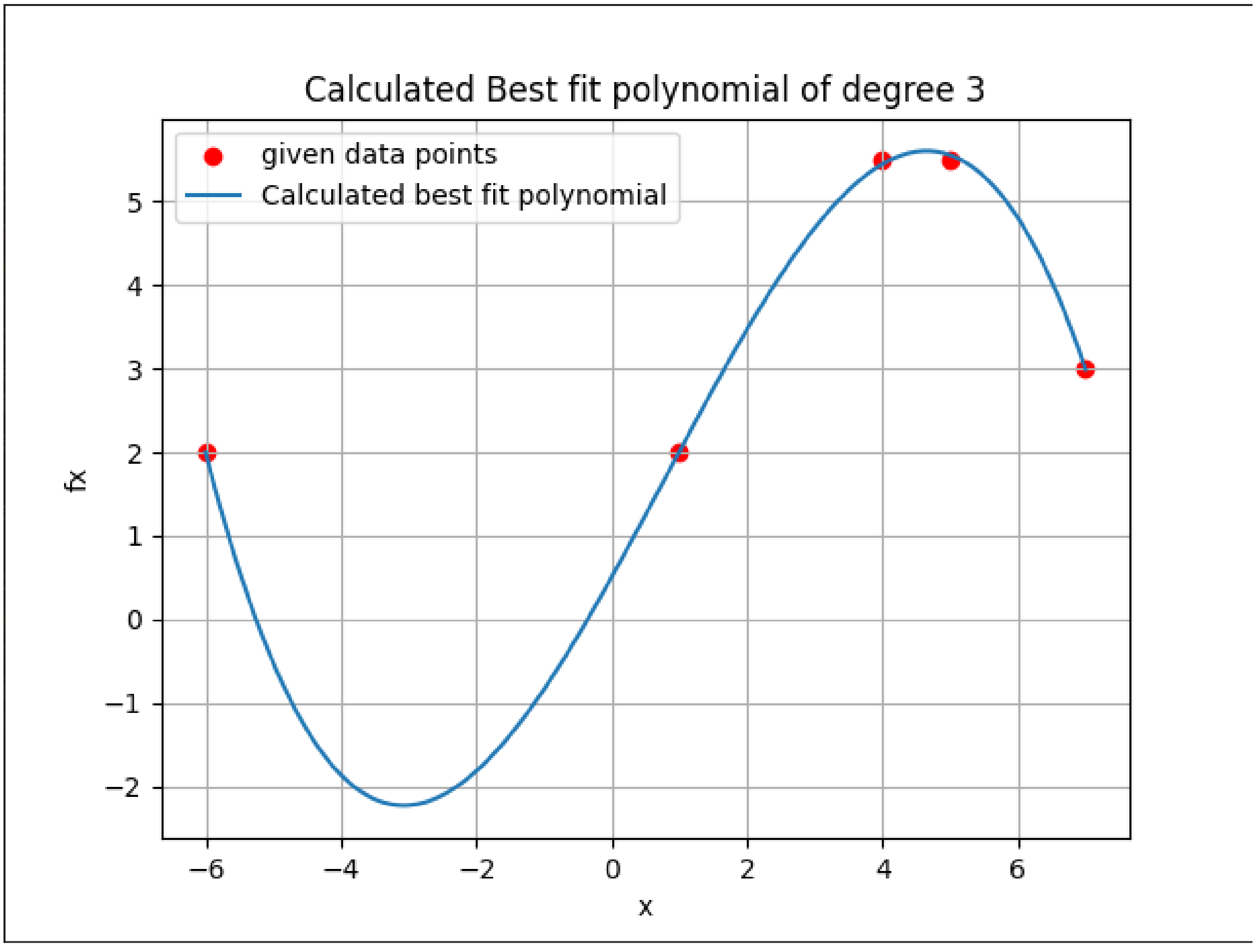
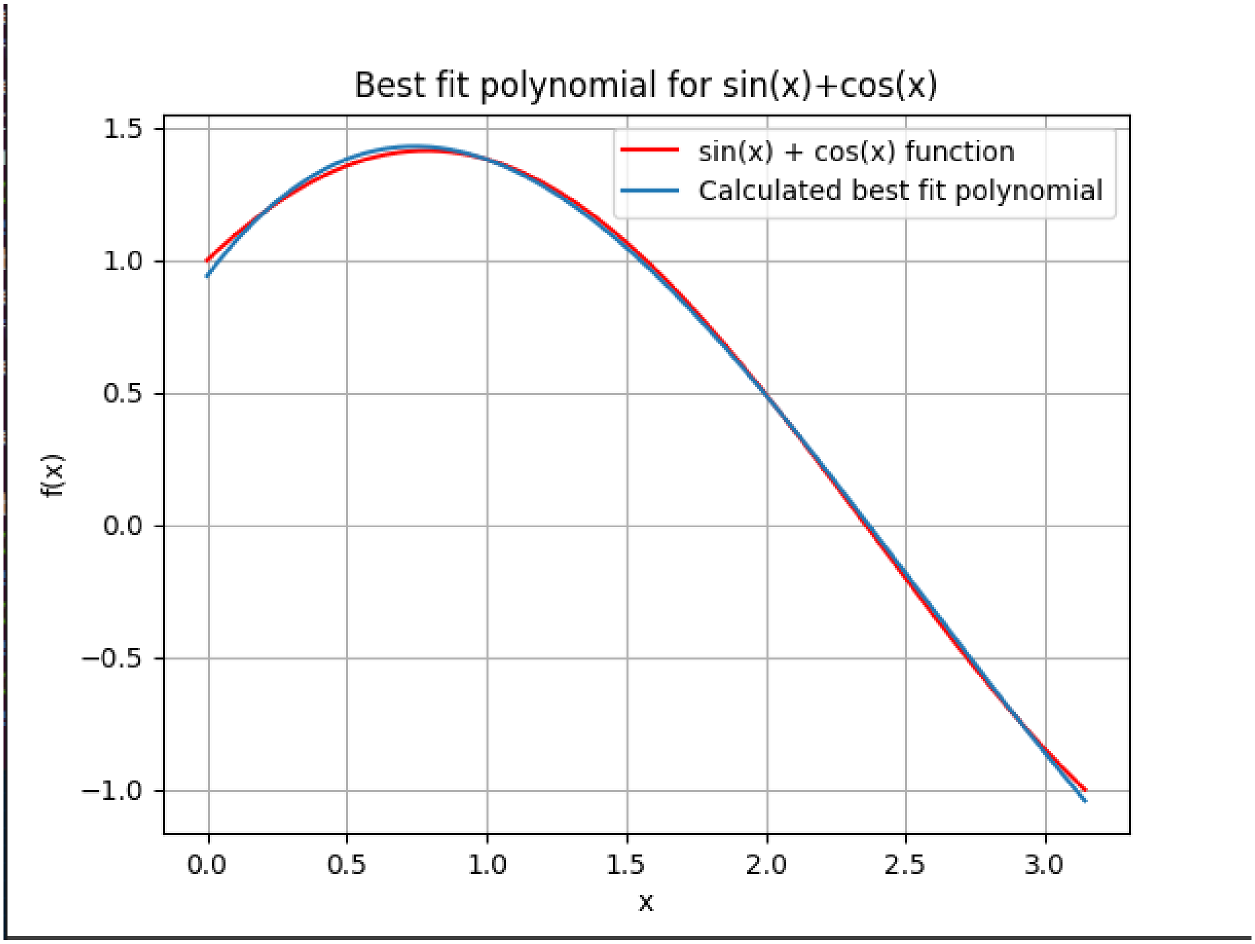
Report lab 5

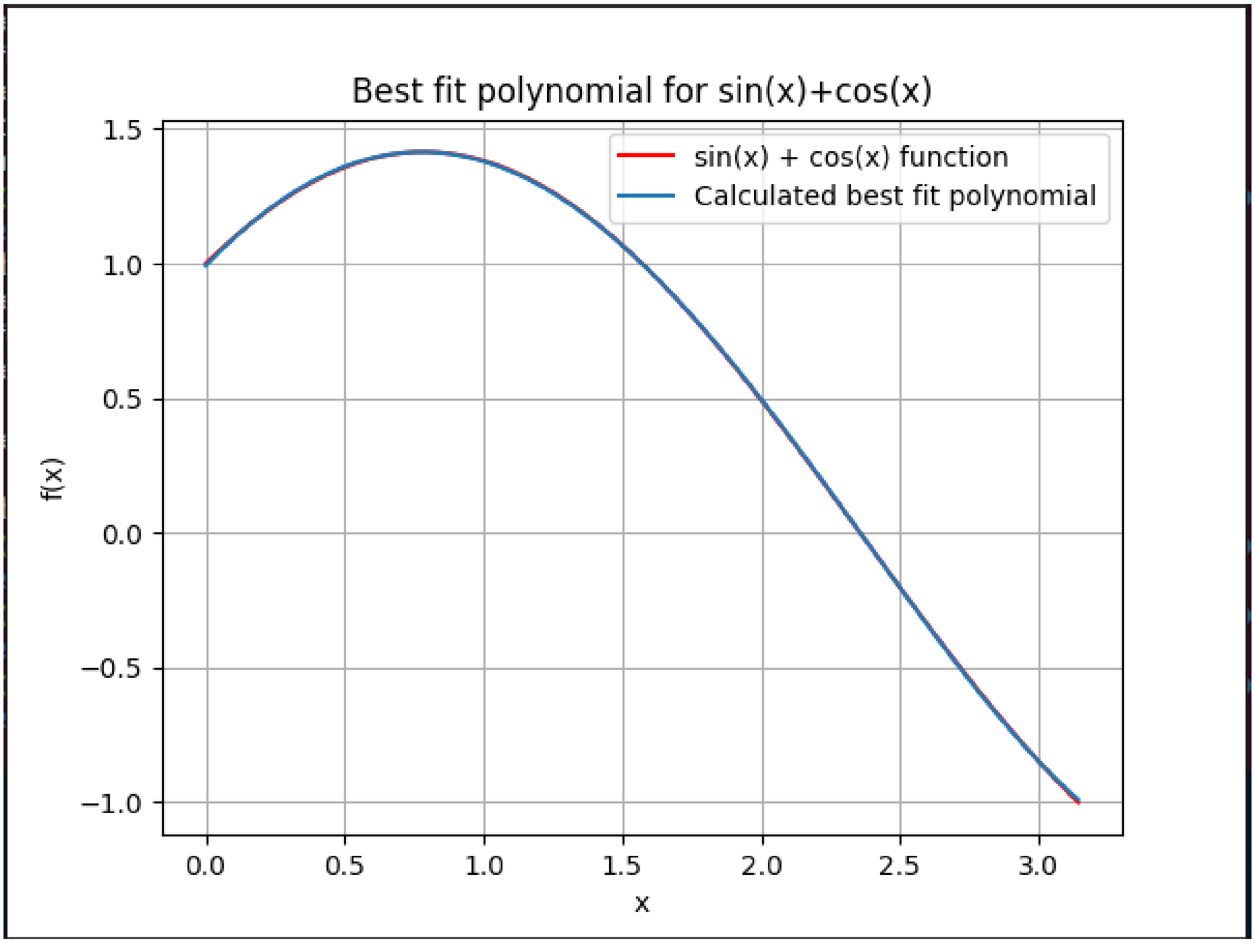
Q1.

So , i created a function fitfunc to calculate the best fit polynomial for the points which takes input points then applies the normal equation them , and then from numpy 's linear algebra solver i find the coefficients of the best fit polynomial .
Then i just plot the polynomial between minimum and maximum of x coordinates of the given points.



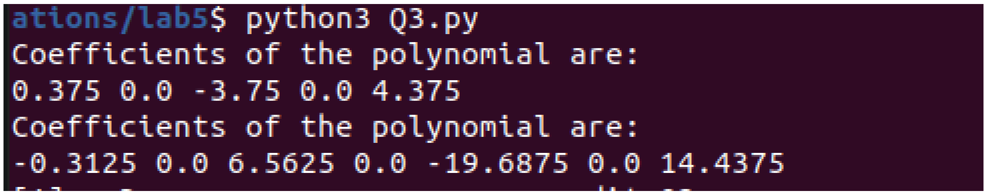
Q2..
The fitfunc from previous question was edited , this time i used scipy's integrate module to calculate the normal equations. Then solved similarly as previous question. Plotted both the best fit polynomial and original curve , to compare.





Q3.
Created nthlegendrepolynomial function class which return the object polynomial containing nth legendre polynomial , the implementation is straightforward application of legendre polynomial formula.

Coefficients of 4 th and 6 th legendre polynomial

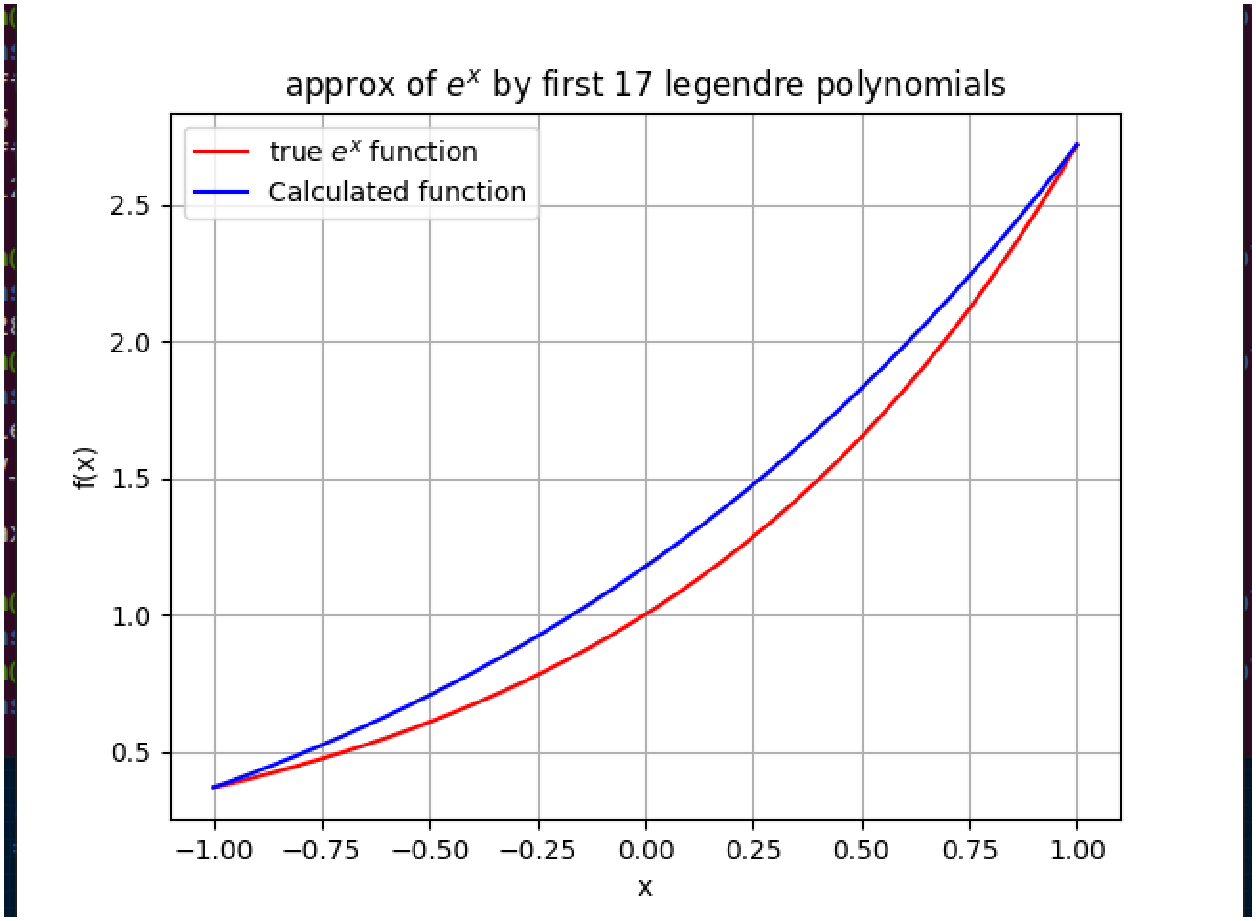


```
ations/lab5$ python3 Q3.py
Coefficients of the polynomial are:
0.375 0.0 -3.75 0.0 4.375
Coefficients of the polynomial are:
-0.3125 0.0 6.5625 0.0 -19.6875 0.0 14.4375
```

Q4.

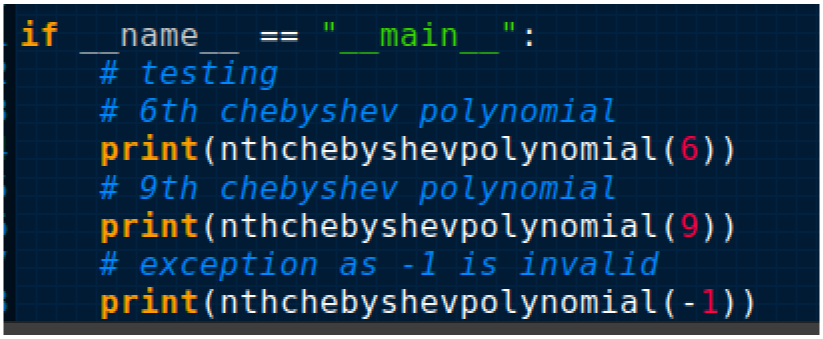
Created function fitusing lagrange to implement least square using legendre polynomial. Implementation involves formula told in lecture pdf .

For the integrations i used scipy’s integrate module.

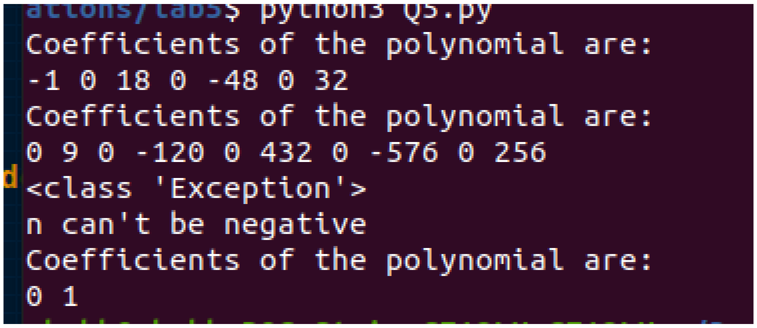


Q5.

Created the class nthchebyshevpolynomial for chebyshev polynomial, applied the formula and recursion mentioned in reference text .

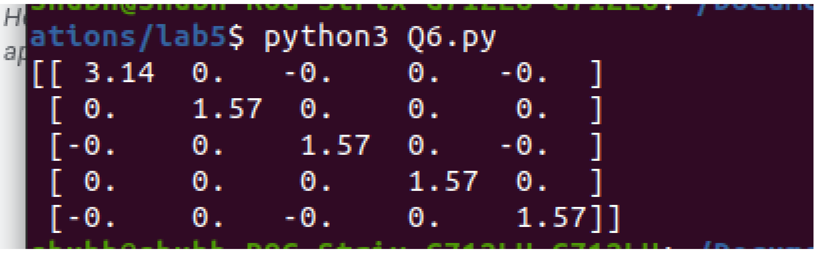


```python
if __name__ == "__main__":
    # testing
    # 6th chebyshev polynomial
    print(nthchebyshevpolynomial(6))
    # 9th chebyshev polynomial
    print(nthchebyshevpolynomial(9))
    # exception as -1 is invalid
    print(nthchebyshevpolynomial(-1))
```



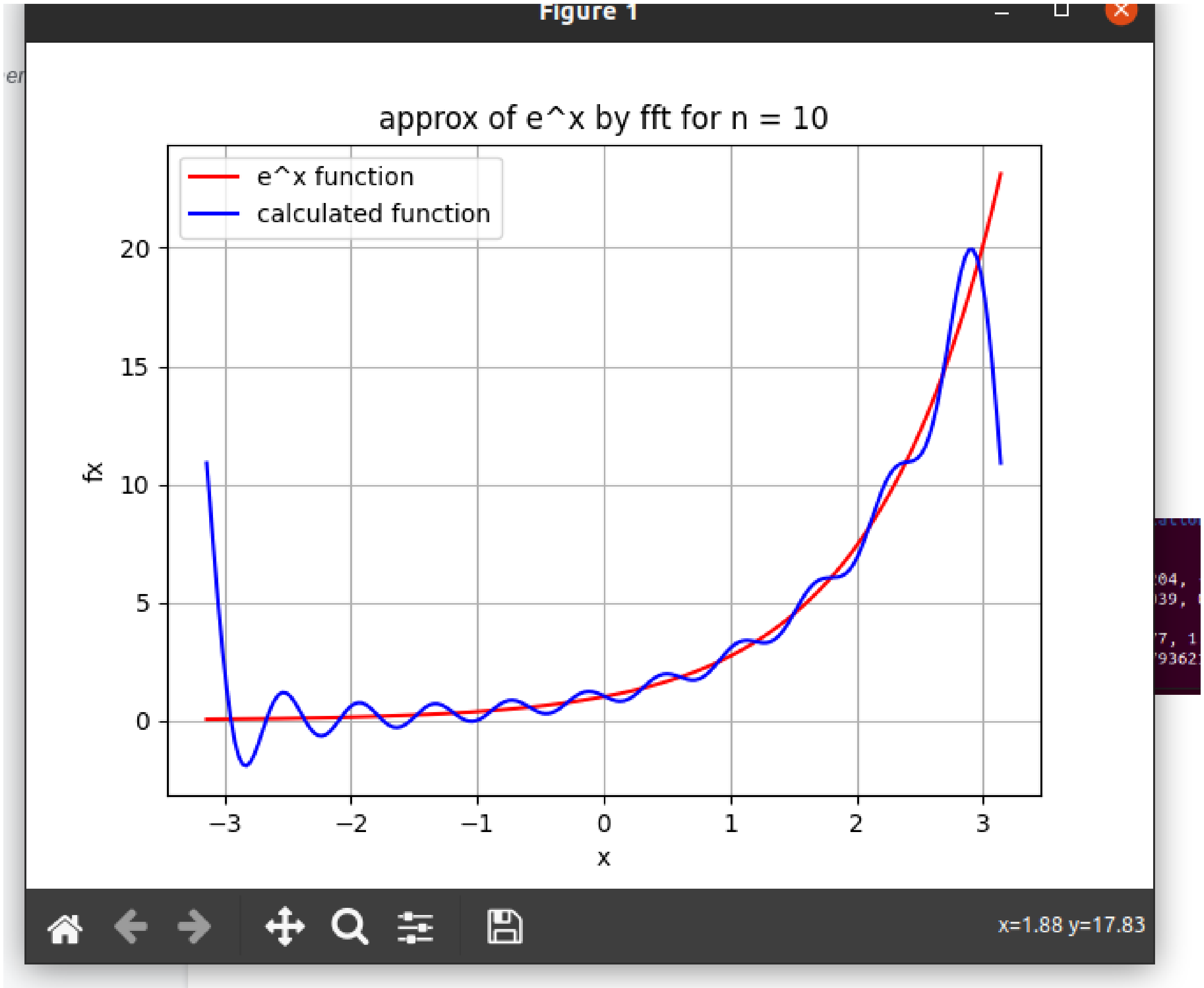
```
ations/lab5$ python3 Q5.py
Coefficients of the polynomial are:
-1 0 18 0 -48 0 32
Coefficients of the polynomial are:
0 9 0 -120 0 432 0 -576 0 256
<class 'Exception'>
n can't be negative
Coefficients of the polynomial are:
0 1
```

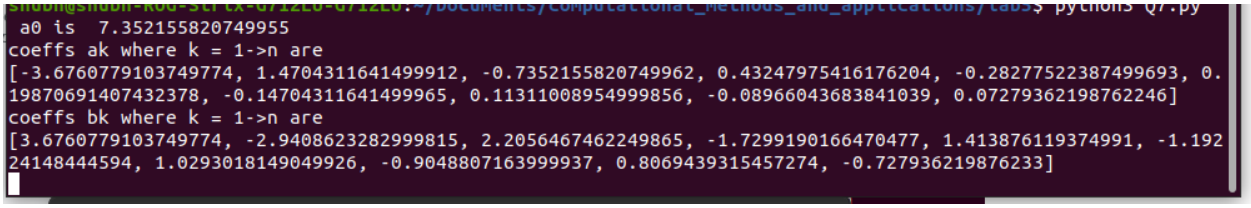
Q6.
Just implemented what was asked and output to show orthogonality.



```
ations/lab5$ python3 Q6.py
[[ 3.14  0.    -0.     0.    -0.  ]
 [ 0.     1.57  0.     0.     0.  ]
 [-0.     0.     1.57  0.    -0.  ]
 [ 0.     0.     0.     1.57  0.  ]
 [-0.     0.    -0.     0.     1.57]]
```

Q7.
Just implemented the formulas mentioned in the referenced text.



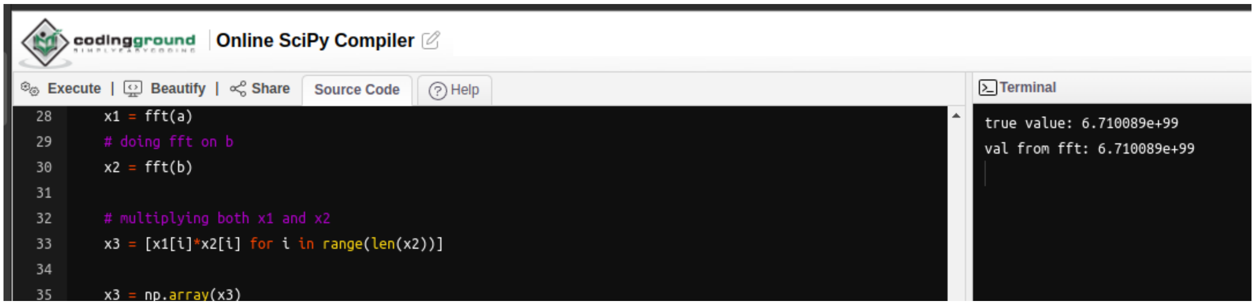


```
shubh@shubh-ROG-Strix-G712LU-G712LU:~/Documents/Computational_Methods_and_applications/labs$ python3 Q7.py
 a0 is  7.352155820749955
coeffs ak where k = 1->n are
[-3.6760779103749774, 1.4704311641499912, -0.7352155820749962, 0.43247975416176204, -0.28277522387499693, 0.
19870691407432378, -0.14704311641499965, 0.11311008954999856, -0.08966043683841039, 0.07279362198762246]
coeffs bk where k = 1->n are
[3.6760779103749774, -2.9408623282999815, 2.2056467462249865, -1.7299190166470477, 1.413876119374991, -1.192
24148444594, 1.0293018149049926, -0.9048807163999937, 0.8069439315457274, -0.727936219876233]
```

Q8

Used fft module for fast fourier transform and ifft for inversing.

Fft was not working inmy computer so used online compiler .



```
    x1 = fft(a)
    # doing fft on b
    x2 = fft(b)

    # multiplying both x1 and x2
    x3 = [x1[i]*x2[i] for i in range(len(x2))]

    x3 = np.array(x3)
```

```
true value: 6.710089e+99
val from fft: 6.710089e+99
```