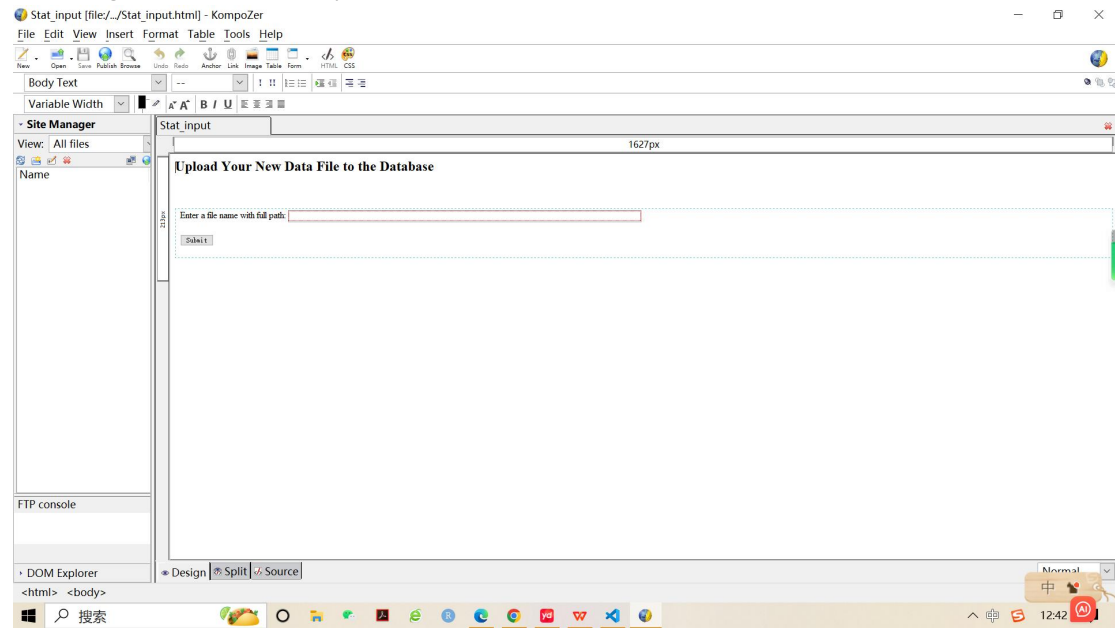
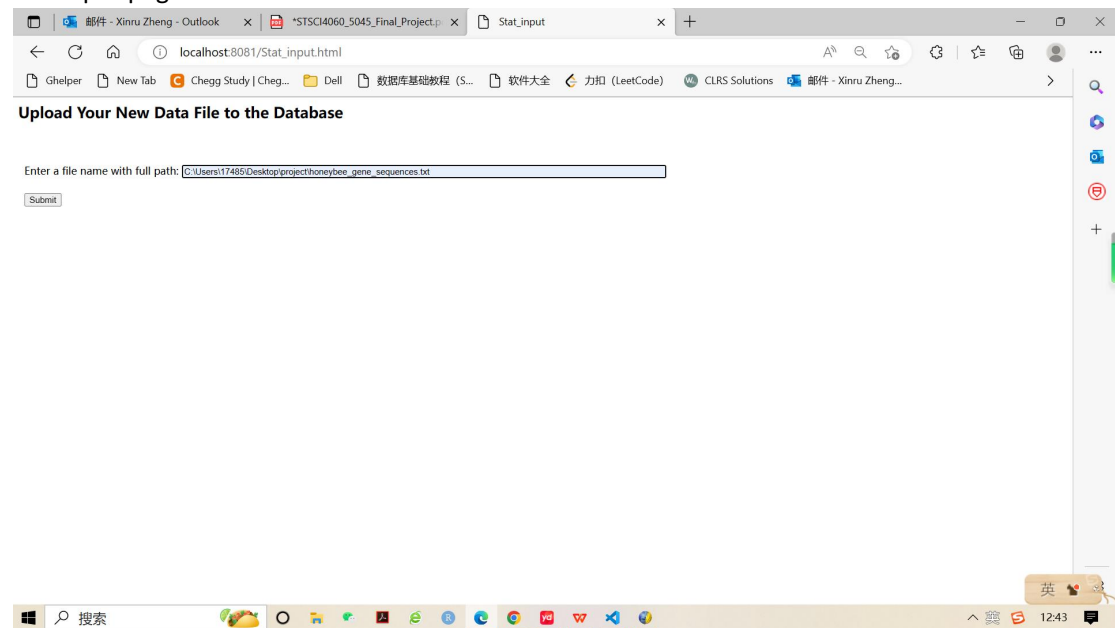


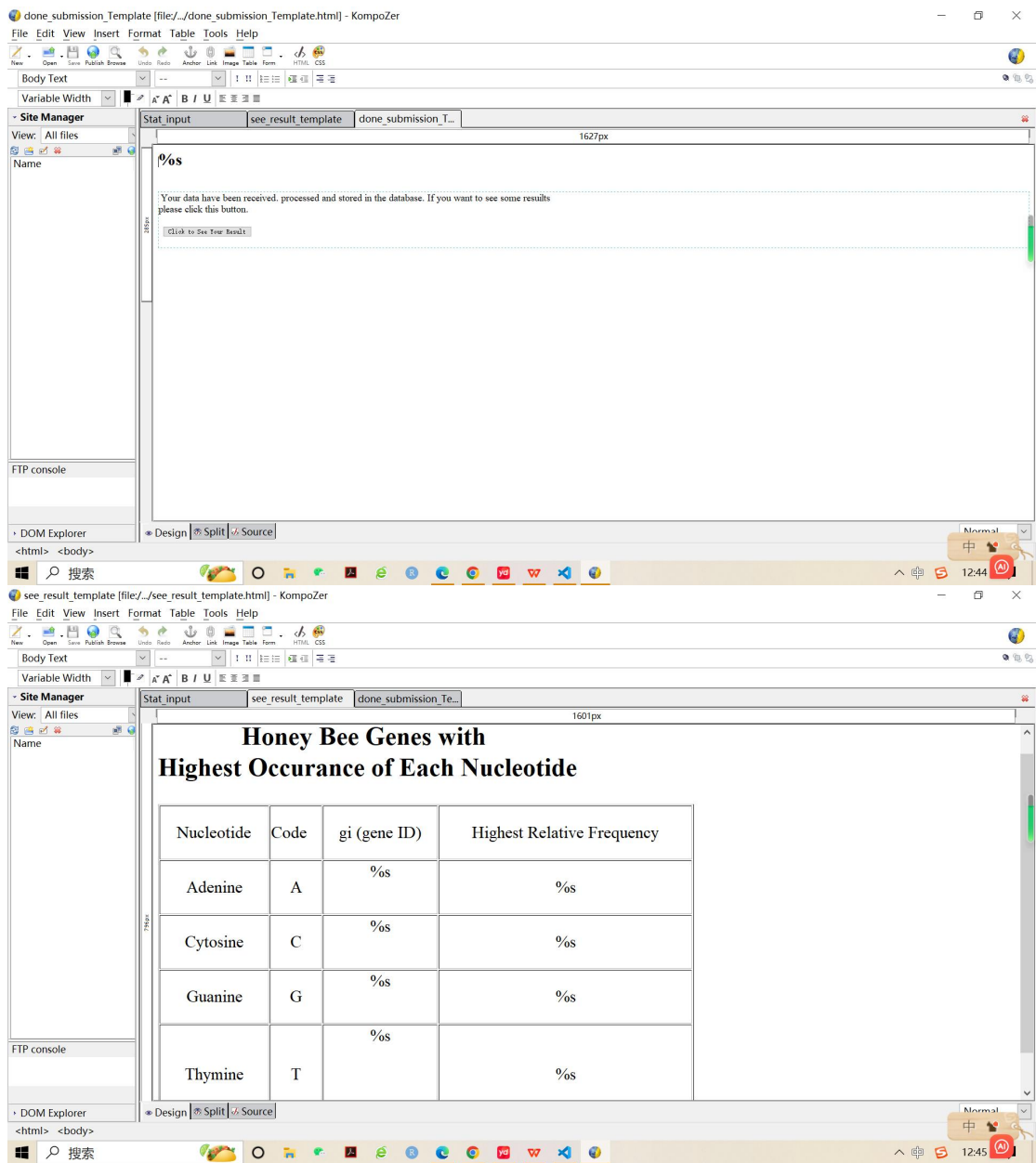
The design window of the input



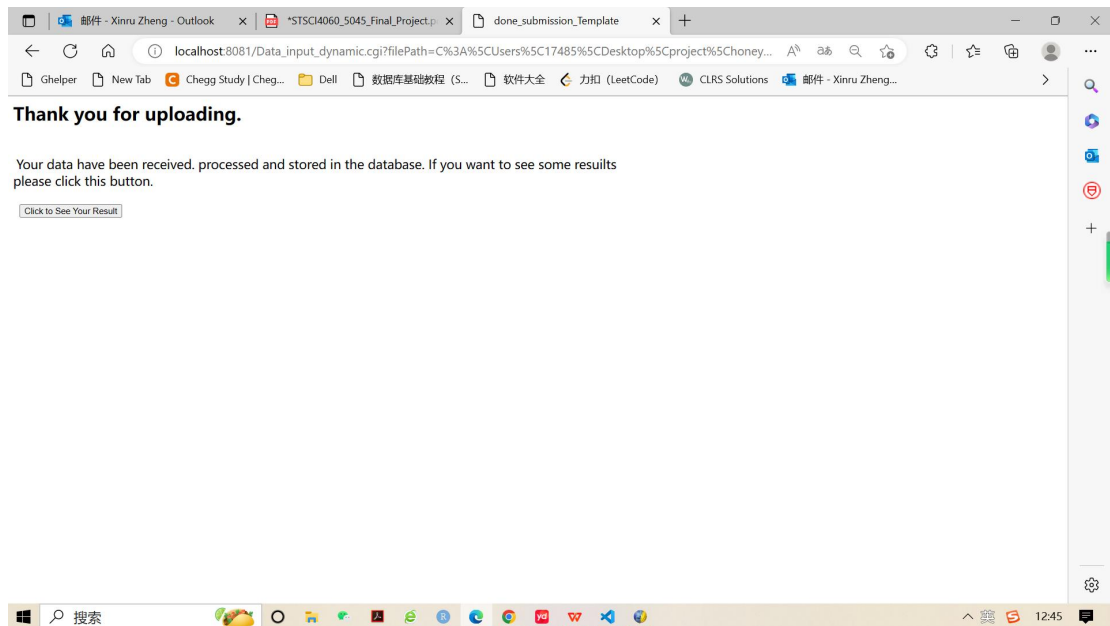
The input page



The design window of template



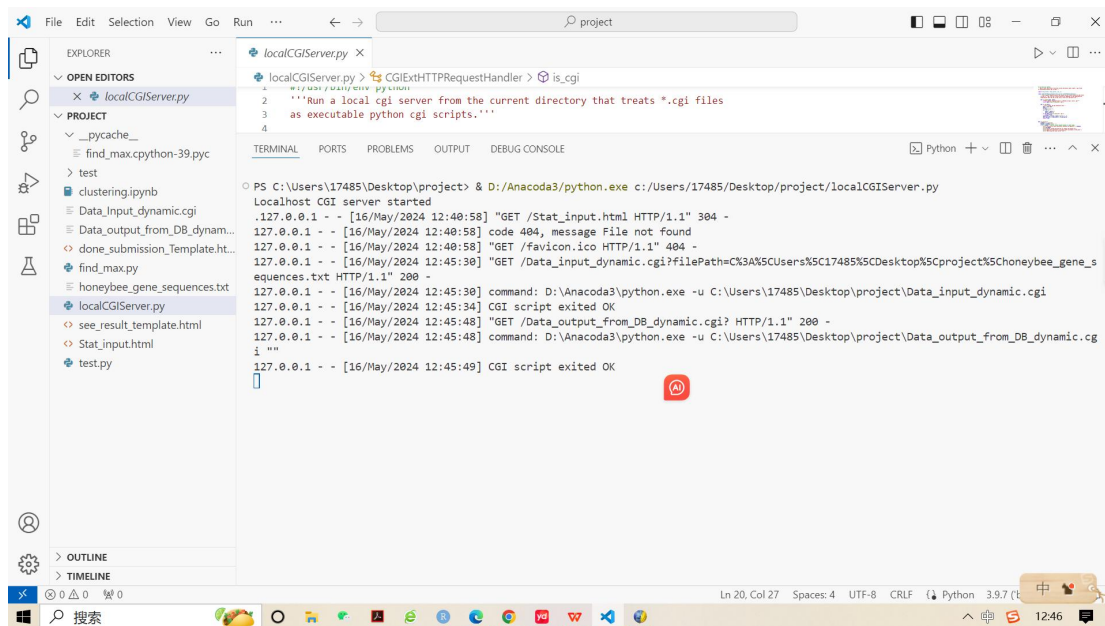
confirmation web page



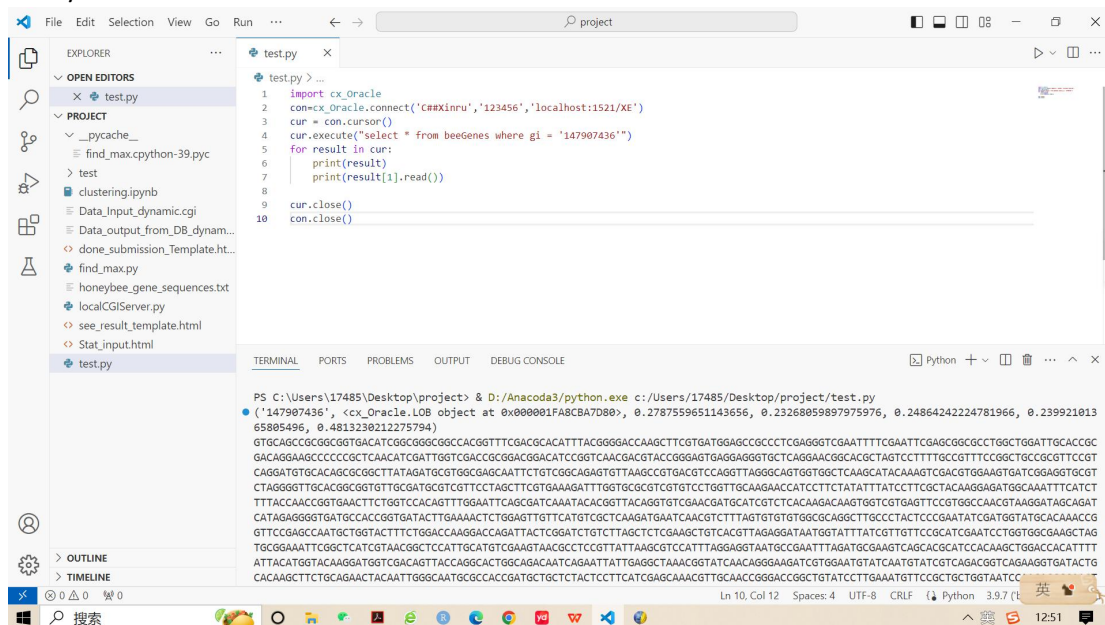
Result

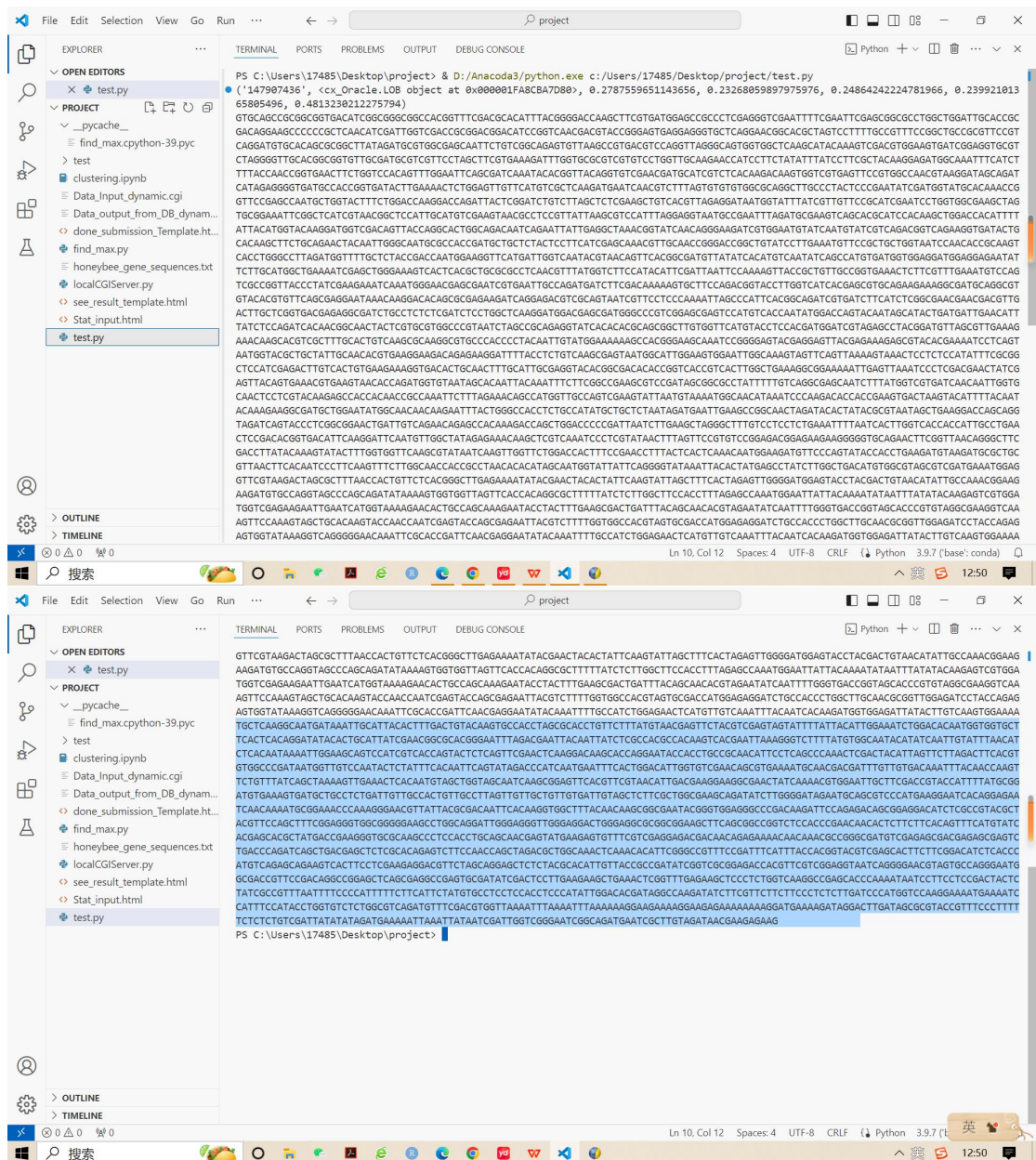
| Nucleotide | Code | gi (gene ID) | Highest Relative Frequency |
|------------|------|---|----------------------------|
| Adenine | A | <ul style="list-style-type: none">197245430 | 0.5141843971631206 |
| Cytosine | C | <ul style="list-style-type: none">292494906 | 0.3838951310861423 |
| Guanine | G | <ul style="list-style-type: none">292494901 | 0.35705045278137126 |
| Thymine | T | <ul style="list-style-type: none">334725196343410563334725204334725205334725203 | 0.4424498416050686 |

Local CGI output



Query result



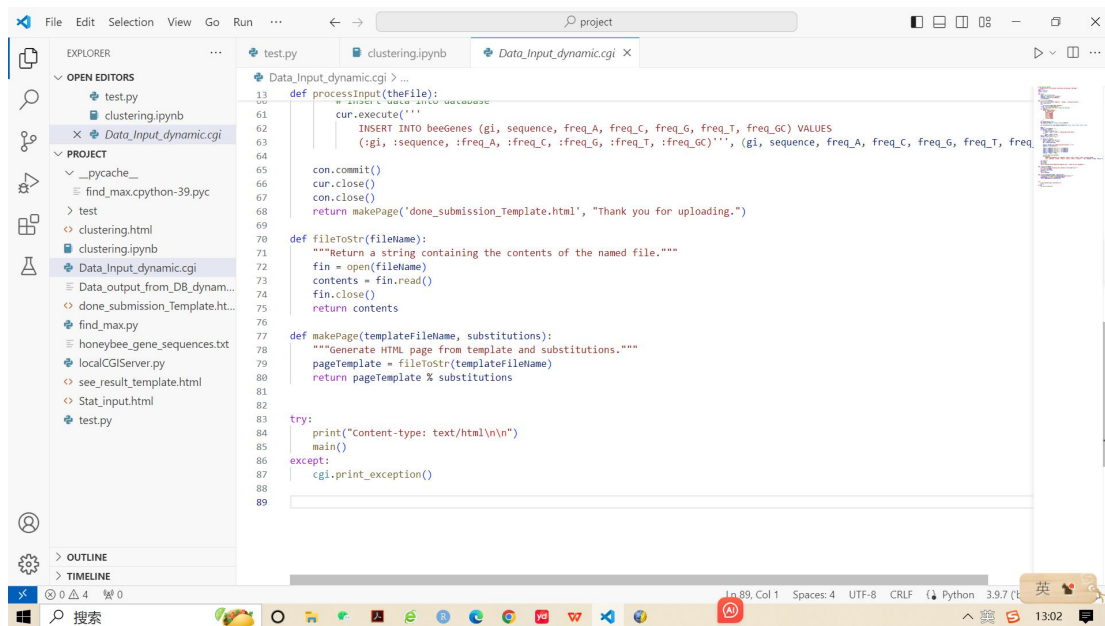


Code:

1. Data INput dynamic.cgi

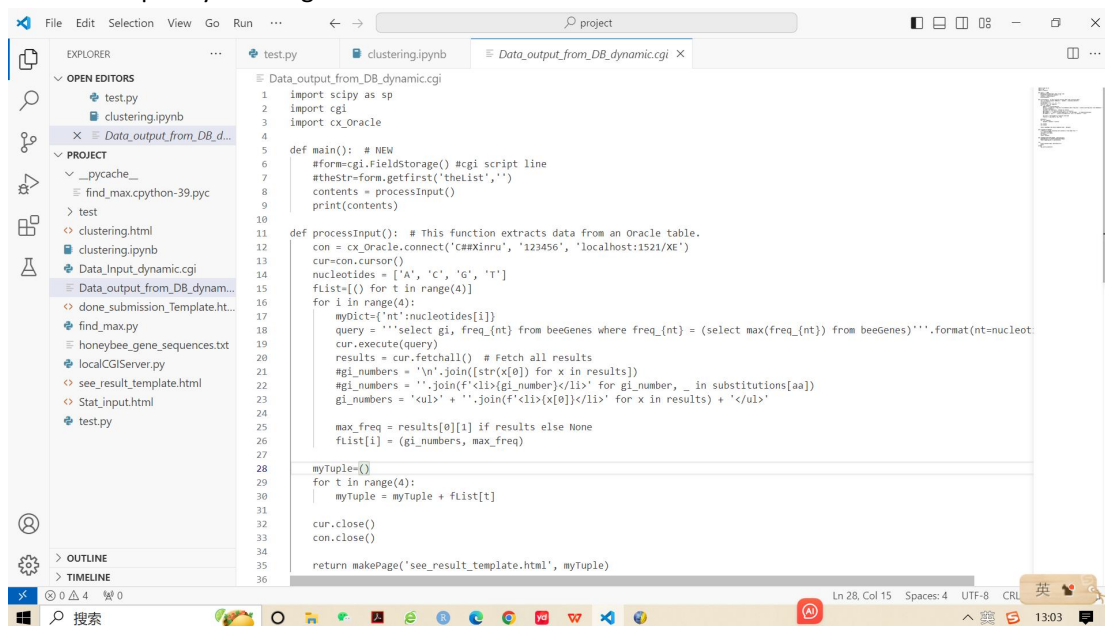

```
1 #!/usr/bin/env python
2 """Prompt the user for personal information and display a web page."""
3 import cgi
4 import cx_Oracle
5 import find_max
6
7 def main():
8     form = cgi.FieldStorage()
9     theFile = form.getFirst('filePath')
10    contents = processInput(theFile)
11    print(contents)
12
13 def processInput(theFile):
14     con = cx_Oracle.connect('c##xinru', '123456', 'localhost:1521/XE')
15     cur = con.cursor()
16     try:
17         cur.execute('DROP TABLE beeGenes')
18     except cx_Oracle.DatabaseError:
19         pass # Assume error due to table not existing
20     cur.execute('''
21         CREATE TABLE beeGenes (
22             g1 VARCHAR2(10),
23             sequence CLOB,
24             freq_A NUMBER,
25             freq_C NUMBER,
26             freq_G NUMBER,
27             freq_T NUMBER,
28             freq_GC NUMBER
29         )
30     ''')
31     cur.bindarraysize = 50
32     max_sequence_length = find_max.find_max(theFile)
33     cur.setinputsizes(10, max_sequence_length+100, float, float, float, float, float)
34
35     myStr = ''
```

```
35
36
37     myStr = ''
38     inFile = open(theFile, 'r')
39     for aline in inFile:
40         if aline.startswith('>'):
41             myStr = myStr + aline + '""gene_seq_starts_here""'
42         else:
43             myStr = myStr + aline
44     strL = myStr.replace('\n', '')
45     records = strL.split('>')[1:]
46
47     for record in records:
48         start = record.find('g1') + 3
49         end = record.find(']', start)
50         g1 = record[start:end]
51
52         start = record.find('""gene_seq_starts_here""') + 25
53         sequence = record[start:]
54
55         freq_A = sequence.count('A') / len(sequence)
56         freq_C = sequence.count('C') / len(sequence)
57         freq_G = sequence.count('G') / len(sequence)
58         freq_T = sequence.count('T') / len(sequence)
59         freq_GC = freq_G + freq_C
60
61         # Insert data into database
62         cur.execute('''
63             INSERT INTO beeGenes (g1, sequence, freq_A, freq_C, freq_G, freq_T, freq_GC) VALUES
64             (:g1, :sequence, :freq_A, :freq_C, :freq_G, :freq_T, :freq_GC)''', (g1, sequence, freq_A, freq_C, freq_G, freq_T, freq_GC))
65
66     con.commit()
67     cur.close()
68     con.close()
69     return makePage('done_submission_Template.html', "Thank you for uploading.")
```

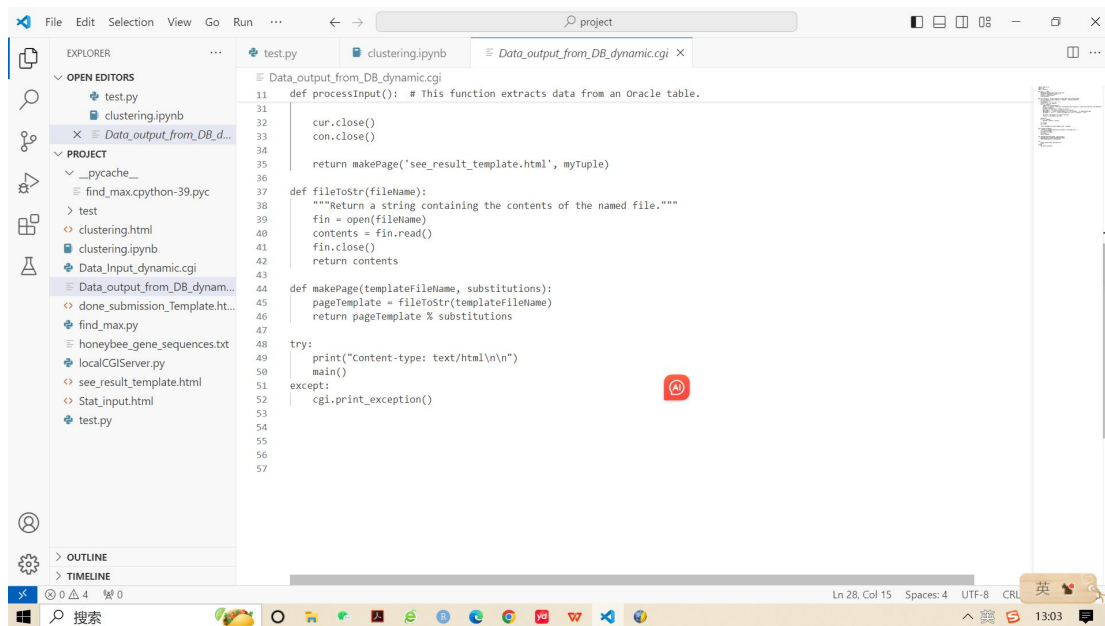


```
13 def processInput(theFile):
14     cur.execute('INSERT INTO beegenes (gi, sequence, freq_A, freq_C, freq_G, freq_T, freq_GC) VALUES
15         (:gi, :sequence, :freq_A, :freq_C, :freq_G, :freq_T, :freq_GC)', (gi, sequence, freq_A, freq_C, freq_G, freq_T, freq
61
62     cur.execute('INSERT INTO beegenes (gi, sequence, freq_A, freq_C, freq_G, freq_T, freq_GC) VALUES
63         (:gi, :sequence, :freq_A, :freq_C, :freq_G, :freq_T, :freq_GC)', (gi, sequence, freq_A, freq_C, freq_G, freq_T, freq
64
65     con.commit()
66     cur.close()
67     con.close()
68     return makePage('done_submission_Template.html', "Thank you for uploading.")
69
70 def fileToStr(fileName):
71     """Return a string containing the contents of the named file."""
72     fin = open(fileName)
73     contents = fin.read()
74     fin.close()
75     return contents
76
77 def makePage(templateFileName, substitutions):
78     """Generate HTML page from template and substitutions."""
79     pageTemplate = fileToStr(templateFileName)
80     return pageTemplate % substitutions
81
82
83 try:
84     print("Content-type: text/html\n\n")
85     main()
86 except:
87     cgi.print_exception()
88
89
```

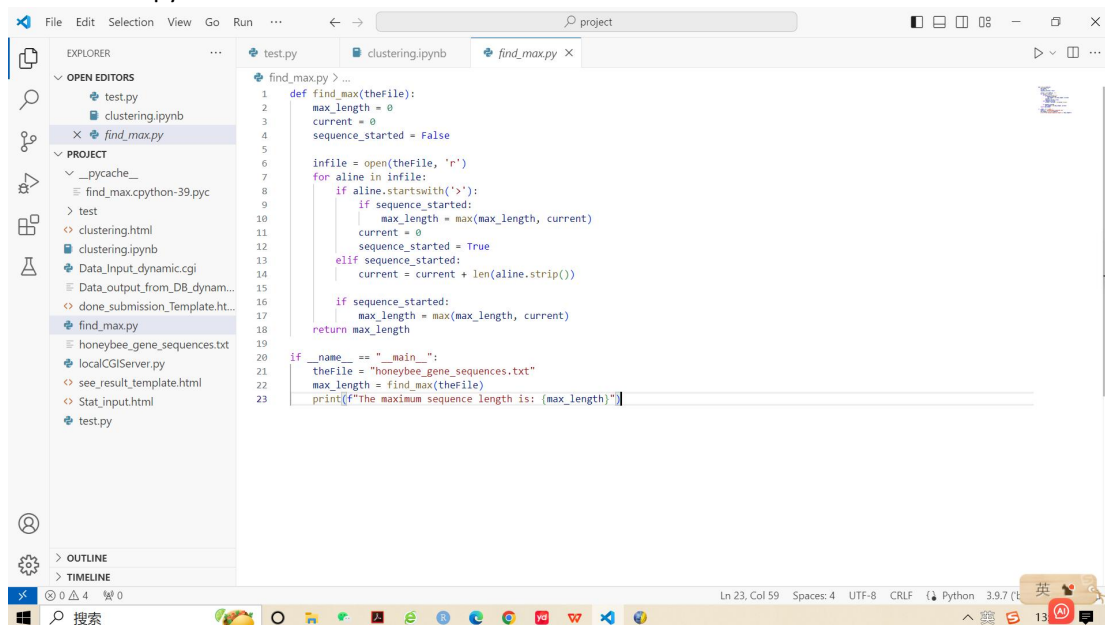
2. Data output dynamic.cgi



```
1 import cgi
2 import sys
3 import cx_Oracle
4
5 def main(): # NEW
6     #form=cgi.FieldStorage() #cgi script line
7     #theStr=form.getvalue('theList','')
8     contents = processInput()
9     print(contents)
10
11 def processInput(): # This function extracts data from an Oracle table.
12     con = cx_Oracle.connect('C##Xinru', '123456', 'localhost:1521/XE')
13     cur=con.cursor()
14     nucleotides = ['A', 'C', 'G', 'T']
15     flist=[] for t in range(4):
16         for i in range(4):
17             myDict={'nt':nucleotides[i]}
18             query = 'select gi, freq_(nt) from beegenes where freq_(nt) = (select max(freq_(nt)) from beegenes)'.format(nt=nucleot
19             cur.execute(query)
20             results = cur.fetchall() # Fetch all results
21             #gi_numbers = '\n'.join([str(x[0]) for x in results])
22             #gi_numbers = '\n'.join(f'<li>{gi_number}</li>' for gi_number, _ in substitutions[aa])
23             gi_numbers = '<ul>' + '\n'.join(f'<li>{x[0]}</li>' for x in results) + '</ul>'
24
25             max_freq = results[0][1] if results else None
26             flist[i] = (gi_numbers, max_freq)
27
28     myTuple=()
29     for t in range(4):
30         myTuple = myTuple + flist[t]
31
32     cur.close()
33     con.close()
34
35     return makePage('see_result_template.html', myTuple)
36
```



3. Find max.py



4. clustering.ipynb

Visual Studio Code interface showing a Jupyter Notebook (clustering.ipynb) with Python code for connecting to an Oracle database, fetching data, and performing K-Means clustering.

Top Screenshot:

- Explorer:** test.py, clustering.ipynb, Data_Input_dynamic.cgi, Data_Output_dynamic.cgi, done_submission_Template.html, find_max.py, honeybee_gene_sequences.txt, localCGIServer.py, see_result_template.html, Stat_input.html, test.py.
- Code Editor:** clustering.ipynb > ...
 - Cell [1]:

```
import cx_Oracle
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```
 - Cell [2]:

```
# Connect to the Oracle database
connection = cx_Oracle.connect('c##Xinru', '123456', 'localhost:1521/XE')
cursor = connection.cursor()

# Query to fetch the data
query = '''
SELECT freq_A, freq_T, freq_GC
FROM beeGenes
'''

# Execute the query and fetch all data
cursor.execute(query)
data = cursor.fetchall()

# Close cursor and connection
cursor.close()
connection.close()

# Convert the data to a NumPy array
data_array = np.array(data, dtype=np.float32)
```

Bottom Screenshot:

- Code Editor:** clustering.ipynb > ...
 - Cell [3]:

```
# K-Means settings
kmeans = KMeans(n_clusters=7, init='random', n_init=10, max_iter=500, tol=1e-4, random_state=0)
# Fit K-Means model
kmeans.fit(data_array)

# Plotting
fig = plt.figure(figsize=(14, 14))
ax = fig.add_subplot(111, projection='3d')

colors = ['red', 'blue', 'aqua', 'black', 'purple', 'magenta', 'green']
clusters = kmeans.labels_
centroids = kmeans.cluster_centers_

# Scatter plot for each cluster
for i in range(7):
    ax.scatter(data_array[clusters == i, 0], data_array[clusters == i, 1], data_array[clusters == i, 2],
              c=colors[i], label=f'Cluster {i+1}', s=20, marker='o')

ax.scatter(centroids[:, 0], centroids[:, 1], centroids[:, 2], c='red', s=100, marker='*', label='Centroids')

# Labels and titles
ax.set_xlabel('Freq_A')
ax.set_ylabel('Freq_T')
ax.set_zlabel('Freq_GC')
ax.set_title('K-Means Clustering of BeeGenes')

# Legend
ax.legend(loc='upper right')

plt.show()
```

