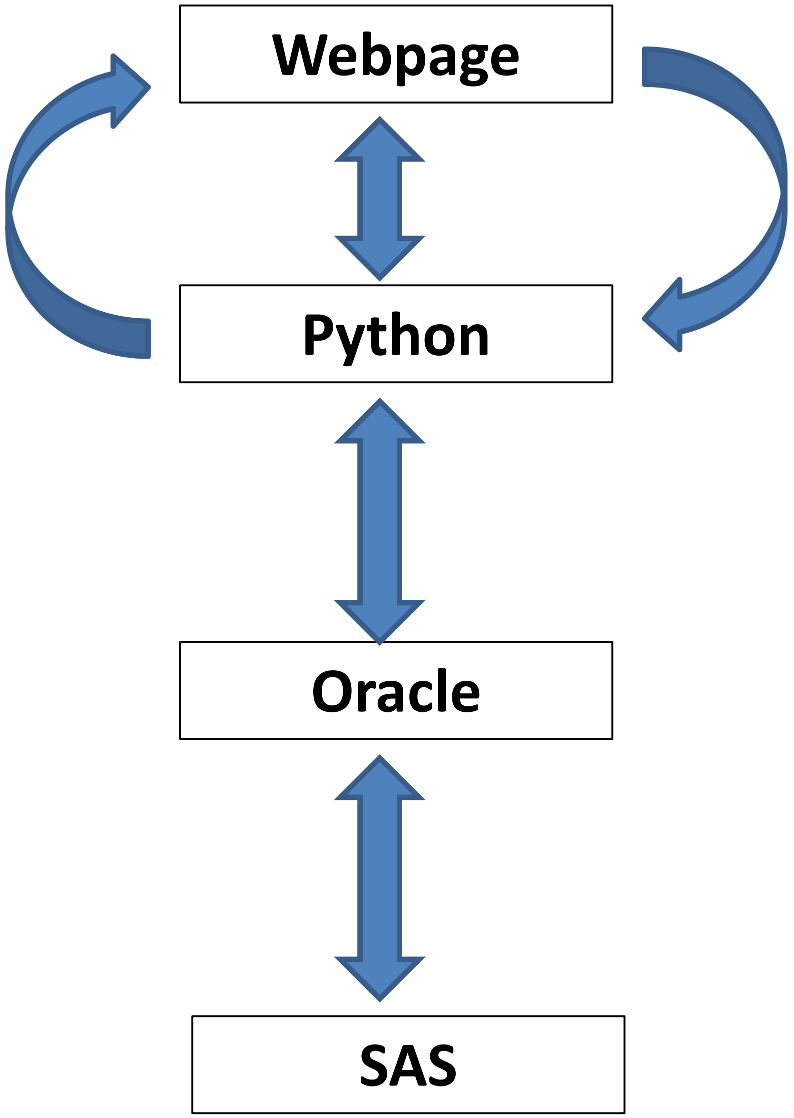
STSCI 4060

Lecture File 11

Xiaolong Yang
(xy44@cornell.edu)

Python, Database-Driven Web Pages, and SAS

Python, Oracle, Webpage and SAS Integration



Prepare/Gather Data of Interest

Where to get data?

- Data that people already collected and managed in a format ready for use
- Data that have been collected but need treatment
- Data that are somewhat all over different places, e.g., internet
- Data to be collected by you or someone through various experiments

Where and How to Store your Data?

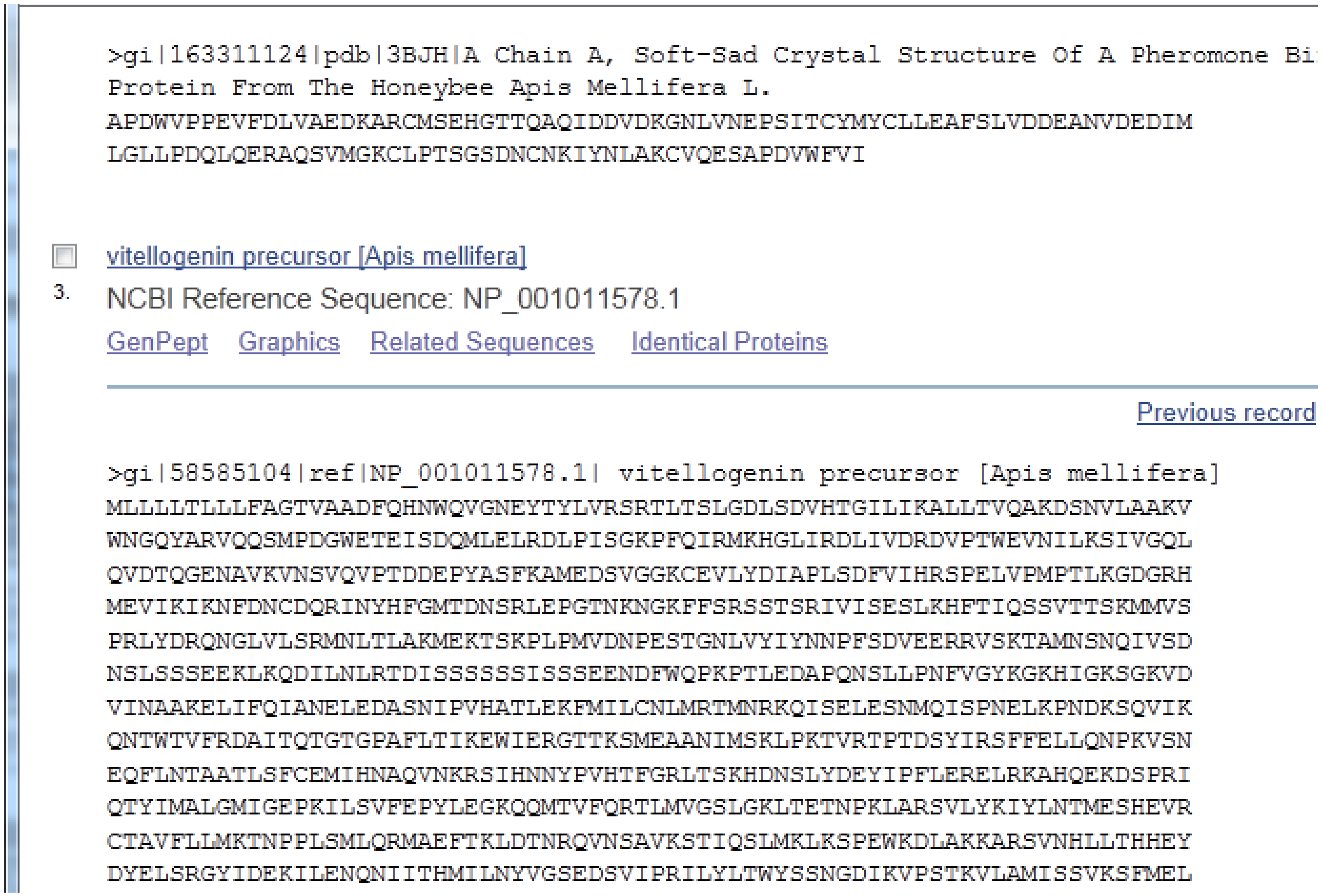
- Your personal media, e.g., disks, flash drives, etc.
- Your personal computer (flat files, MS Excel files, etc.)
- Databases of different levels
    - Local databases
    - Global databases connected via Internet

Our Scenario

- Collect data of interest from internet
- Study the data collected
- Treat/transform the data with Python programs
- Store the transformed data in an Oracle database through a network using Python
- Access the database through a network using Python, which is an example of database-driven, Python-powered dynamic webpage
- Analyze the data stored in the database with SAS, etc.

Gather Our Data of Interest

- Honey bee proteins that have been discovered and studied.
- Available from the Internet, especially from NCBI (http://www.ncbi.nlm.nih.gov/) , for example



```
>gi|163311124|pdb|3BJH|A Chain A, Soft-Sad Crystal Structure Of A Pheromone Bi
Protein From The Honeybee Apis Mellifera L.
APDWVPPEVFDLVAEDKARCMSEHGTTQAQIDDVDKGNLVNEPSITCYMYCLLEAFSLVDDEANVDEDIM
LGLLPDQLQERAQSVMGKCLPTSGSDNCNKIYNLAKCVQESAPDVWFVI
```

vitellogenin precursor [Apis mellifera]

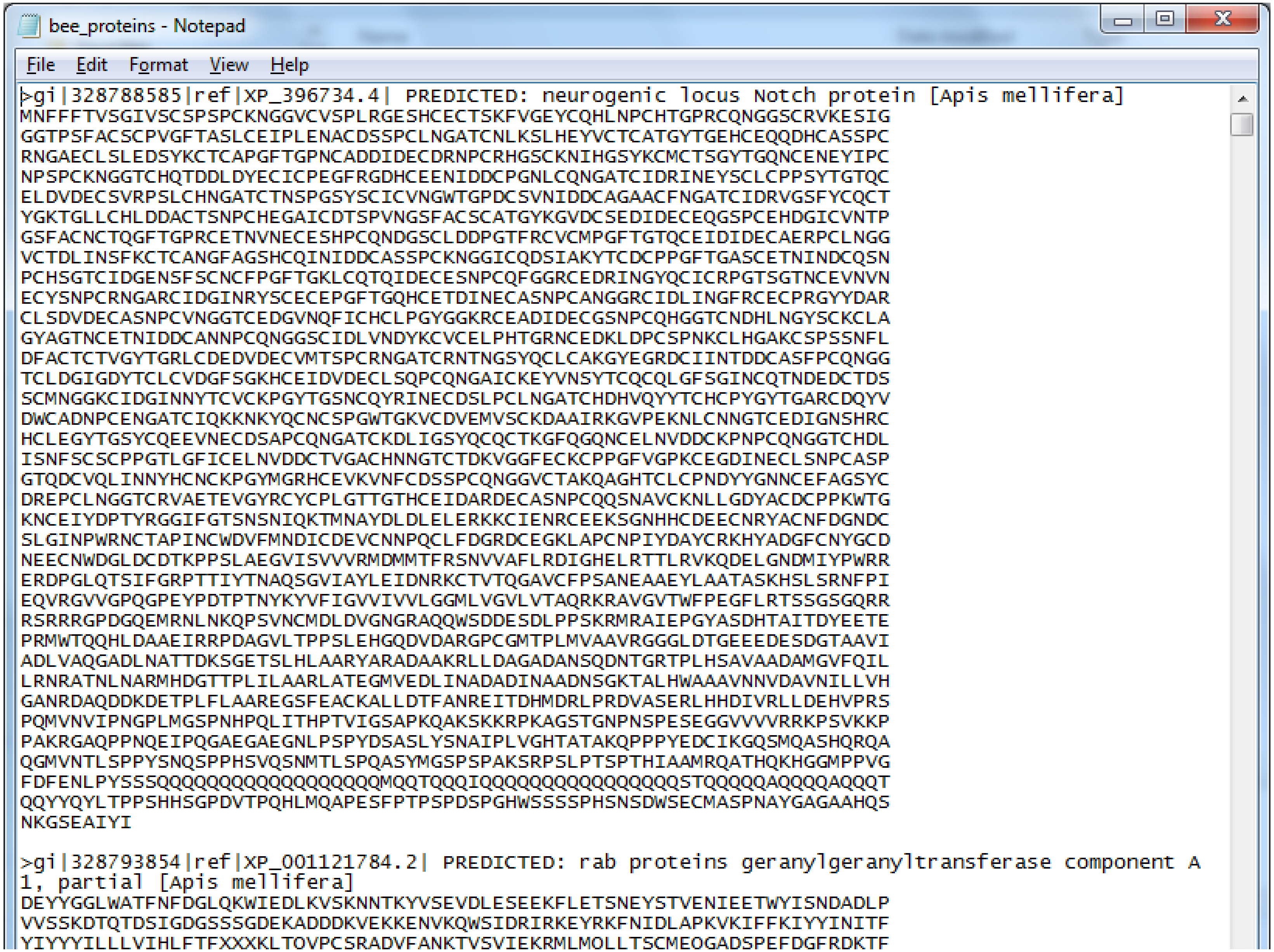
3. NCBI Reference Sequence: NP_001011578.1

GenPept Graphics Related Sequences Identical Proteins

Previous record

```
>gi|58585104|ref|NP_001011578.1| vitellogenin precursor [Apis mellifera]
MLLLLTLLLFAGTVAADFQHNWQVGNEYTYLVRSRTLTSLGDLSDVHTGILIKALLTVQAKDSNVLAAKV
WNGQYARVQQSMPDGWETEISDQMLELRDLPISGKPFQIRMKHGLIRDLIVDRDVPTWEVNILKSIVGQL
QVDTQGENAVKVNSVQVPTDDEPYASFKAMEDSVGGKCEVLYDIAPLSDFVIHRSPELVPMPTLKGDGRH
MEVIKIKNFDNCDQRINYHFGMTDNSRLEPGTNKNGKFFSRSSTSRIVISESLKHFTIQSSVTTSKMMVS
PRLYDRQNGLVLSRMNLTLAKMEKTSKPLPMVDNPESTGNLVYIYNNPFSDVEERRVSKTAMNSNQIVSD
NSLSSSEEKLKQDILNLRTDISSSSSSISSSEENDFWQPKPTLEDAPQNSLLPNFVGYKGKHIGKSGKVD
VINAAKELIFQIANELEDASNIPVHATLEKFMILCNLMRTMNRKQISELESNMQISPNELKPNDKSQVIK
QNTWTVFRDAITQTGTGPAFLTIKEWIERGTTKSMEAANIMSKLPKTVRTPTDSYIRSFFELLQNPKVSN
EQFLNTAATLSFCEMIHNAQVNKRSIHNNYPVHTFGRLTSKHDNSLYDEYIPFLERELRKAHQEKDSPRI
QTYIMALGMIGEPKILSVFEPYLEGKQQMTVFQRTLMVGSLGKLTETNPKLARSVLYKIYLNTMESHEVR
CTAVFLLMKTNPPLSMLQRMAEFTKLDTNRQVNSAVKSTIQSLMKLKSPEWKDLAKKARSVNHLLTHHEY
DYELSRGYIDEKILENQNIITHMILNYVGSEDSVIPRILYLTWYSSNGDIKVPSTKVLAMISSVKSFMEL
```

The Honey Bee Protein Data Set



```
>gi|328788585|ref|XP_396734.4| PREDICTED: neurogenic locus Notch protein [Apis mellifera]
MNFFFTVSGIVSCSPSPCKNGGVCVSPLRGESHCECTSKFVGEYCQHLNPCHTGPRCQNGGSCRVKESIG
GGTPSFACSCPVGFTASLCEIPLENACDSSPCLNGATCNLKSLHEYVCTCATGYTGEHCEQQDHCASSPC
RNGAECLSLEDSYKCTCAPGFTGPNCADDIDECDRNPCRHGSCKNIHGSYKCMCTSGYTGQNCENEYIPC
NPSPCKNGGTCHQTDDLDYECICPEGFRGDHCEENIDDCPGNLCQNGATCIDRINEYSCLCPPSYTGTQC
ELDVDECSVRPSLCHNGATCTNSPGSYSCICVNGWTGPDCSVNIDDCAGAACFNGATCIDRVGSFYCQCT
YGKTGLLCHLDDACTSNPCHEGAICDTSPVNGSFACSCATGYKGVDCSEDIDECEQGSPCEHDGICVNTP
GSFACNCTQGFTGPRCETNVNECESHPCQNDGSCLDDPGTFRCVCMPGFTGTQCEIDIDECAERPCLNGG
VCTDLINSFKCTCANGFAGSHCQINIDDCASSPCKNGGICQDSIAKYTCDCPPGFTGASCETNINDCQSN
PCHSGTCIDGENSFSCNCFPGFTGKLCQTQIDECESNPCQFGGRCEDRINGYQCICRPGTSGTNCEVNVN
ECYSNPCRNGARCIDGINRYSCECEPGFTGQHCETDINECASNPCANGGRCIDLINGFRCECPRGYYDAR
CLSDVDECASNPCVNGGTCEDGVNQFICHCLPGYGGKRCEADIDECGSNPCQHGGTCNDHLNGYSCKCLA
GYAGTNCETNIDDCANNPCQNGGSCIDLVNDYKCVCELPHTGRNCEDKLDPCSPNKCLHGAKCSPSSNFL
DFACTCTVGYTGRLCDEDVDECVMTSPCRNGATCRNTNGSYQCLCAKGYEGRDCIINTDDCASFPCQNGG
TCLDGIGDYTCLCVDGFSGKHCEIDVDECLSQPCQNGAICKEYVNSYTCQCQLGFSGINCQTNDEDCTDS
SCMNGGKCIDGINNYTCVCKPGYTGSNCQYRINECDSLPCLNGATCHDHVQYYTCHCPYGYTGARCDQYV
DWCADNPCENGATCIQKKNKYQCNCSPGWTGKVCDVEMVSCKDAAIRKGVPEKNLCNNGTCEDIGNSHRC
HCLEGYTGSYCQEEVNECDSAPCQNGATCKDLIGSYQCQCTKGFQGQNCELNVDDCKPNPCQNGGTCHDL
ISNFSCSCPPGTLGFICELNVDDCTVGACHNNGTCTDKVGGFECKCPPGFVGPKCEGDINECLSNPCASP
GTQDCVQLINNYHCNCKPGYMGRHCEVKVNFCDSSPCQNGGVCTAKQAGHTCLCPNDYYGNNCEFAGSYC
DREPCLNGGTCRVAETEVGYRCYCPLGTTGTHCEIDARDECASNPCQQSNAVCKNLLGDYACDCPPKWTG
KNCEIYDPTYRGGIFGTSNSNIQKTMNAYDLDLELERKKCIENRCEEKSGNHHCDEECNRYACNFDGNDC
SLGINPWRNCTAPINCWDVFMNDICDEVCNNPQCLFDGRDCEGKLAPCNPIYDAYCRKHYADGFCNYGCD
NEECNWDGLDCDTKPPSLAEGVISVVVRMDMMTFRSNVVAFLRDIGHELRTTLRVKQDELGNDMIYPWRR
ERDPGLQTSIFGRPTTIYTNAQSGVIAYLEIDNRKCTVTQGAVCFPSANEAAEYLAATASKHSLSRNFPI
EQVRGVVGPQGPEYPDTPTNYKYVFIGVVIVVLGGMLVGVLVTAQRKRAVGVTWFPEGFLRTSSGSGQRR
RSRRRGPDGQEMRNLNKQPSVNCMDLDVGNGRAQQWSDDESDLPPSKRMRAIEPGYASDHTAITDYEETE
PRMWTQQHLDAAEIRRPDAGVLTPPSLEHGQDVDARGPCGMTPLMVAAVRGGGLDTGEEEDESDGTAAVI
ADLVAQGADLNATTDKSGETSLHLAARYARADAAKRLLDAGADANSQDNTGRTPLHSAVAADAMGVFQIL
LRNRATNLNARMHDGTTPLILAARLATEGMVEDLINADADINAADNSGKTALHWAAAVNNVDAVNILLVH
GANRDAQDDKDETPLFLAAREGSFEACKALLDTFANREITDHMDRLPRDVASERLHHDIVRLLDEHVPRS
PQMVNVIPNGPLMGSPNHPQLITHPTVIGSAPKQAKSKKRPKAGSTGNPNSPESEGGVVVVRRKPSVKKP
PAKRGAQPPNQEIPQGAEGAEGNLPSPYDSASLYSNAIPLVGHTATAKQPPPYEDCIKGQSMQASHQRQA
QGMVNTLSPPYSNQSPPHSVQSNMTLSPQASYMGSPSPAKSRPSLPTSPTHIAAMRQATHQKHGGMPPVG
FDFENLPYSSSQQQQQQQQQQQQQQQQMQQTQQQIQQQQQQQQSTQQQQQAQQQQAQQQT
QQYYQYLTPPSHHSGPDVTPQHLMQAPESFPTPSPDSPGHWSSSSPHSNSDWSECMASPNAYGAGAAHQS
NKGSEAIYI

>gi|328793854|ref|XP_001121784.2| PREDICTED: rab proteins geranylgeranyltransferase component A
1, partial [Apis mellifera]
DEYYGGLWATFNFDGLQKWIEDLKVSKNNTKYVSEVDLESEEKFLETSNEYSTVENIEETWYISNDADLP
VVSSKDTQTDSIGDGSSSGDEKADDDKVEKKENVKQWSIDRIRKEYRKFNIDLAPKVKIFFKIYYINITF
YIYYYILLLVIHLFTFXXXKLTQVPCSRADVFANKTVSVIEKRMLMQLLTSCMEQGADSPEFDGFRDKTF
```

Inspect The Honey Bee Protein Data Set

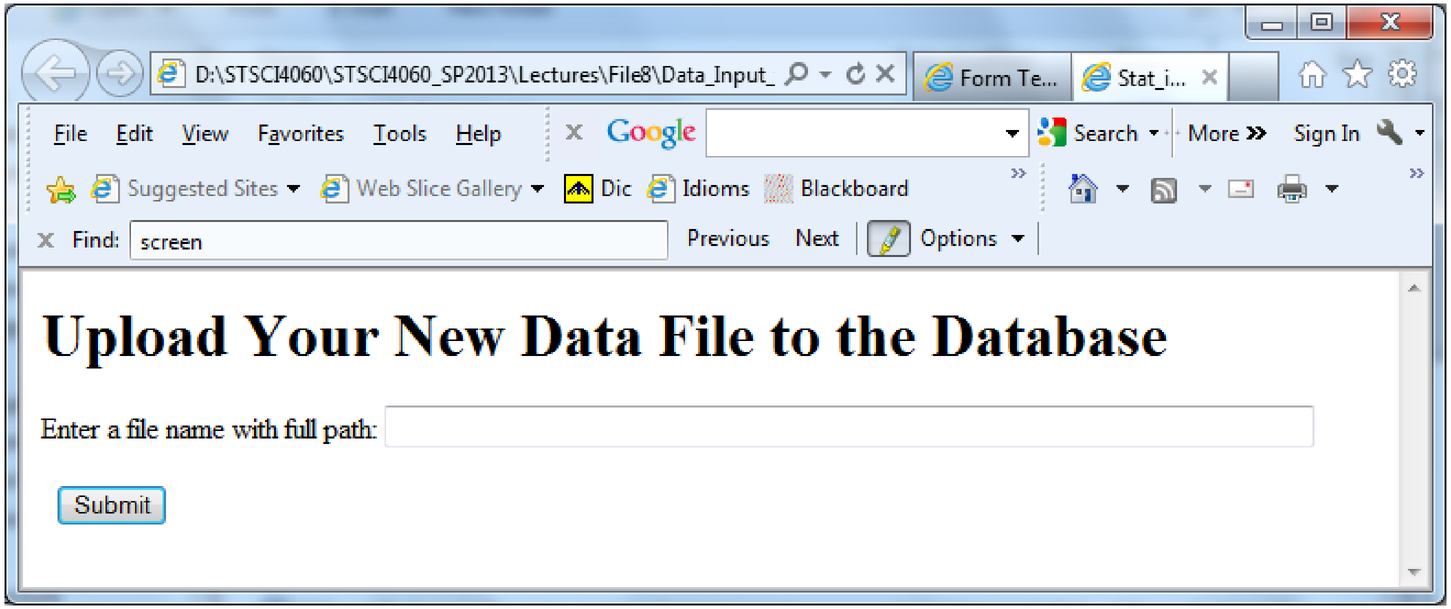
- An entry (protein) always starts with "**>gi|**" (in our case)
- It starts with a short description about the protein, but the length varies
- There is a substring "**[*Apis mellifera*]** " right before the actual protein sequence
- The sequence has a **carriage return** at the end of each line of the sequence
- Our interest:
  - * Only interested in one species of honey bees, <u>*Apis mellifera*</u>
  - * The <u>gi number</u> (which can be used as a protein ID)
  - * The continuous <u>protein sequence</u>
  - * The <u>relative frequency</u> of the amino acids in each protein

Our Plan to Treat the Data Set

- Use Python as the programming tool
- Submit this data file to the database (Oracle) through a specific web page
- Read in the content as a string
- Change content into a continuous string
- Split the string into a list of bee proteins that only belong to *Apis mellifera*
- Extract the gi number and the sequence of each protein
- Calculate the relative frequencies of the 20 amino acids
- Store the data/information in the Oracle database through a network

Design a Web Page to Upload the Data

Browser View →

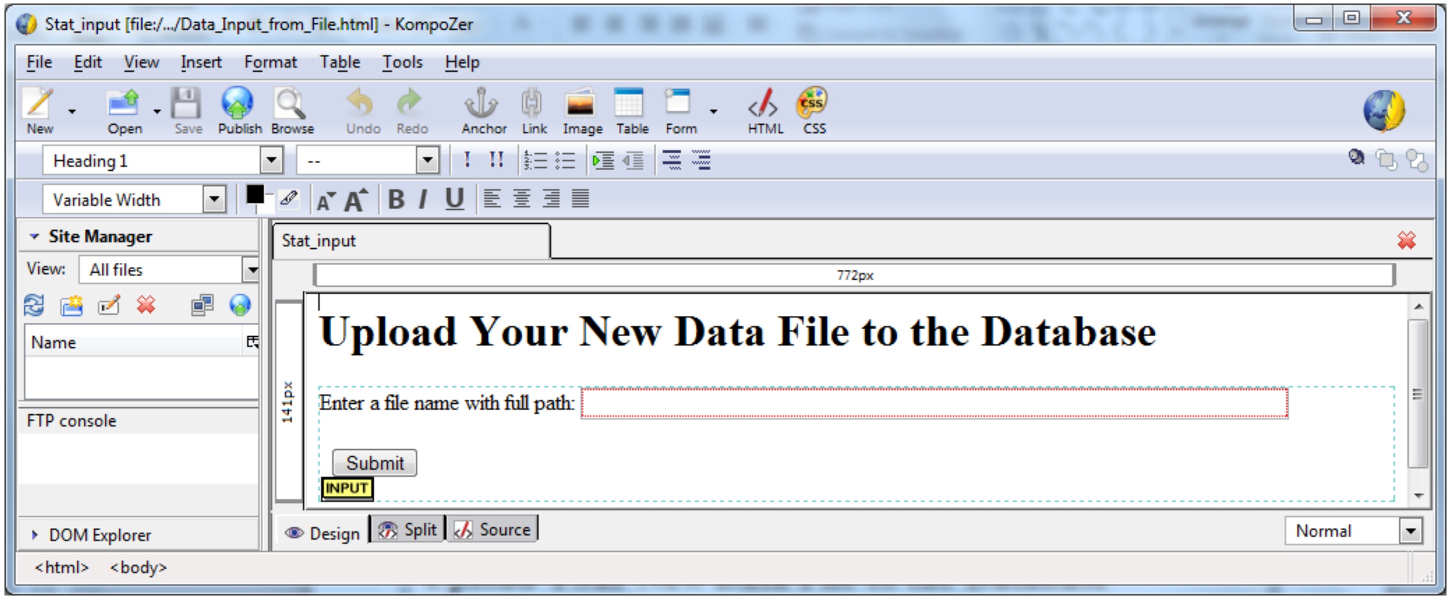


Upload Your New Data File to the Database

Enter a file name with full path:

Submit

KompoZer Design View →


Stat_input [file:/.../Data_Input_from_File.html] - KompoZer
File Edit View Insert Format Table Tools Help
New Open Save Publish Browse Undo Redo Anchor Link Image Table Form HTML CSS
Heading 1
Variable Width
Site Manager
View: All files
Name
FTP console
DOM Explorer
Stat_input
772px
141px
INPUT
Design Split Source
Normal
<html> <body>

Upload Your New Data File to the Database

Enter a file name with full path:

Submit

Python Coding: the General Algorithm

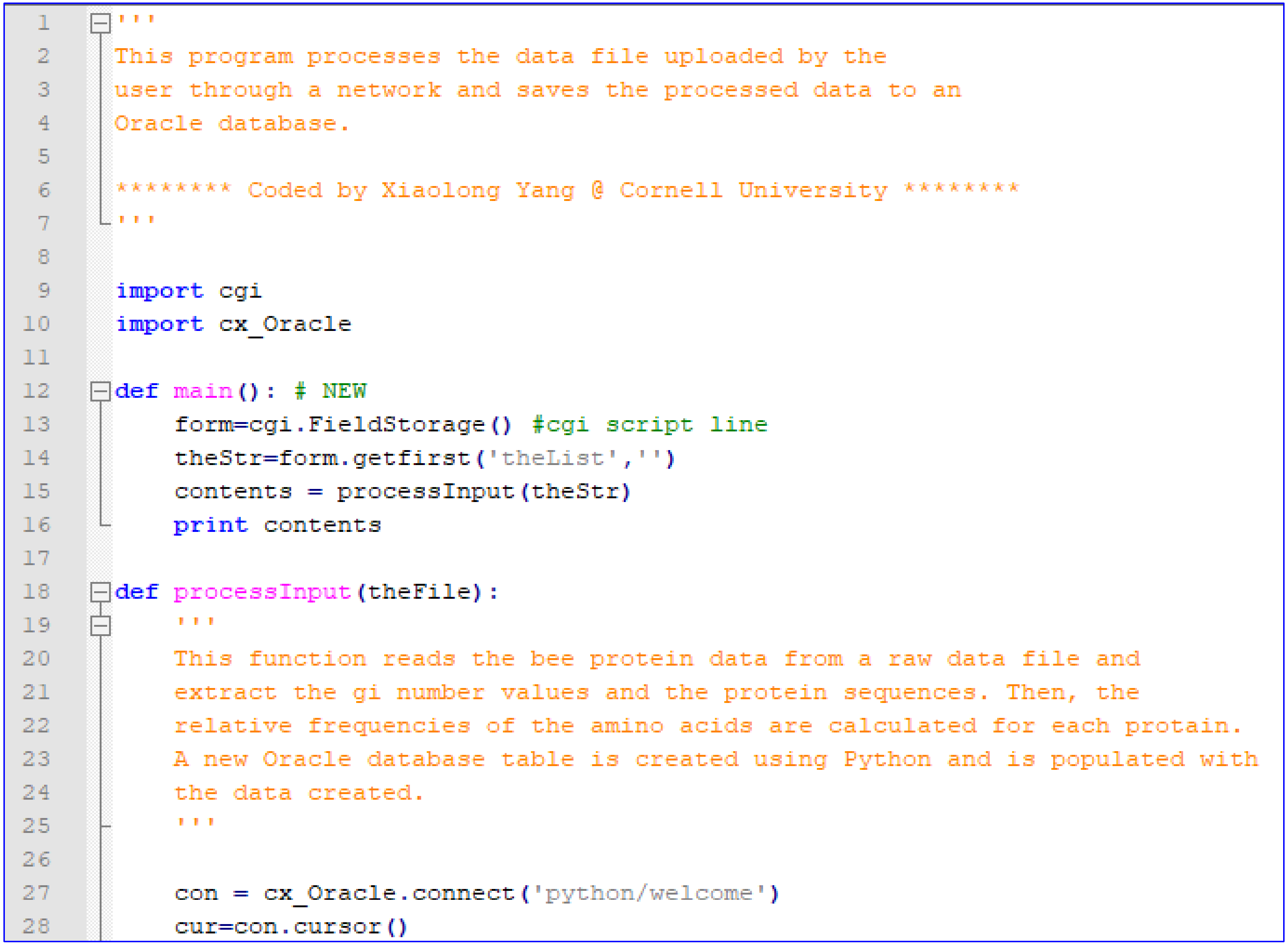
- Code two CGI (common gateway interface) files, one for reading and processing data and one for displaying database contents.
- Use Python functions to achieve the goal: main(), processInput(), …
- The main() function receives the file name from the web page and pass it to the processInput() function and finally display the results on a web page.
- The processInput() function creates a new database table in Oracle, reads and processes the data as planned earlier. More details on database handling:

  Use special features (e.g., ">gi|", "[Apis mellifera]") of the dataset to extract the gi numbers and protein sequences:

  * Use bind variables to populate the database table.

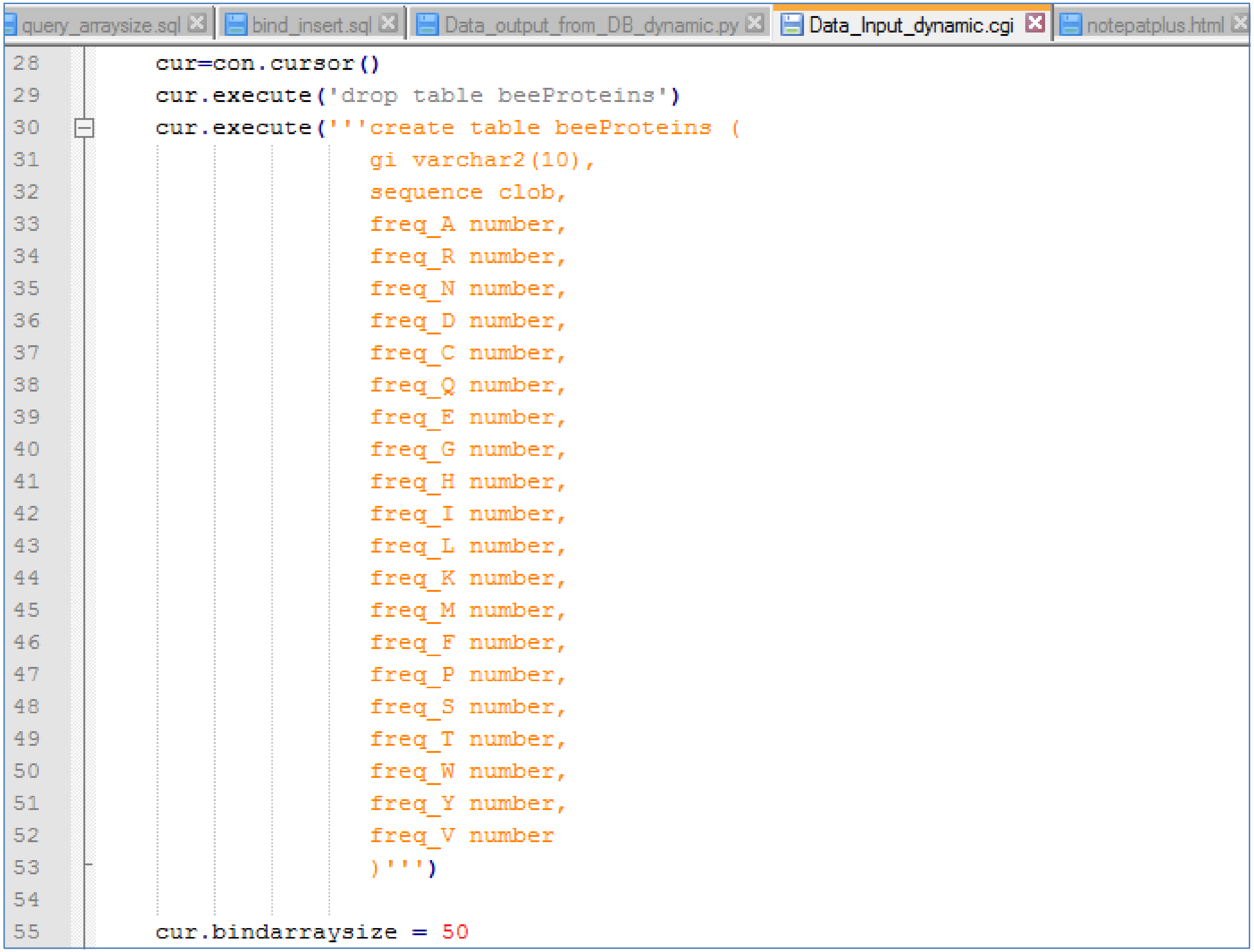
  * Write other functions when necessary, e.g., fileToStr() and makePage().
- Design two webpage templates, one to confirm uploading data to the database and one to display some contents retrieved from the database.

The 1st CGI File



```python
'''
This program processes the data file uploaded by the
user through a network and saves the processed data to an
Oracle database.

******** Coded by Xiaolong Yang @ Cornell University ********
'''

import cgi
import cx_Oracle

def main(): # NEW
    form=cgi.FieldStorage() #cgi script line
    theStr=form.getfirst('theList','')
    contents = processInput(theStr)
    print contents

def processInput(theFile):
    '''
    This function reads the bee protein data from a raw data file and
    extract the gi number values and the protein sequences. Then, the
    relative frequencies of the amino acids are calculated for each protain.
    A new Oracle database table is created using Python and is populated with
    the data created.
    '''

    con = cx_Oracle.connect('python/welcome')
    cur=con.cursor()
```

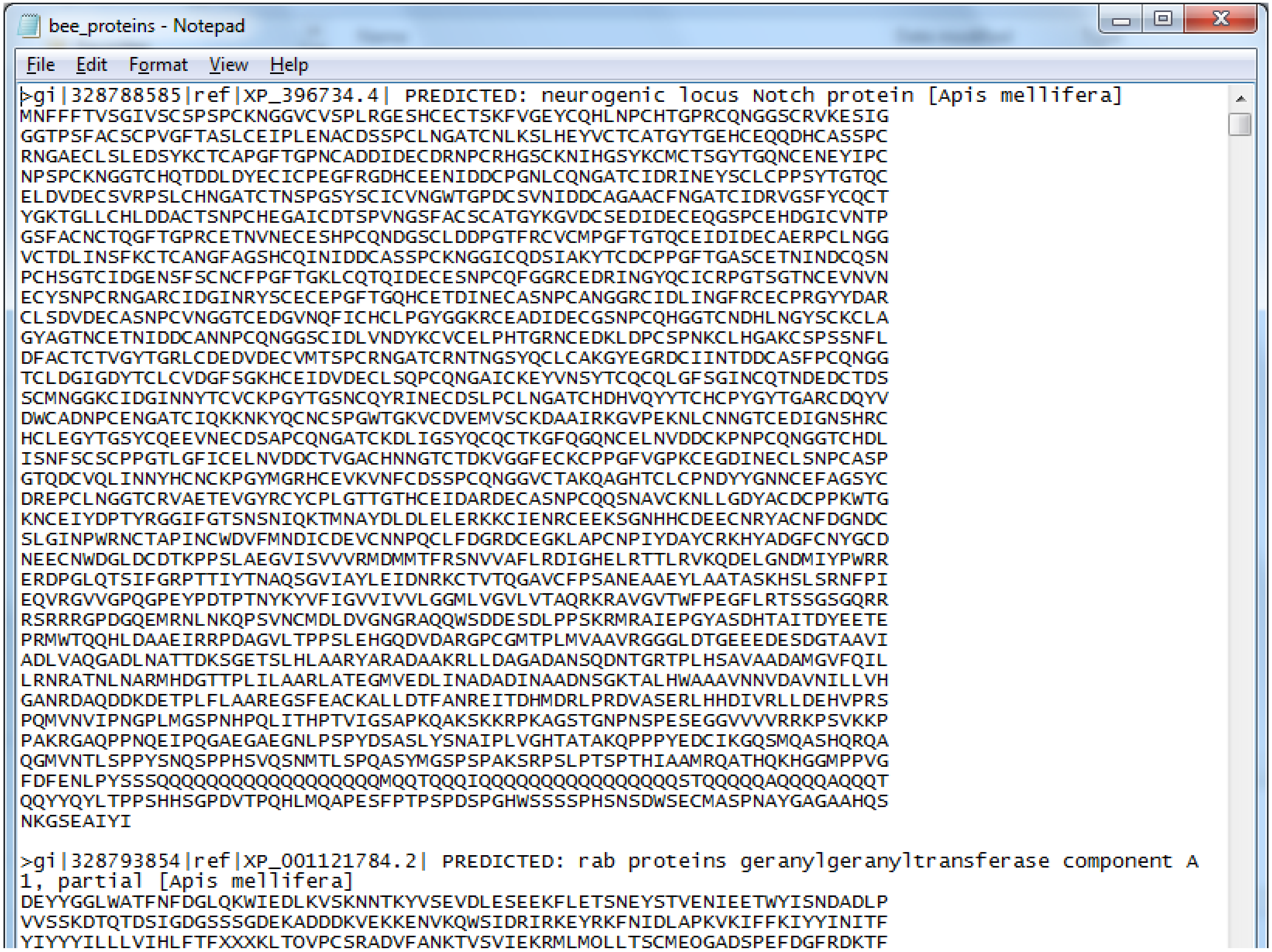
The 1st CGI File (cont'd)



query_arraysize.sql | bind_insert.sql | Data_output_from_DB_dynamic.py | Data_Input_dynamic.cgi | notepatplus.html

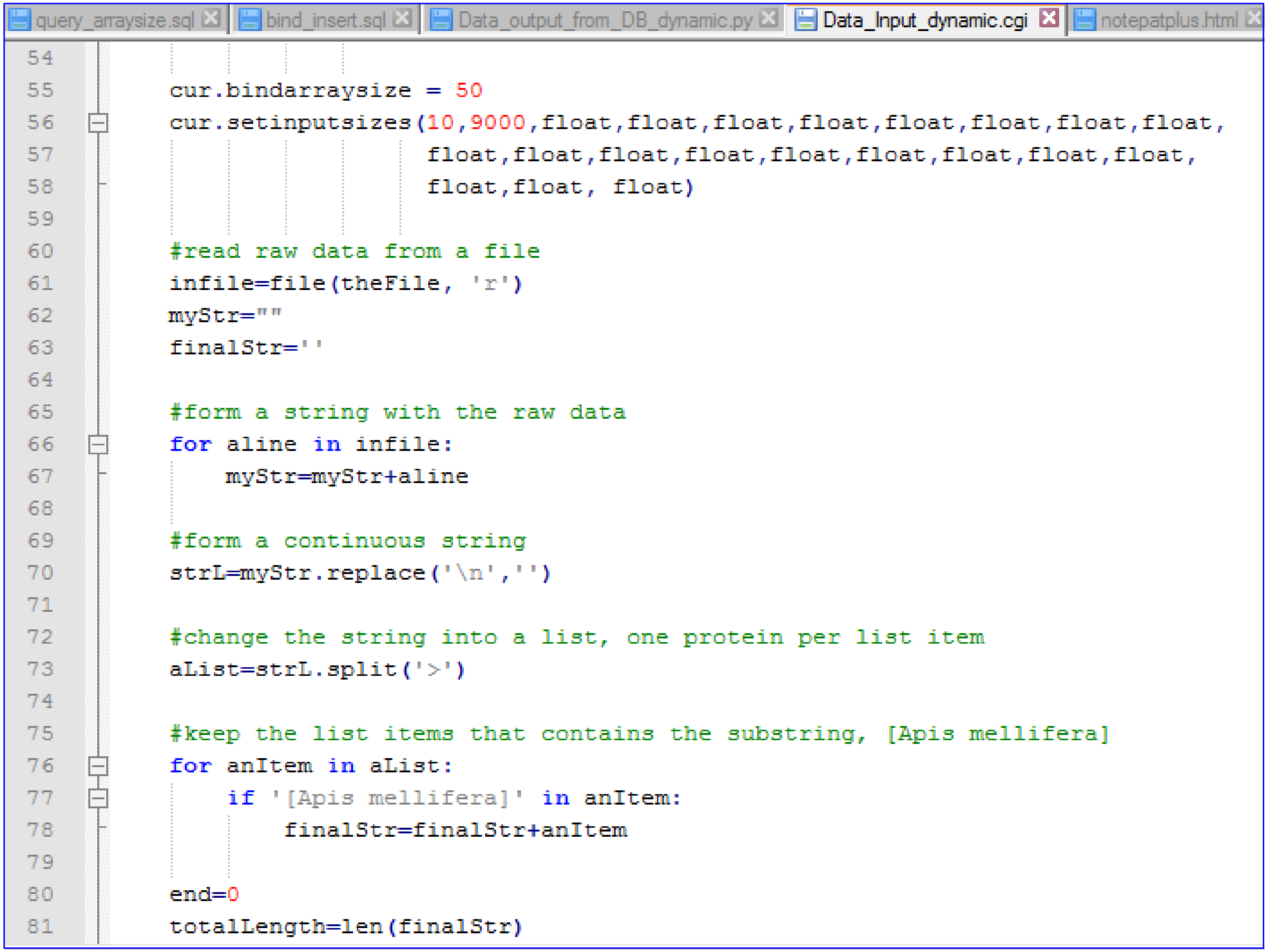
```
    cur=con.cursor()
    cur.execute('drop table beeProteins')
    cur.execute('''create table beeProteins (
                gi varchar2(10),
                sequence clob,
                freq_A number,
                freq_R number,
                freq_N number,
                freq_D number,
                freq_C number,
                freq_Q number,
                freq_E number,
                freq_G number,
                freq_H number,
                freq_I number,
                freq_L number,
                freq_K number,
                freq_M number,
                freq_F number,
                freq_P number,
                freq_S number,
                freq_T number,
                freq_W number,
                freq_Y number,
                freq_V number
                )''')

    cur.bindarraysize = 50
```

The Original Honey Bee Protein Data Set



```
>gi|328788585|ref|XP_396734.4| PREDICTED: neurogenic locus Notch protein [Apis mellifera]
MNFFFTVSGIVSCSPSPCKNGGVCVSPLRGESHCECTSKFVGEYCQHLNPCHTGPRCQNGGSCRVKESIG
GGTPSFACSCPVGFTASLCEIPLENACDSSPCLNGATCNLKSLHEYVCTCATGYTGEHCEQQDHCASSPC
RNGAECLSLEDSYKCTCAPGFTGPNCADDIDECDRNPCRHGSCKNIHGSYKCMCTSGYTGQNCENEYIPC
NPSPCKNGGTCHQTDDLDYECICPEGFRGDHCEENIDDCPGNLCQNGATCIDRINEYSCLCPPSYTGTQC
ELDVDECSVRPSLCHNGATCTNSPGSYSCICVNGWTGPDCSVNIDDCAGAACFNGATCIDRVGSFYCQCT
YGKTGLLCHLDDACTSNPCHEGAICDTSPVNGSFACSCATGYKGVDCSEDIDECEQGSPCEHDGICVNTP
GSFACNCTQGFTGPRCETNVNECESHPCQNDGSCLDDPGTFRCVCMPGFTGTQCEIDIDECAERPCLNGG
VCTDLINSFKCTCANGFAGSHCQINIDDCASSPCKNGGICQDSIAKYTCDCPPGFTGASCETNINDCQSN
PCHSGTCIDGENSFSCNCFPGFTGKLCQTQIDECESNPCQFGGRCEDRINGYQCICRPGTSGTNCEVNVN
ECYSNPCRNGARCIDGINRYSCECEPGFTGQHCETDINECASNPCANGGRCIDLINGFRCECPRGYYDAR
CLSDVDECASNPCVNGGTCEDGVNQFICHCLPGYGGKRCEADIDECGSNPCQHGGTCNDHLNGYSCKCLA
GYAGTNCETNIDDCANNPCQNGGSCIDLVNDYKCVCELPHTGRNCEDKLDPCSPNKCLHGAKCSPSSNFL
DFACTCTVGYTGRLCDEDVDECVMTSPCRNGATCRNTNGSYQCLCAKGYEGRDCIINTDDCASFPCQNGG
TCLDGIGDYTCLCVDGFSGKHCEIDVDECLSQPCQNGAICKEYVNSYTCQCQLGFSGINCQTNDEDCTDS
SCMNGGKCIDGINNYTCVCKPGYTGSNCQYRINECDSLPCLNGATCHDHVQYYTCHCPYGYTGARCDQYV
DWCADNPCENGATCIQKKNKYQCNCSPGWTGKVCDVEMVSCKDAAIRKGVPEKNLCNNGTCEDIGNSHRC
HCLEGYTGSYCQEEVNECDSAPCQNGATCKDLIGSYQCQCTKGFQGQNCELNVDDCKPNPCQNGGTCHDL
ISNFSCSCPPGTLGFICELNVDDCTVGACHNNGTCTDKVGGFECKCPPGFVGPKCEGDINECLSNPCASP
GTQDCVQLINNYHCNCKPGYMGRHCEVKVNFCDSSPCQNGGVCTAKQAGHTCLCPNDYYGNNCEFAGSYC
DREPCLNGGTCRVAETEVGYRCYCPLGTTGTHCEIDARDECASNPCQQSNAVCKNLLGDYACDCPPKWTG
KNCEIYDPTYRGGIFGTSNSNIQKTMNAYDLDLELERKKCIENRCEEKSGNHHCDEECNRYACNFDGNDC
SLGINPWRNCTAPINCWDVFMNDICDEVCNNPQCLFDGRDCEGKLAPCNPIYDAYCRKHYADGFCNYGCD
NEECNWDGLDCDTKPPSLAEGVISVVVRMDMMTFRSNVVAFLRDIGHELRTTLRVKQDELGNDMIYPWRR
ERDPGLQTSIFGRPTTIYTNAQSGVIAYLEIDNRKCTVTQGAVCFPSANEAAEYLAATASKHSLSRNFPI
EQVRGVVGPQGPEYPDTPTNYKYVFIGVVIVVLGGMLVGVLVTAQRKRAVGVTWFPEGFLRTSSGSGQRR
RSRRRGPDGQEMRNLNKQPSVNCMDLDVGNGRAQQWSDDESDLPPSKRMRAIEPGYASDHTAITDYEETE
PRMWTQQHLDAAEIRRPDAGVLTPPSLEHGQDVDARGPCGMTPLMVAAVRGGGLDTGEEEDESDGTAAVI
ADLVAQGADLNATTDKSGETSLHLAARYARADAAKRLLDAGADANSQDNTGRTPLHSAVAADAMGVFQIL
LRNRATNLNARMHDGTTPLILAARLATEGMVEDLINADADINAADNSGKTALHWAAAVNNVDAVNILLVH
GANRDAQDDKDETPLFLAAREGSFEACKALLDTFANREITDHMDRLPRDVASERLHHDIVRLLDEHVPRS
PQMVNVIPNGPLMGSPNHPQLITHPTVIGSAPKQAKSKKRPKAGSTGNPNSPESEGGVVVVRRKPSVKKP
PAKRGAQPPNQEIPQGAEGAEGNLPSPYDSASLYSNAIPLVGHTATAKQPPPYEDCIKGQSMQASHQRQA
QGMVNTLSPPYSNQSPPHSVQSNMTLSPQASYMGSPSPAKSRPSLPTSPTHIAAMRQATHQKHGGMPPVG
FDFENLPYSSSQQQQQQQQQQQQQQQQQQMQQTQQQIQQQQQQQQQQQQQQQQSTQQQQQAQQQQAQQQT
QQYYQYLTPPSHHSGPDVTPQHLMQAPESFPTPSPDSPGHWSSSSPHSNSDWSECMASPNAYGAGAAHQS
NKGSEAIYI

>gi|328793854|ref|XP_001121784.2| PREDICTED: rab proteins geranylgeranyltransferase component A
1, partial [Apis mellifera]
DEYYGGLWATFNFDGLQKWIEDLKVSKNNTKYVSEVDLESEEKFLETSNEYSTVENIEETWYISNDADLP
VVSSKDTQTDSIGDGSSSGDEKADDDKVEKKENVKQWSIDRIRKEYRKFNIDLAPKVKIFFKIYYINITF
YIYYYILLLVIHLFTFXXXKLTQVPCSRADVFANKTVSVIEKRMLMQLLTSCMEQGADSPEFDGFRDKTF
```

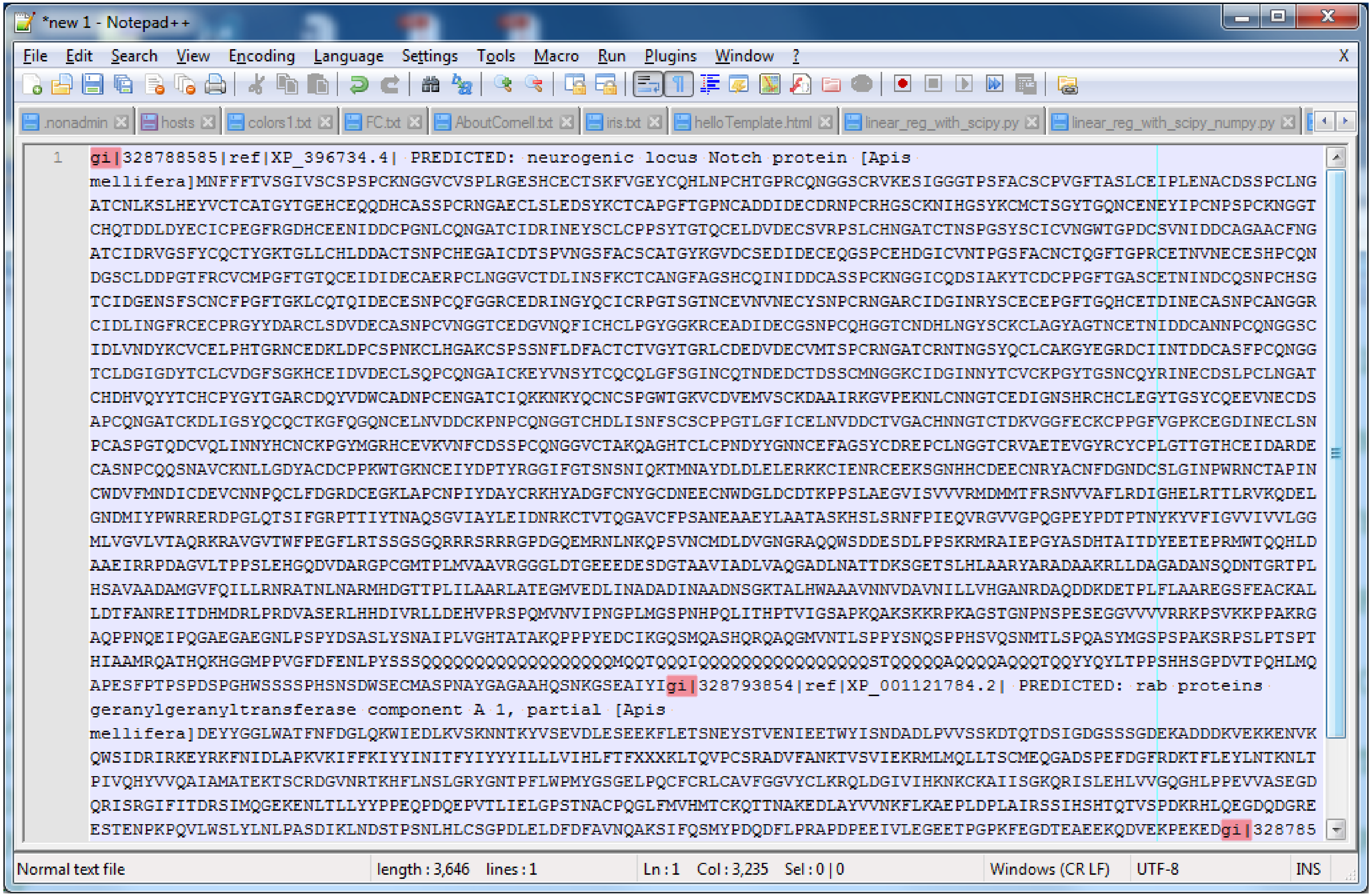
The 1st CGI File (cont'd)



query_arraysize.sql | bind_insert.sql | Data_output_from_DB_dynamic.py | Data_Input_dynamic.cgi | notepatplus.html

```

    cur.bindarraysize = 50
    cur.setinputsizes(10,9000,float,float,float,float,float,float,float,float,
                      float,float,float,float,float,float,float,float,float,
                      float,float, float)

    #read raw data from a file
    infile=file(theFile, 'r')
    myStr=""
    finalStr=''

    #form a string with the raw data
    for aline in infile:
        myStr=myStr+aline

    #form a continuous string
    strL=myStr.replace('\n','')

    #change the string into a list, one protein per list item
    aList=strL.split('>')

    #keep the list items that contains the substring, [Apis mellifera]
    for anItem in aList:
        if '[Apis mellifera]' in anItem:
            finalStr=finalStr+anItem

    end=0
    totalLength=len(finalStr)
```

The Processed Bee Protein Data

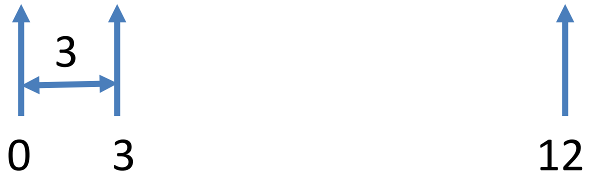


```
gi|328788585|ref|XP_396734.4| PREDICTED: neurogenic locus Notch protein [Apis
mellifera]MNFFFTVSGIVSCSPSPCKNGGVCVSPLRGESHCECTSKFVGEYCQHLNPCHTGPRCQNGGSCRVKESIGGGTPSFACSCPVGFTASLCEIPLENACDSSPCLNG
ATCNLKSLHEYVCTCATGYTGEHCEQQDHCASSPCRNGAECLSLEDSYKCTCAPGFTGPNCADDIDECDRNPCRHGSCKNIHGSYKCMCTSGYTGQNCENEYIPCNPSPCKNGGT
CHQTDDLDYECICPEGFRGDHCEENIDDCPGNLCQNGATCIDRINEYSCLCPPSYTGTQCELDVDECSVRPSLCHNGATCTNSPGSYSCICVNGWTGPDCSVNIDDCAGAACFNG
ATCIDRVGSFYCQCTYGKTGLLCHLDDACTSNPCHEGAICDTSPVNGSFACSCATGYKGVDCSEDIDECEQGSPCEHDGICVNTPGSFACNCTQGFTGPRCETNVNECESHPCQN
DGSCLDDPGTFRCVCMPGFTGTQCEIDIDECAERPCLNGGVCTDLINSFKCTCANGFAGSHCQINIDDCASSPCKNGGICQDSIAKYTCDCPPGFTGASCETNINDCQSNPCHSG
TCIDGENSFSCNCFPGFTGKLCQTQIDECESNPCQFGGRCEDRINGYQCICRPGTSGTNCEVNVNECYSNPCRNGARCIDGINRYSCECEPGFTGQHCETDINECASNPCANGGR
CIDLINGFRCECPRGYYDARCLSDVDECASNPCVNGGTCEDGVNQFICHCLPGYGGKRCEADIDECGSNPCQHGGTCNDHLNGYSCKCLAGYAGTNCETNIDDCANNPCQNGGSC
IDLVNDYKCVCELPHTGRNCEDKLDPCSPNKCLHGAKCSPSSNFLDFACTCTVGYTGRLCDEDVDECVMTSPCRNGATCRNTNGSYQCLCAKGYEGRDCIINTDDCASFPCQNGG
TCLDGIGDYTCLCVDGFSGKHCEIDVDECLSQPCQNGAICKEYVNSYTCQCQLGFSGINCQTNDEDCTDSSCMNGGKCIDGINNYTCVCKPGYTGSNCQYRINECDSLPCLNGAT
CHDHVQYYTCHCPYGYTGARCDQYVDWCADNPCENGATCIQKKNKYQCNCSPGWTGKVCDVEMVSCKDAAIRKGVPEKNLCNNGTCEDIGNSHRCHCLEGYTGSYCQEEVNECDS
APCQNGATCKDLIGSYQCQCTKGFQGQNCELNVDDCKPNPCQNGGTCHDLISNFSCSCPPGTLGFICELNVDDCTVGACHNNGTCTDKVGGFECKCPPGFVGPKCEGDINECLSN
PCASPGTQDCVQLINNYHCNCKPGYMGRHCEVKVNFCDSSPCQNGGVCTAKQAGHTCLCPNDYYGNNCEFAGSYCDREPCLNGGTCRVAETEVGYRCYCPLGTTGTHCEIDARDE
CASNPCQQSNAVCKNLLGDYACDCPPKWTGKNCEIYDPTYRGGIFGTSNSNIQKTMNAYDLDLELERKKCIENRCEEKSGNHHCDEECNRYACNFDGNDCSLGINPWRNCTAPIN
CWDVFMNDICDEVCNNPQCLFDGRDCEGKLAPCNPIYDAYCRKHYADGFCNYGCDNEECNWDGLDCDTKPPSLAEGVISVVVRMDMMTFRSNVVAFLRDIGHELRTTLRVKQDEL
GNDMIYPWRRERDPGLQTSIFGRPTTIYTNAQSGVIAYLEIDNRKCTVTQGAVCFPSANEAAEYLAATASKHSLSRNFPIEQVRGVVGPQGPEYPDTPTNYKYVFIGVVIVVLGG
MLVGVLVTAQRKRAVGVTWFPEGFLRTSSGSGQRRRSRRRGPDGQEMRNLNKQPSVNCMDLDVGNGRAQQWSDDESDLPPSKRMRAIEPGYASDHTAITDYEETEPRMWTQQHLD
AAEIRRPDAGVLTPPSLEHGQDVDARGPCGMTPLMVAAVRGGGLDTGEEEDESDGTAAVIADLVAQGADLNATTDKSGETSLHLAARYARADAAKRLLDAGADANSQDNTGRTPL
HSAVAADAMGVFQILLRNRATNLNARMHDGTTPLILAARLATEGMVEDLINADADINAADNSGKTALHWAAAVNNVDAVNILLVHGANRDAQDDKDETPLFLAAREGSFEACKAL
LDTFANREITDHMDRLPRDVASERLHHDIVRLLDEHVPRSPQMVNVIPNGPLMGSPNHPQLITHPTVIGSAPKQAKSKKRPKAGSTGNPNSPESEGGVVVVRRKPSVKKPPAKRG
AQPPNQEIPQGAEGAEGNLPSPYDSASLYSNAIPLVGHTATAKQPPPYEDCIKGQSMQASHQRQAQGMVNTLSPPYSNQSPPHSVQSNMTLSPQASYMGSPSPAKSRPSLPTSPT
HIAAMRQATHQKHGGMPPVGFDFENLPYSSSQQQQQQQQQQQQQQQQQQMQQTQQQIQQQQQQQQQQQQQQQQSTQQQQQAQQQQAQQQTQQYYQYLTPPSHHSGPDVTPQHLMQ
APESFPTPSPDSPGHWSSSSPHSNSDWSECMASPNAYGAGAAHQSNKGSEAIYIgi|328793854|ref|XP_001121784.2| PREDICTED: rab proteins
geranylgeranyltransferase component A 1, partial [Apis
mellifera]DEYYGGLWATFNFDGLQKWIEDLKVSKNNTKYVSEVDLESEEKFLETSNEYSTVENIEETWYISNDADLPVVSSKDTQTDSIGDGSSSGDEKADDDKVEKKENVK
QWSIDRIRKEYRKFNIDLAPKVKIFFKIYYINITFYIYYYILLLVIHLFTFXXXKLTQVPCSRADVFANKTVSVIEKRMLMQLLTSCMEQGADSPEFDGFRDKTFLEYLNTKNLT
PIVQHYVVQAIAMATEKTSCRDGVNRTKHFLNSLGRYGNTPFLWPMYGSGELPQCFCRLCAVFGGVYCLKRQLDGIVIHKNKCKAIISGKQRISLEHLVVGQGHLPPEVVASEGD
QRISRGIFITDRSIMQGEKENLTLLYYPPEQPDQEPVTLIELGPSTNACPQGLFMVHMTCKQTTNAKEDLAYVVNKFLKAEPLDPLAIRSSIHSHTQTVSPDKRHLQEGDQDGRE
ESTENPKPQVLWSLYLNLPASDIKLNDSTPSNLHLCSGPDLELDFDFAVNQAKSIFQSMYPDQDFLPRAPDPEEIVLEGEETPGPKFEGDTEAEEKQDVEKPEKEDgi|328785
```

Normal text file | length : 3,646 lines : 1 | Ln : 1 Col : 3,235 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

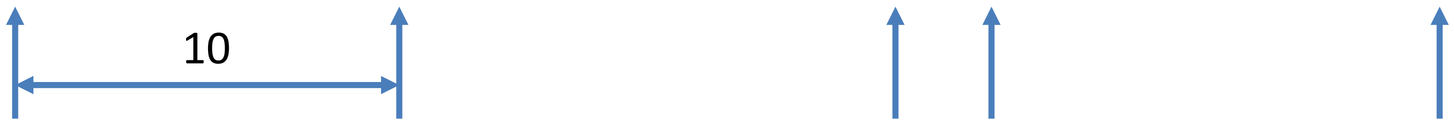
Slice out the Sub-strings of Interest from the String Containing the Bee Protein Data

gi|328788585|ref|XP_396734.4|

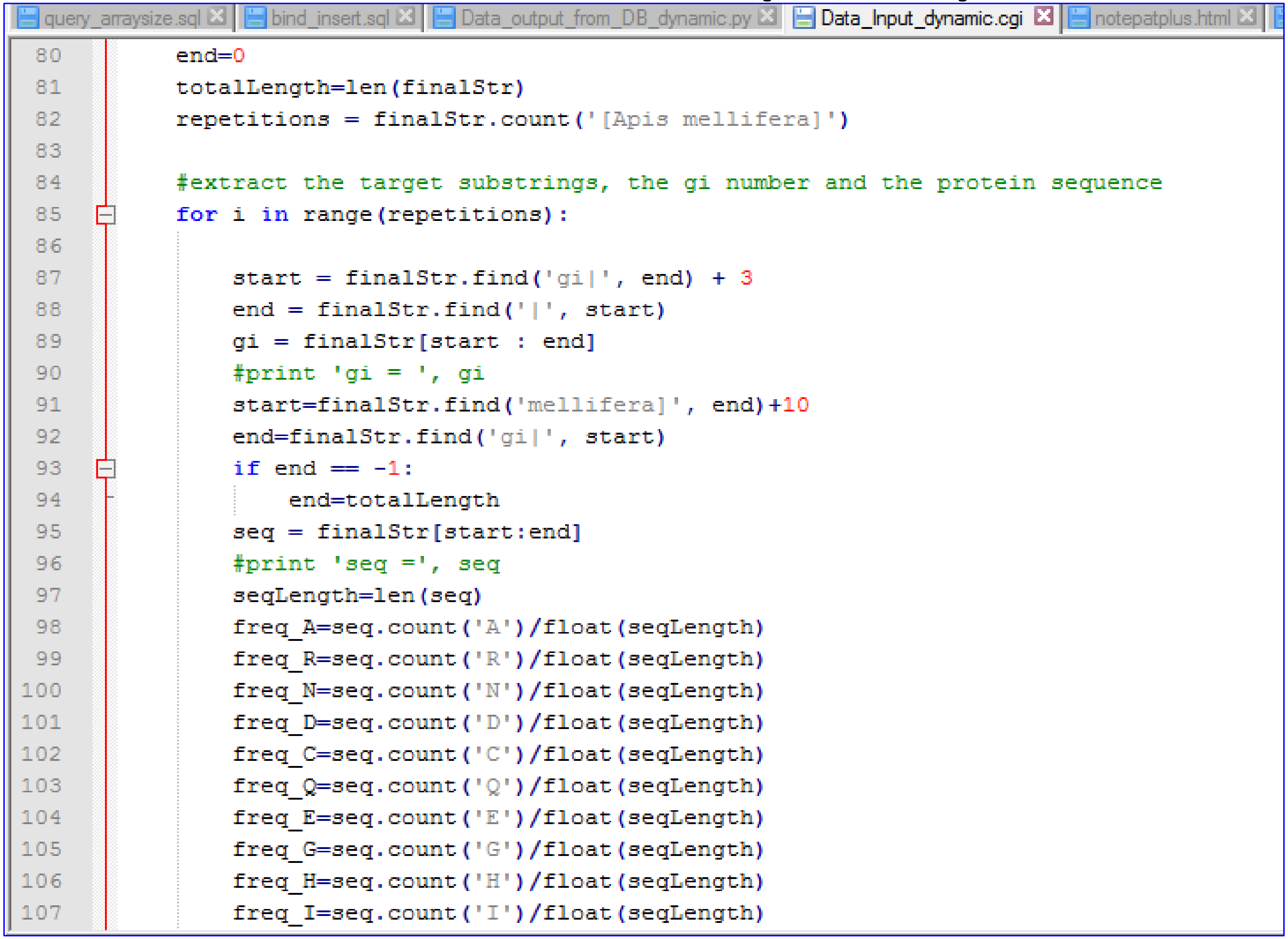


PREDICTED: neurogenic locus Notch protein

[Apis mellifera]MNFFF…AIYIgi|328793854|



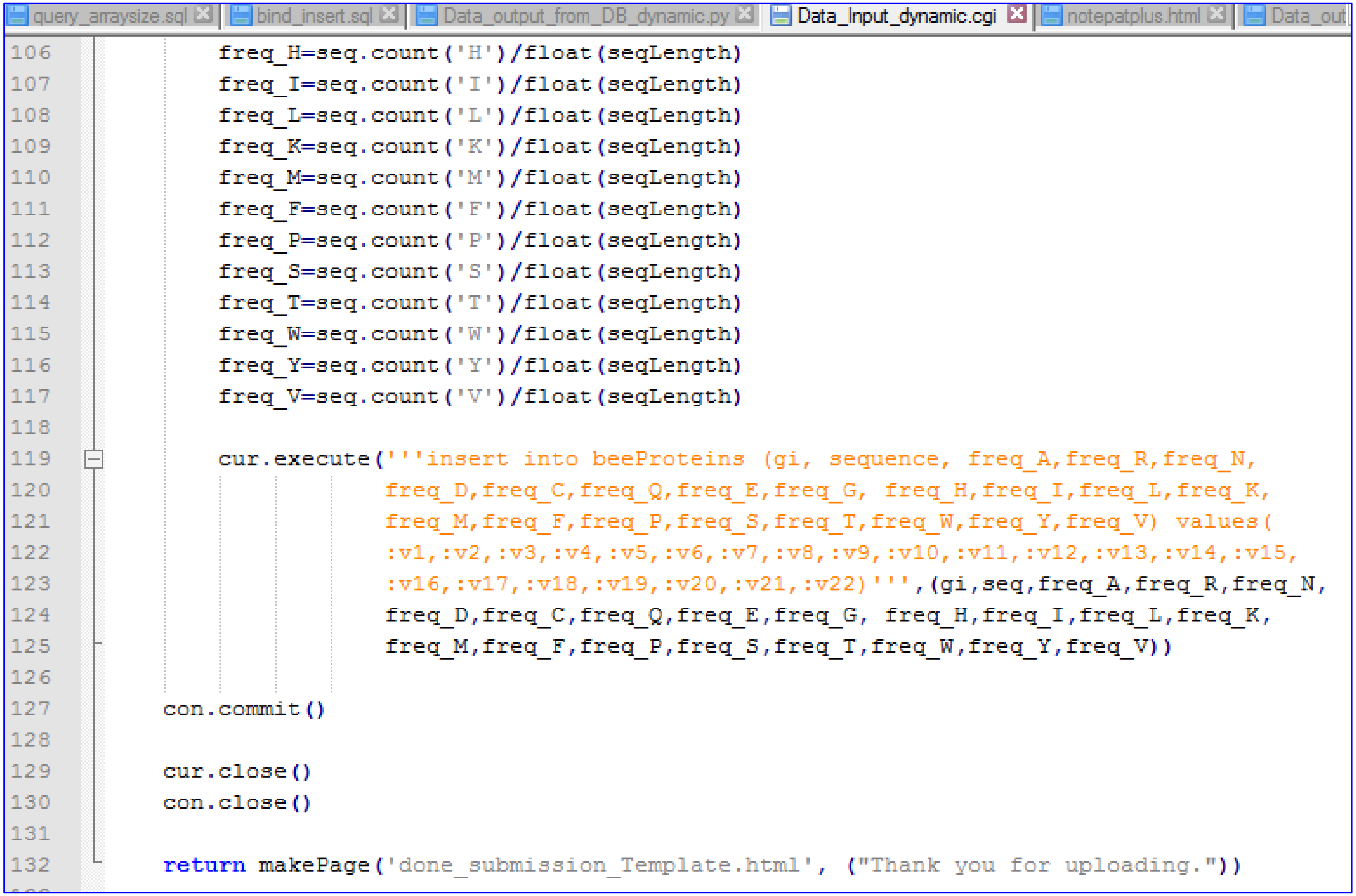
The 1st CGI File (cont'd)



query_arraysize.sql | bind_insert.sql | Data_output_from_DB_dynamic.py | Data_Input_dynamic.cgi | notepatplus.html

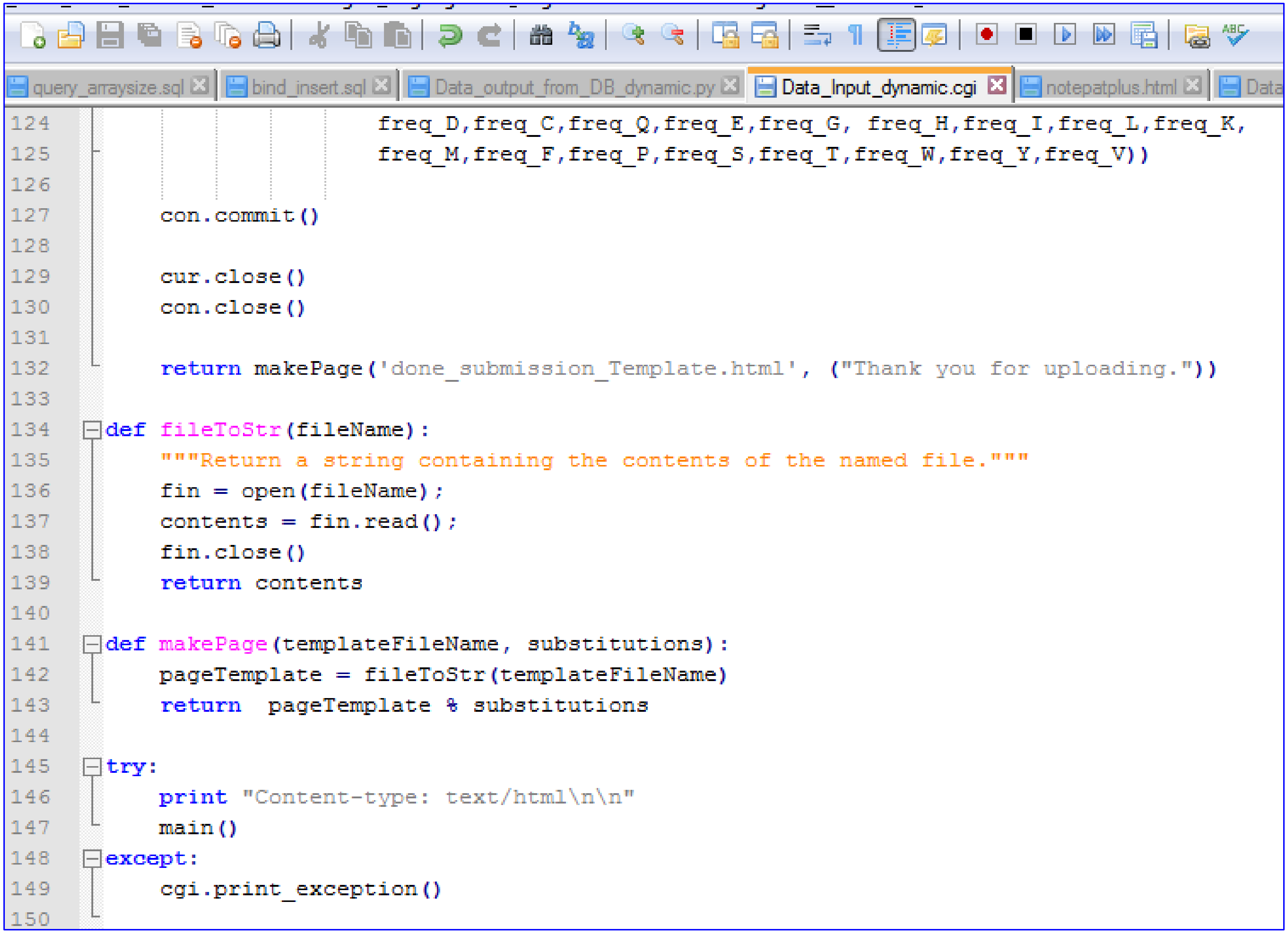
```
    end=0
    totalLength=len(finalStr)
    repetitions = finalStr.count('[Apis mellifera]')

    #extract the target substrings, the gi number and the protein sequence
    for i in range(repetitions):

        start = finalStr.find('gi|', end) + 3
        end = finalStr.find('|', start)
        gi = finalStr[start : end]
        #print 'gi = ', gi
        start=finalStr.find('mellifera]', end)+10
        end=finalStr.find('gi|', start)
        if end == -1:
            end=totalLength
        seq = finalStr[start:end]
        #print 'seq =', seq
        seqLength=len(seq)
        freq_A=seq.count('A')/float(seqLength)
        freq_R=seq.count('R')/float(seqLength)
        freq_N=seq.count('N')/float(seqLength)
        freq_D=seq.count('D')/float(seqLength)
        freq_C=seq.count('C')/float(seqLength)
        freq_Q=seq.count('Q')/float(seqLength)
        freq_E=seq.count('E')/float(seqLength)
        freq_G=seq.count('G')/float(seqLength)
        freq_H=seq.count('H')/float(seqLength)
        freq_I=seq.count('I')/float(seqLength)
```

The 1st CGI File (cont'd)



```
        freq_H=seq.count('H')/float(seqLength)
        freq_I=seq.count('I')/float(seqLength)
        freq_L=seq.count('L')/float(seqLength)
        freq_K=seq.count('K')/float(seqLength)
        freq_M=seq.count('M')/float(seqLength)
        freq_F=seq.count('F')/float(seqLength)
        freq_P=seq.count('P')/float(seqLength)
        freq_S=seq.count('S')/float(seqLength)
        freq_T=seq.count('T')/float(seqLength)
        freq_W=seq.count('W')/float(seqLength)
        freq_Y=seq.count('Y')/float(seqLength)
        freq_V=seq.count('V')/float(seqLength)

        cur.execute('''insert into beeProteins (gi, sequence, freq_A,freq_R,freq_N,
                        freq_D,freq_C,freq_Q,freq_E,freq_G, freq_H,freq_I,freq_L,freq_K,
                        freq_M,freq_F,freq_P,freq_S,freq_T,freq_W,freq_Y,freq_V) values(
                        :v1,:v2,:v3,:v4,:v5,:v6,:v7,:v8,:v9,:v10,:v11,:v12,:v13,:v14,:v15,
                        :v16,:v17,:v18,:v19,:v20,:v21,:v22)''',(gi,seq,freq_A,freq_R,freq_N,
                        freq_D,freq_C,freq_Q,freq_E,freq_G, freq_H,freq_I,freq_L,freq_K,
                        freq_M,freq_F,freq_P,freq_S,freq_T,freq_W,freq_Y,freq_V))

    con.commit()

    cur.close()
    con.close()

    return makePage('done_submission_Template.html', ("Thank you for uploading."))
```

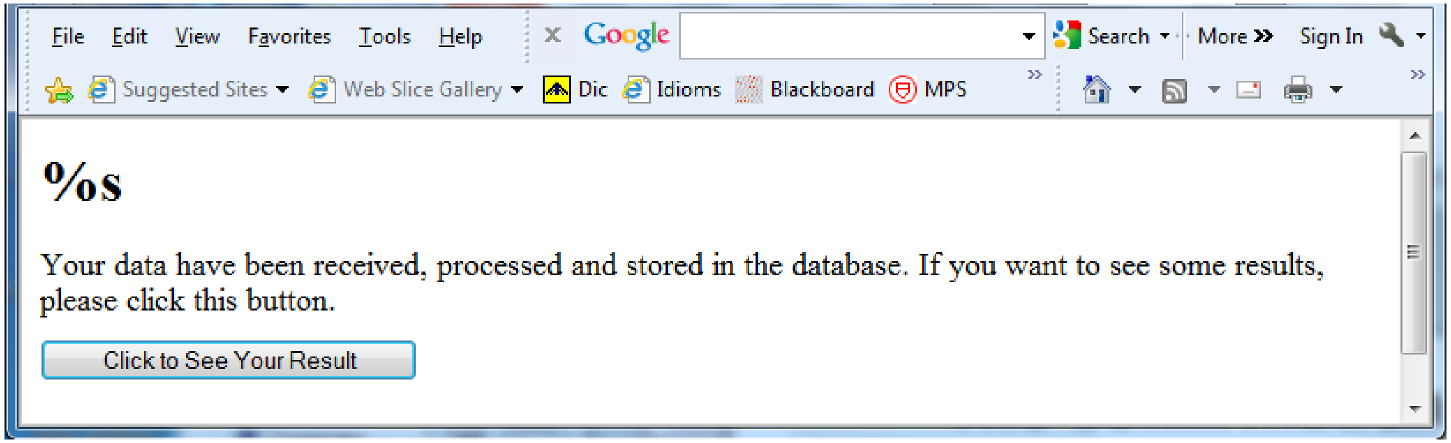
The 1st CGI File (cont'd)



```
                                    freq_D,freq_C,freq_Q,freq_E,freq_G, freq_H,freq_I,freq_L,freq_K,
                                    freq_M,freq_F,freq_P,freq_S,freq_T,freq_W,freq_Y,freq_V))

        con.commit()

        cur.close()
        con.close()

        return makePage('done_submission_Template.html', ("Thank you for uploading."))

def fileToStr(fileName):
    """Return a string containing the contents of the named file."""
    fin = open(fileName);
    contents = fin.read();
    fin.close()
    return contents

def makePage(templateFileName, substitutions):
    pageTemplate = fileToStr(templateFileName)
    return  pageTemplate % substitutions

try:
    print "Content-type: text/html\n\n"
    main()
except:
    cgi.print_exception()
```

The Confirmation Page Template

IE View →

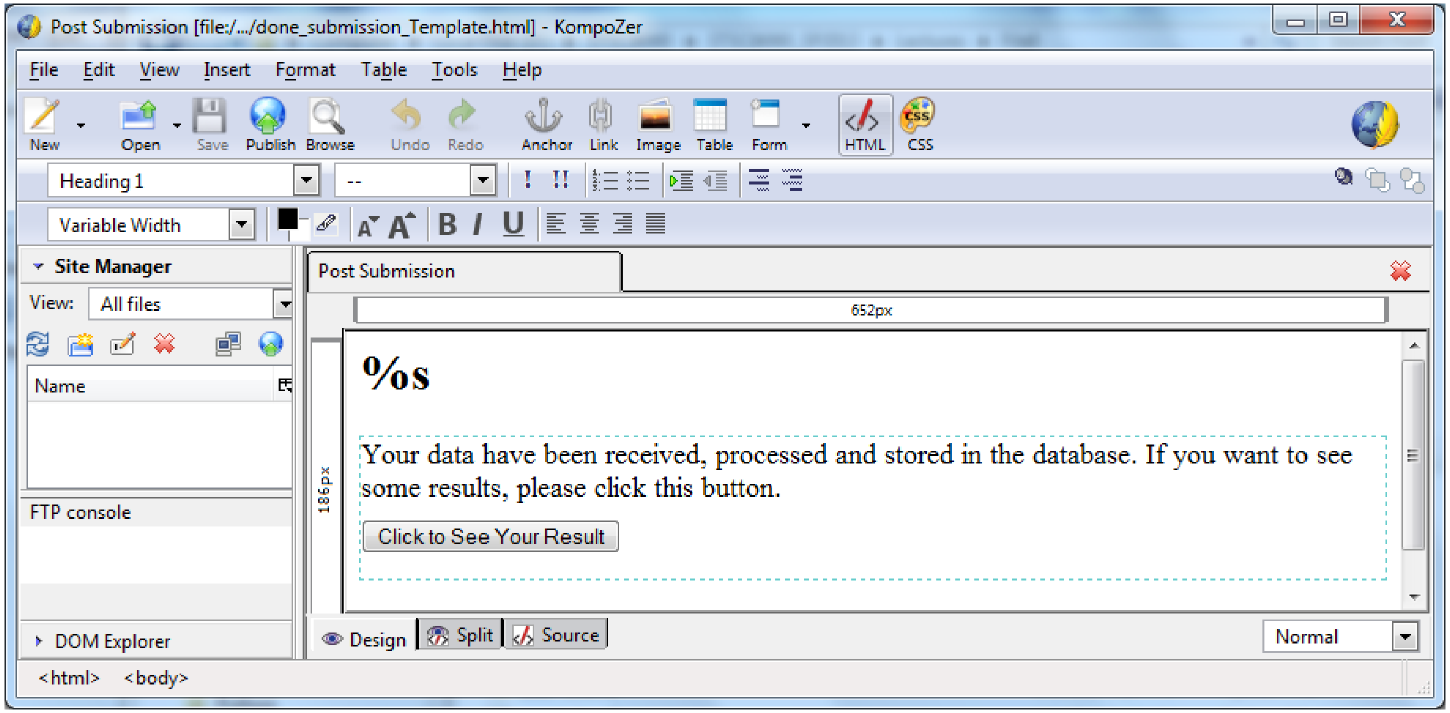


%s

Your data have been received, processed and stored in the database. If you want to see some results, please click this button.

Click to See Your Result

KompoZer Design View →

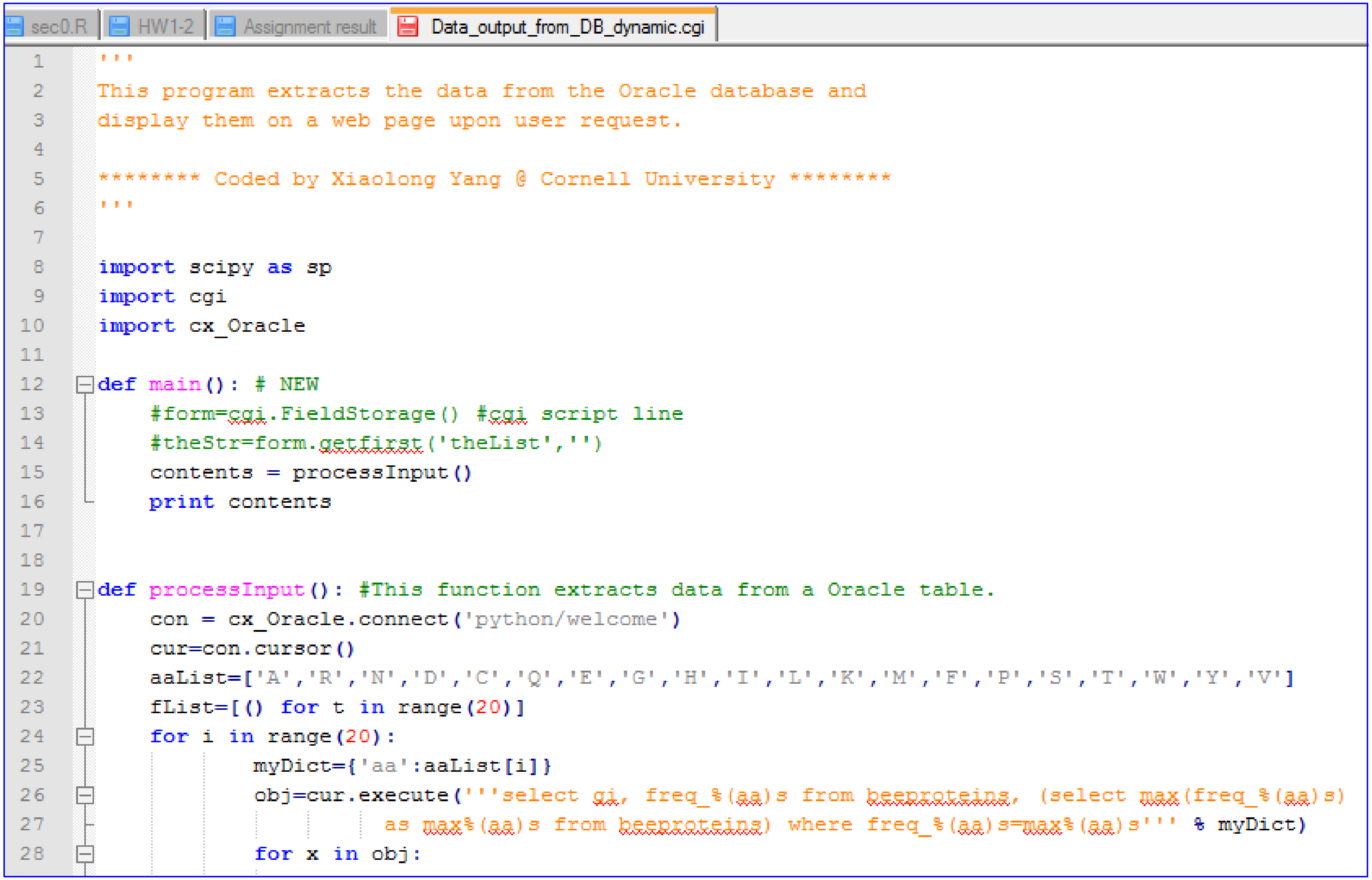


%s

Your data have been received, processed and stored in the database. If you want to see some results, please click this button.

Click to See Your Result

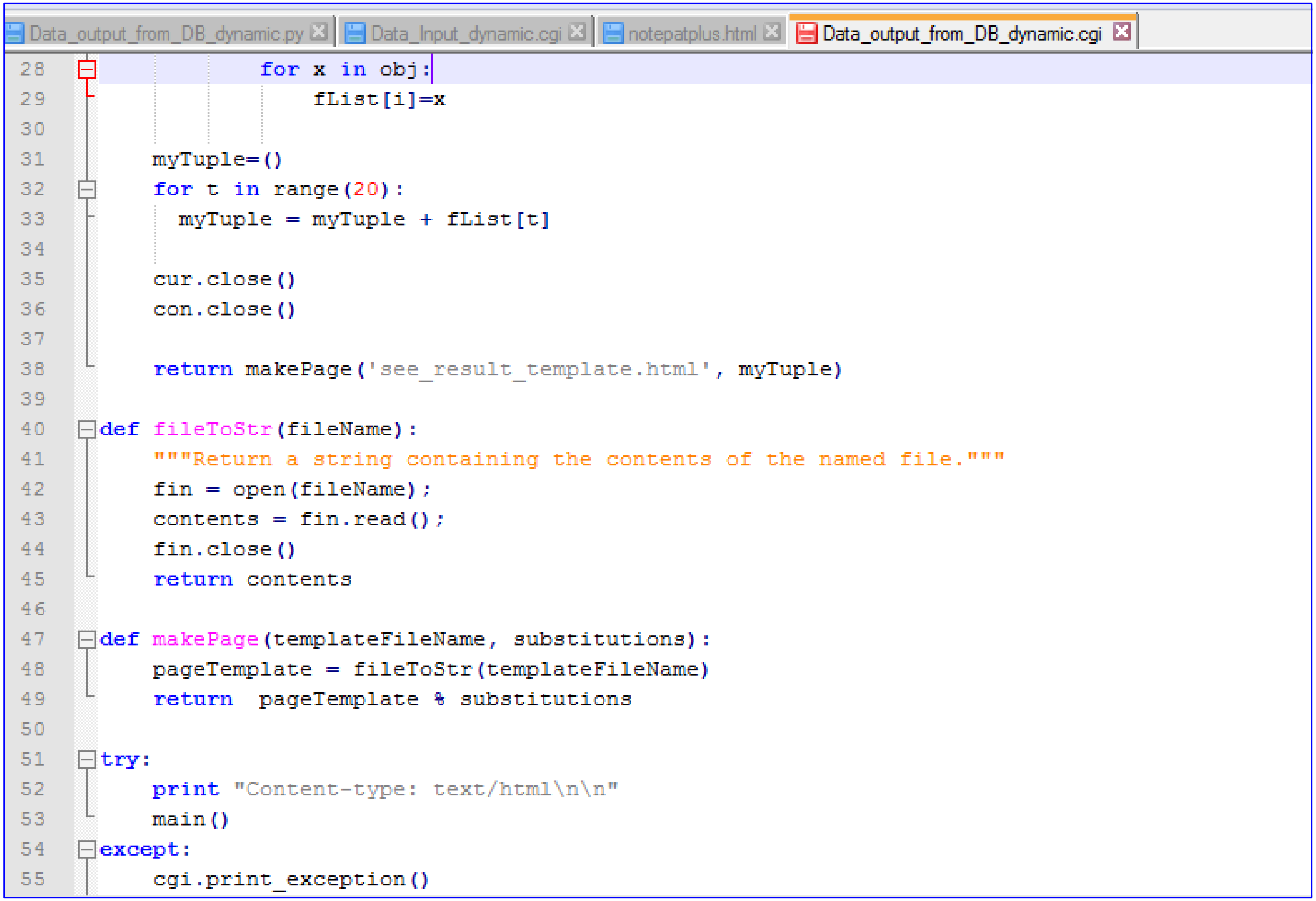
The 2nd CGI File



sec0.R | HW1-2 | Assignment result | Data_output_from_DB_dynamic.cgi

```
'''
This program extracts the data from the Oracle database and
display them on a web page upon user request.

******** Coded by Xiaolong Yang @ Cornell University ********
'''

import scipy as sp
import cgi
import cx_Oracle

def main(): # NEW
    #form=cgi.FieldStorage() #cgi script line
    #theStr=form.getfirst('theList','')
    contents = processInput()
    print contents


def processInput(): #This function extracts data from a Oracle table.
    con = cx_Oracle.connect('python/welcome')
    cur=con.cursor()
    aaList=['A','R','N','D','C','Q','E','G','H','I','L','K','M','F','P','S','T','W','Y','V']
    fList=[() for t in range(20)]
    for i in range(20):
        myDict={'aa':aaList[i]}
        obj=cur.execute('''select gi, freq_%(aa)s from beeproteins, (select max(freq_%(aa)s)
            as max%(aa)s from beeproteins) where freq_%(aa)s=max%(aa)s''' % myDict)
        for x in obj:
```

The 2nd CGI File (cont’d)

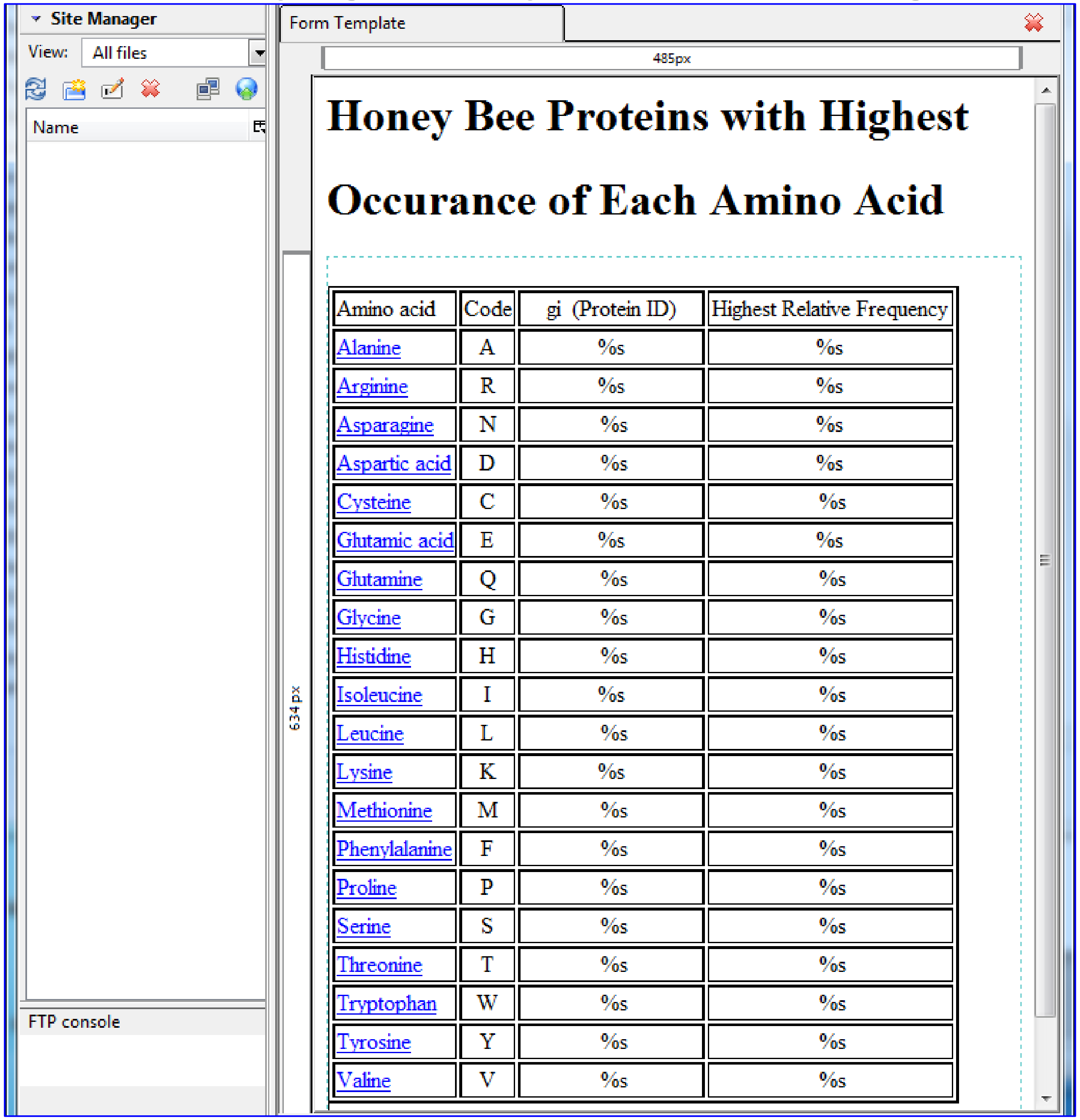


Data_output_from_DB_dynamic.py | Data_Input_dynamic.cgi | notepatplus.html | Data_output_from_DB_dynamic.cgi

```
                for x in obj:
                    fList[i]=x

        myTuple=()
        for t in range(20):
          myTuple = myTuple + fList[t]

        cur.close()
        con.close()

        return makePage('see_result_template.html', myTuple)

def fileToStr(fileName):
    """Return a string containing the contents of the named file."""
    fin = open(fileName);
    contents = fin.read();
    fin.close()
    return contents

def makePage(templateFileName, substitutions):
    pageTemplate = fileToStr(templateFileName)
    return  pageTemplate % substitutions

try:
    print "Content-type: text/html\n\n"
    main()
except:
    cgi.print_exception()
```

The Database-Driven Webpage Template: the Design View

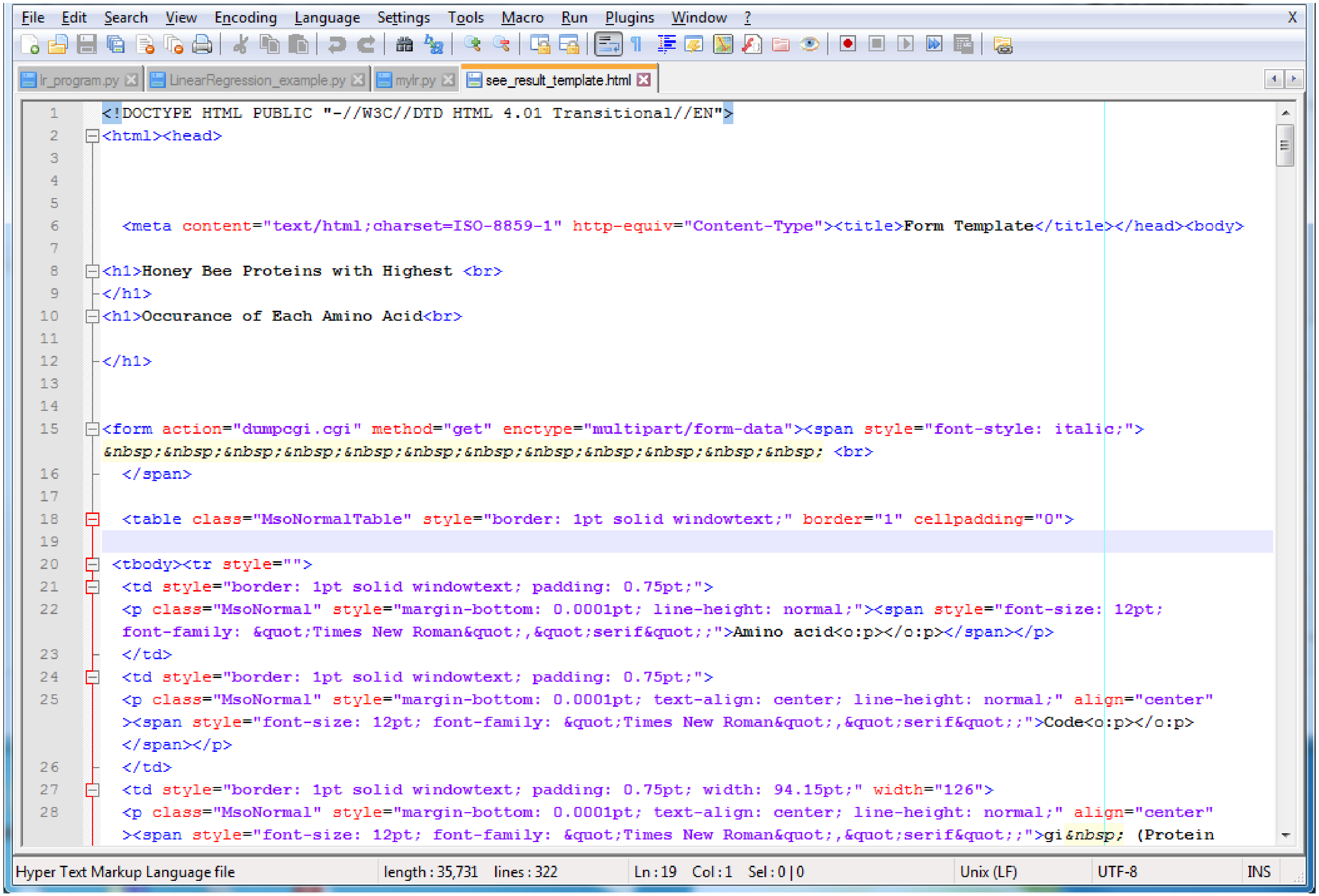
see_result_template.html



Honey Bee Proteins with Highest Occurance of Each Amino Acid

Amino acid	Code	gi (Protein ID)	Highest Relative Frequency
Alanine	A	%s	%s
Arginine	R	%s	%s
Asparagine	N	%s	%s
Aspartic acid	D	%s	%s
Cysteine	C	%s	%s
Glutamic acid	E	%s	%s
Glutamine	Q	%s	%s
Glycine	G	%s	%s
Histidine	H	%s	%s
Isoleucine	I	%s	%s
Leucine	L	%s	%s
Lysine	K	%s	%s
Methionine	M	%s	%s
Phenylalanine	F	%s	%s
Proline	P	%s	%s
Serine	S	%s	%s
Threonine	T	%s	%s
Tryptophan	W	%s	%s
Tyrosine	Y	%s	%s
Valine	V	%s	%s

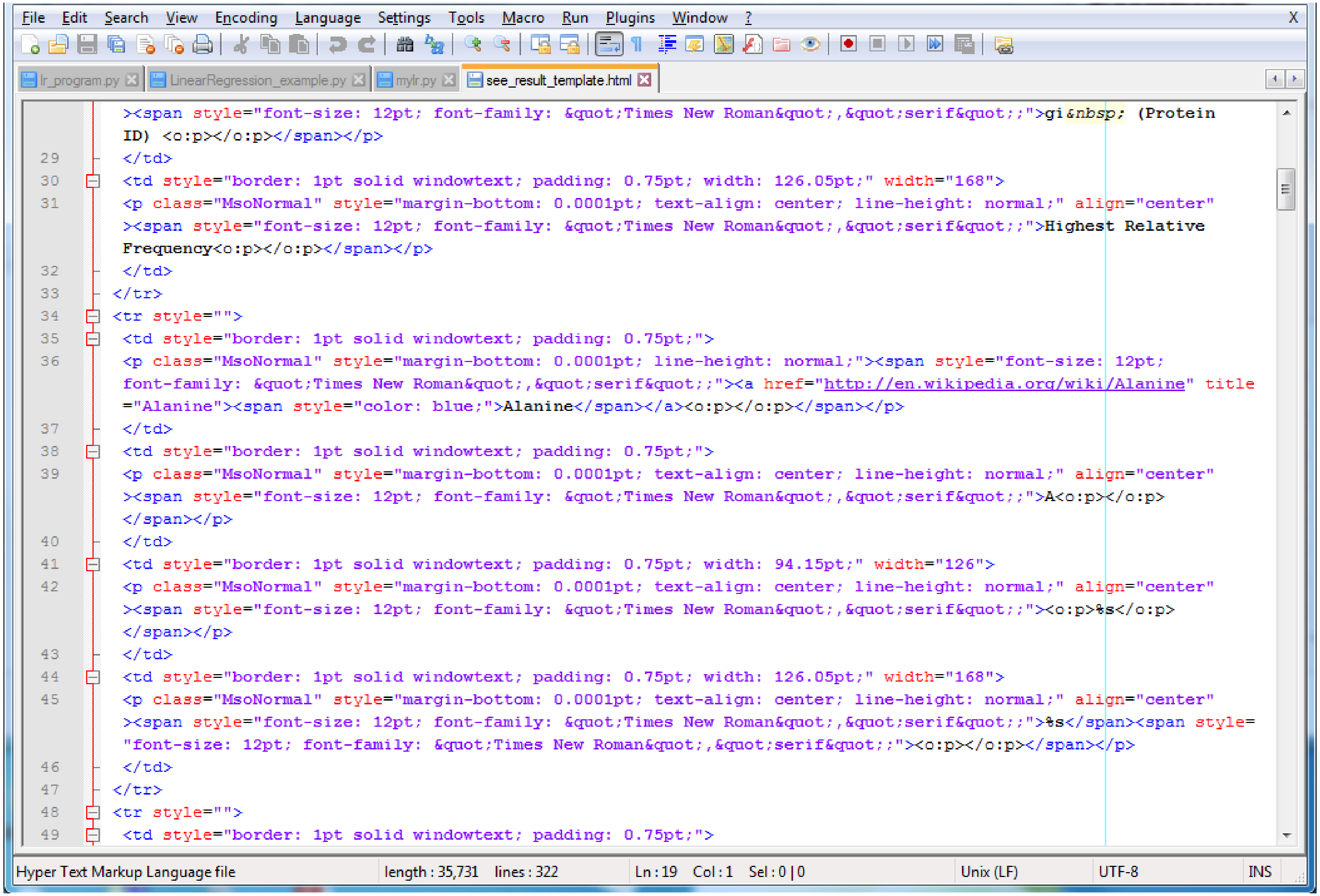
The Database-Driven Webpage Template: the Source View


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
lr_program.py
LinearRegression_example.py
mylr.py
see_result_template.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>



  <meta content="text/html;charset=ISO-8859-1" http-equiv="Content-Type"><title>Form Template</title></head><body>

<h1>Honey Bee Proteins with Highest <br>
</h1>
<h1>Occurance of Each Amino Acid<br>

</h1>


<form action="dumpcgi.cgi" method="get" enctype="multipart/form-data"><span style="font-style: italic;">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; <br>
  </span>

  <table class="MsoNormalTable" style="border: 1pt solid windowtext;" border="1" cellpadding="0">

 <tbody><tr style="">
  <td style="border: 1pt solid windowtext; padding: 0.75pt;">
  <p class="MsoNormal" style="margin-bottom: 0.0001pt; line-height: normal;"><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;">Amino acid<o:p></o:p></span></p>
  </td>
  <td style="border: 1pt solid windowtext; padding: 0.75pt;">
  <p class="MsoNormal" style="margin-bottom: 0.0001pt; text-align: center; line-height: normal;" align="center"><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;">Code<o:p></o:p>
  </span></p>
  </td>
  <td style="border: 1pt solid windowtext; padding: 0.75pt; width: 94.15pt;" width="126">
  <p class="MsoNormal" style="margin-bottom: 0.0001pt; text-align: center; line-height: normal;" align="center"><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;">gi&nbsp; (Protein
```

Hyper Text Markup Language file
length : 35,731 lines : 322
Ln : 19 Col : 1 Sel : 0 | 0
Unix (LF)
UTF-8
INS

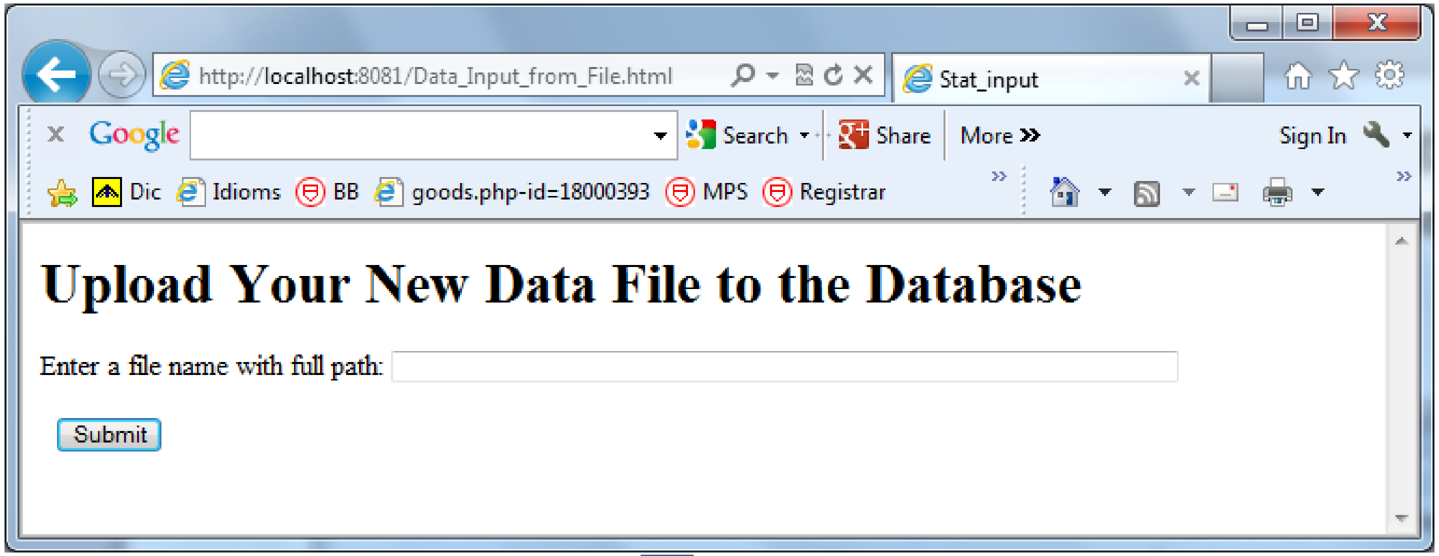
The Database-Driven Webpage Template: the Source View (cont'd)



```
    ><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;">gi&nbsp; (Protein
    ID) <o:p></o:p></span></p>
    </td>
    <td style="border: 1pt solid windowtext; padding: 0.75pt; width: 126.05pt;" width="168">
    <p class="MsoNormal" style="margin-bottom: 0.0001pt; text-align: center; line-height: normal;" align="center"
    ><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;">Highest Relative
    Frequency<o:p></o:p></span></p>
    </td>
  </tr>
  <tr style="">
    <td style="border: 1pt solid windowtext; padding: 0.75pt;">
    <p class="MsoNormal" style="margin-bottom: 0.0001pt; line-height: normal;"><span style="font-size: 12pt;
    font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;"><a href="http://en.wikipedia.org/wiki/Alanine" title
    ="Alanine"><span style="color: blue;">Alanine</span></a><o:p></o:p></span></p>
    </td>
    <td style="border: 1pt solid windowtext; padding: 0.75pt;">
    <p class="MsoNormal" style="margin-bottom: 0.0001pt; text-align: center; line-height: normal;" align="center"
    ><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;">A<o:p></o:p>
    </span></p>
    </td>
    <td style="border: 1pt solid windowtext; padding: 0.75pt; width: 94.15pt;" width="126">
    <p class="MsoNormal" style="margin-bottom: 0.0001pt; text-align: center; line-height: normal;" align="center"
    ><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;"><o:p>%s</o:p>
    </span></p>
    </td>
    <td style="border: 1pt solid windowtext; padding: 0.75pt; width: 126.05pt;" width="168">
    <p class="MsoNormal" style="margin-bottom: 0.0001pt; text-align: center; line-height: normal;" align="center"
    ><span style="font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;">%s</span><span style=
    "font-size: 12pt; font-family: &quot;Times New Roman&quot;,&quot;serif&quot;;"><o:p></o:p></span></p>
    </td>
  </tr>
  <tr style="">
    <td style="border: 1pt solid windowtext; padding: 0.75pt;">
```

Hyper Text Markup Language file | length : 35,731 lines : 322 | Ln : 19 Col : 1 Sel : 0 | 0 | Unix (LF) | UTF-8 | INS

User uploads a data file for processing



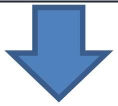
http://localhost:8081/Data_Input_from_File.html

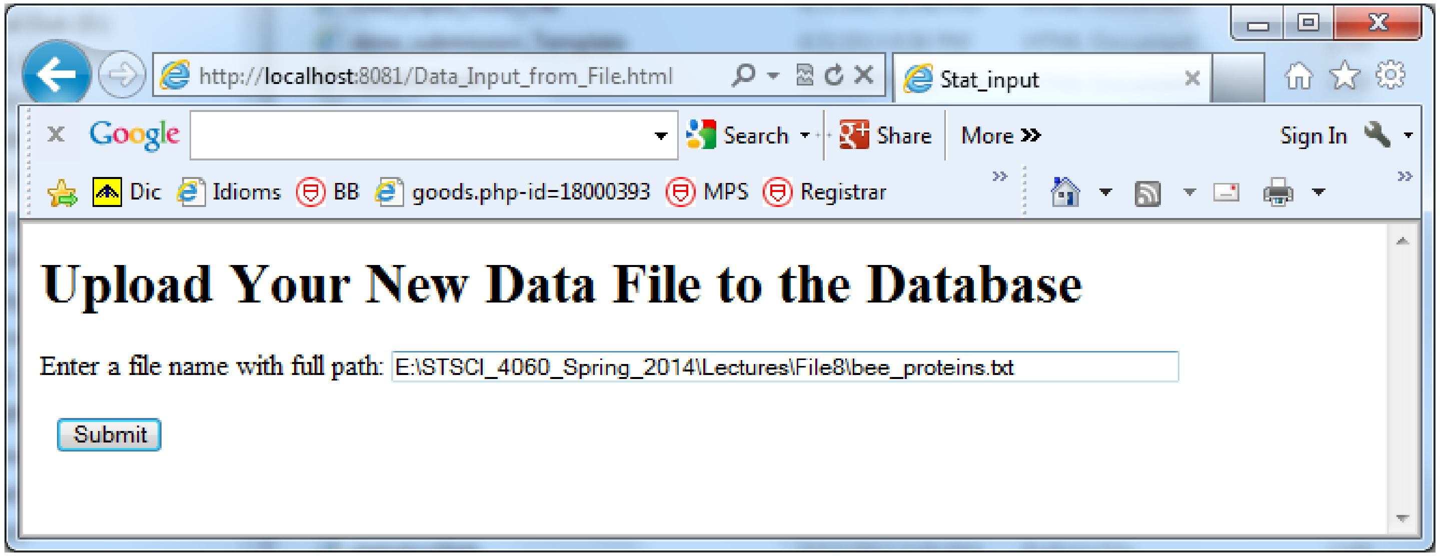
Stat_input

Upload Your New Data File to the Database

Enter a file name with full path:

Submit





http://localhost:8081/Data_Input_from_File.html

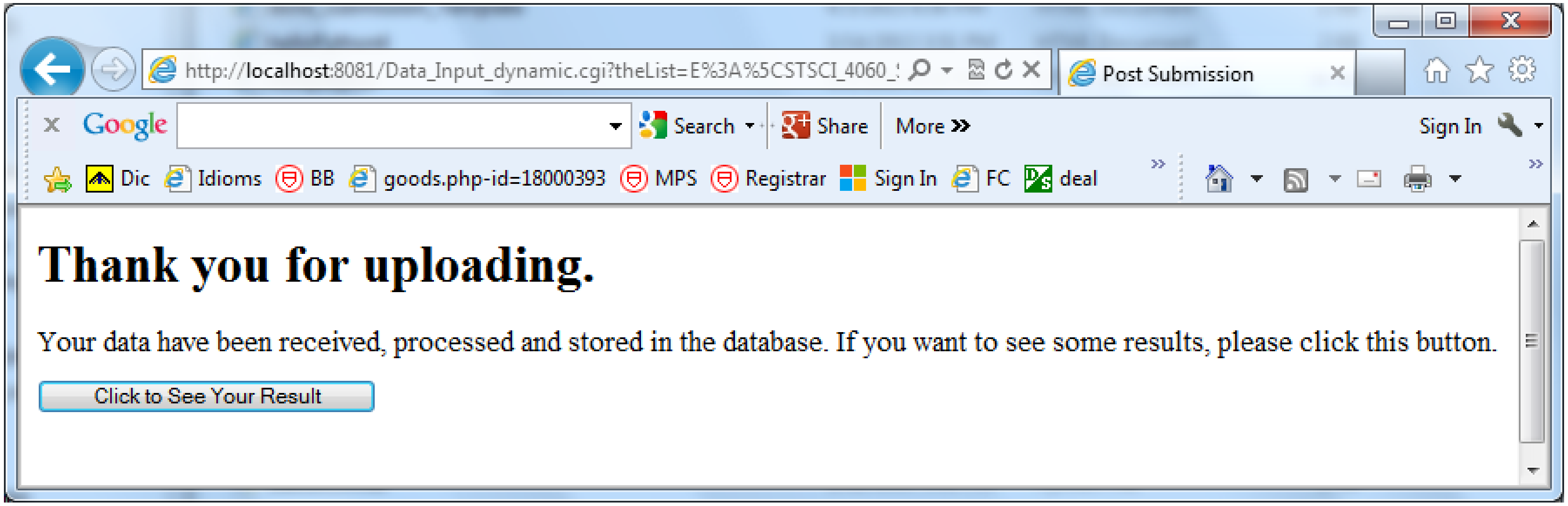
Stat_input

Upload Your New Data File to the Database

Enter a file name with full path: E:\STSCI_4060_Spring_2014\Lectures\File8\bee_proteins.txt

Submit

Feedback: Successful data uploading and processing

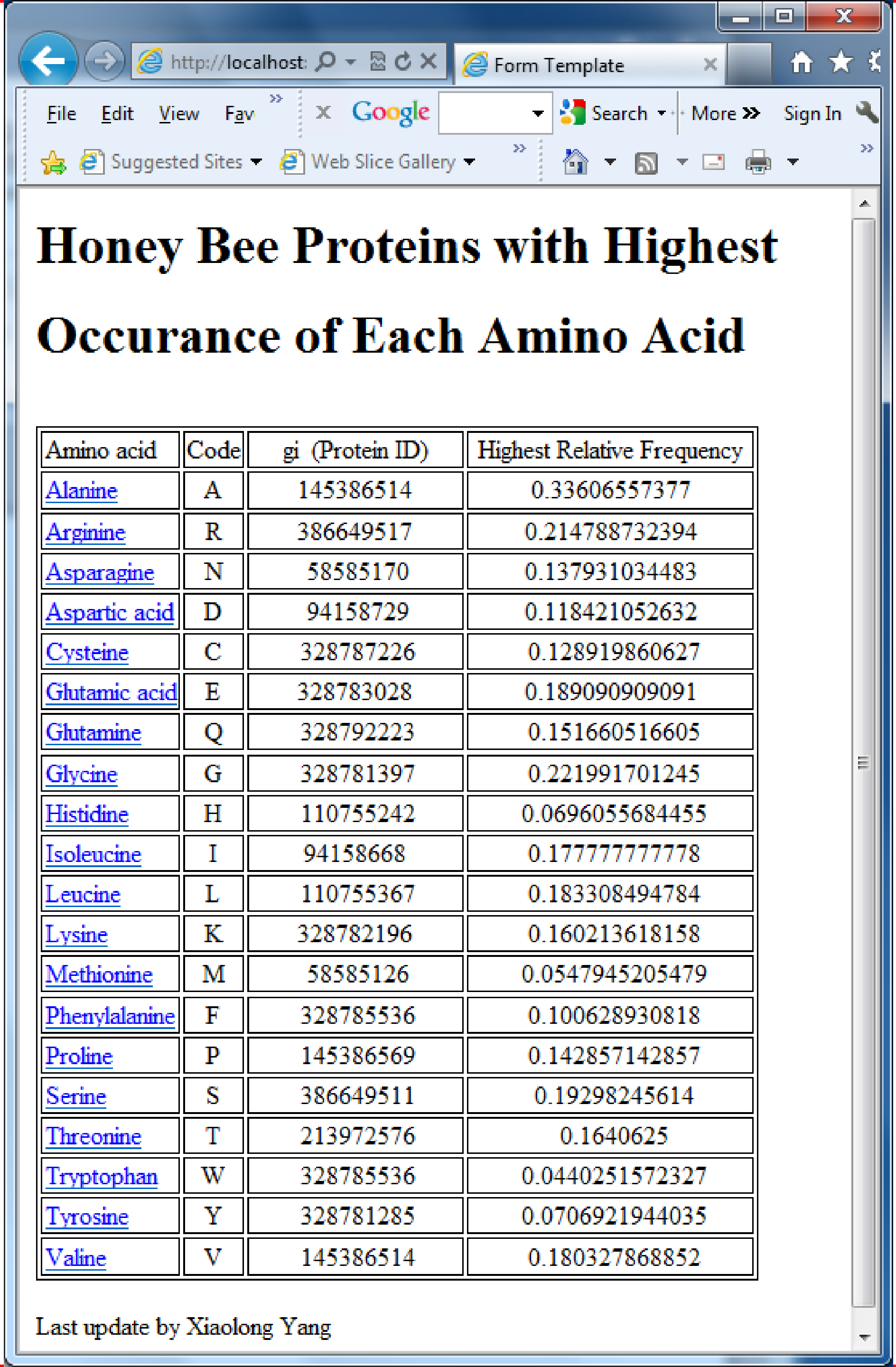


Thank you for uploading.

Your data have been received, processed and stored in the database. If you want to see some results, please click this button.

Click to See Your Result

The Result of Database-driven, Python-control Webpage



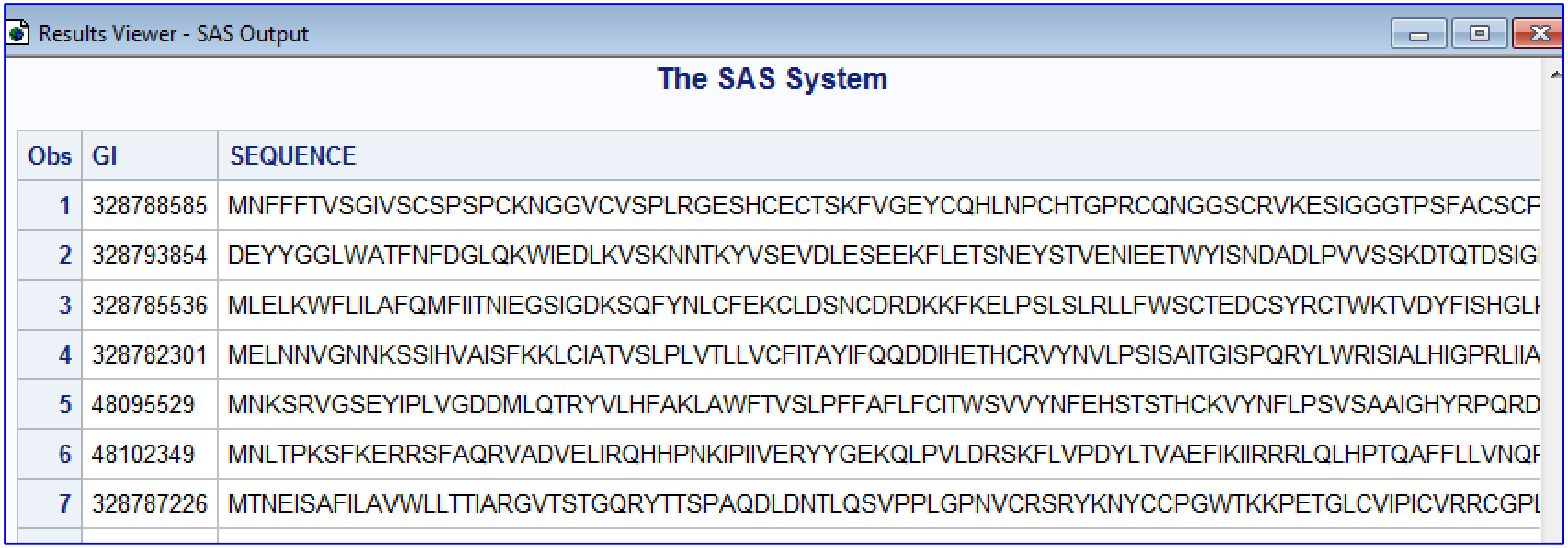
Honey Bee Proteins with Highest Occurance of Each Amino Acid

Amino acid	Code	gi (Protein ID)	Highest Relative Frequency
Alanine	A	145386514	0.33606557377
Arginine	R	386649517	0.214788732394
Asparagine	N	58585170	0.137931034483
Aspartic acid	D	94158729	0.118421052632
Cysteine	C	328787226	0.128919860627
Glutamic acid	E	328783028	0.189090909091
Glutamine	Q	328792223	0.151660516605
Glycine	G	328781397	0.221991701245
Histidine	H	110755242	0.0696055684455
Isoleucine	I	94158668	0.177777777778
Leucine	L	110755367	0.183308494784
Lysine	K	328782196	0.160213618158
Methionine	M	58585126	0.0547945205479
Phenylalanine	F	328785536	0.100628930818
Proline	P	145386569	0.142857142857
Serine	S	386649511	0.19298245614
Threonine	T	213972576	0.1640625
Tryptophan	W	328785536	0.0440251572327
Tyrosine	Y	328781285	0.0706921944035
Valine	V	145386514	0.180327868852

Last update by Xiaolong Yang

Analyze the Data Uploaded, Processed and Stored in the Oracle Database: Display a DB Table in SAS

```
libname beeprot oracle user='python'
password='welcome'
path = 'xe';
proc print data=beeprot.beeproteins;
run;
```



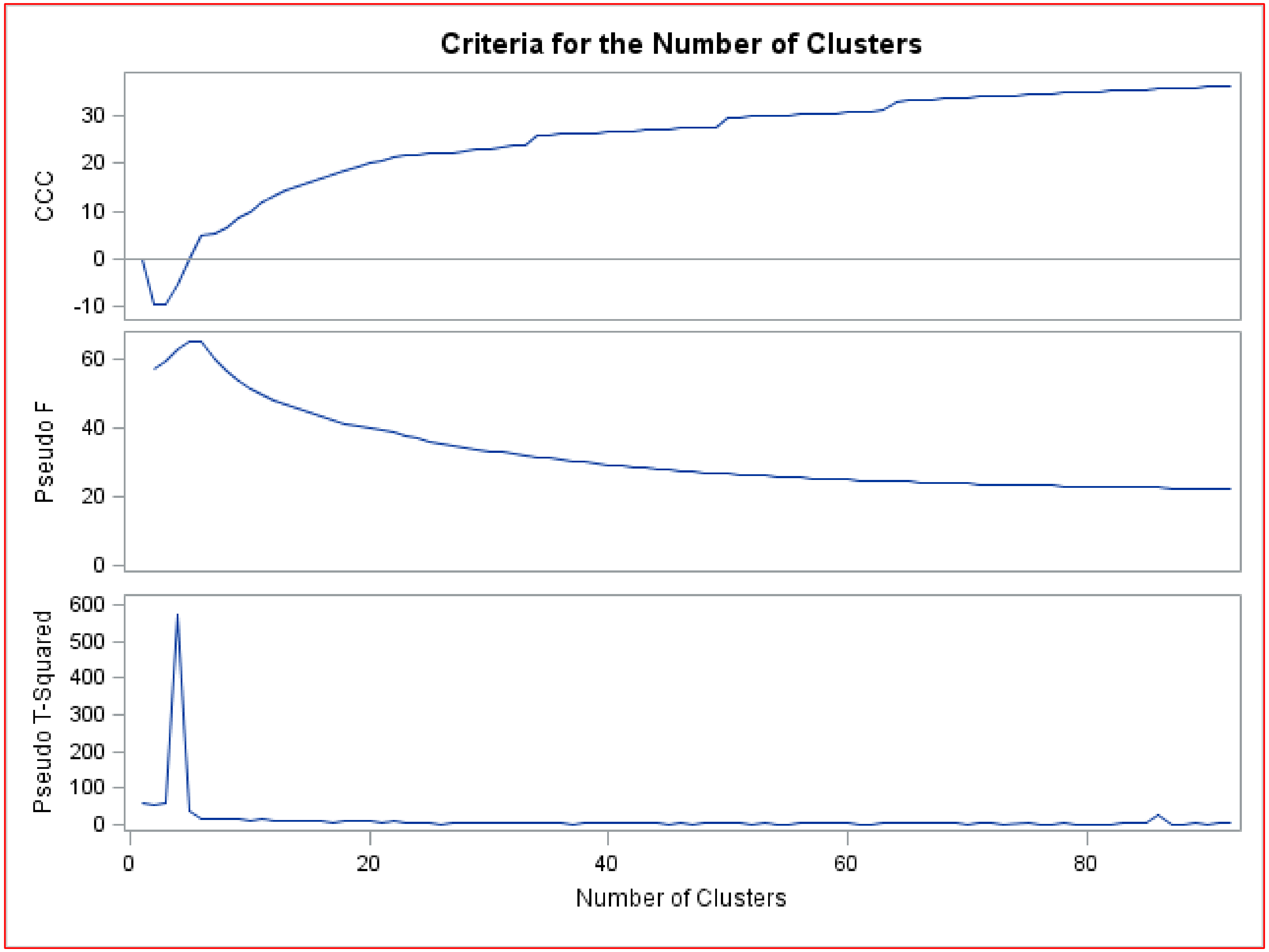
The SAS System

Obs	GI	SEQUENCE
1	328788585	MNFFFTVSGIVSCSPSPCKNGGVCVSPLRGESHCECTSKFVGEYCQHLNPCHTGPRCQNGGSCRVKESIGGGTPSFACSCF
2	328793854	DEYYGGLWATFNFDGLQKWIEDLKVSKNNTKYVSEVDLESEEKFLETSNEYSTVENIEETWYISNDADLPVVSSKDTQTDSIG
3	328785536	MLELKWFLILAFQMFIITNIEGSIGDKSQFYNLCFEKCLDSNCDRDKKFKELPSLSLRLLFWSCTEDCSYRCTWKTVDYFISHGL
4	328782301	MELNNVGNNKSSIHVAISFKKLCIATVSLPLVTLLVCFITAYIFQQDDIHETHCRVYNVLPSISAITGISPQRYLWRISIALHIGPRLIIA
5	48095529	MNKSRVGSEYIPLVGDDMLQTRYVLHFAKLAWFTVSLPFFAFLFCITWSVVYNFEHSTSTHCKVYNFLPSVSAAIGHYRPQRD
6	48102349	MNLTPKSFKERRSFAQRVADVELIRQHHPNKIPIIVERYYGEKQLPVLDRSKFLVPDYLTVAEFIKIIRRRLQLHPTQAFFLLVNQ
7	328787226	MTNEISAFILAVWLLTTIARGVTSTGQRYTTSPAQDLDNTLQSVPPLGPNVCRSRYKNYCCPGWTKKPETGLCVIPICVRRCGPL

Analyze the Data Uploaded, Processed and Stored in the Oracle Database: Cluster Analysis in SAS

```
libname beeprot oracle user='python'
password='welcome'
path = 'xe';

proc cluster data=beeprot.beeproteins
method=ward ccc pseudo
outtree=tree;
var freq_A freq_R freq_N freq_D freq_C
    freq_Q freq_E freq_G freq_H freq_I
    freq_L freq_K freq_M freq_F freq_P
    freq_S freq_T freq_W freq_Y freq_V;
run;
```

Diagrams of CCC, PST2 and PSF

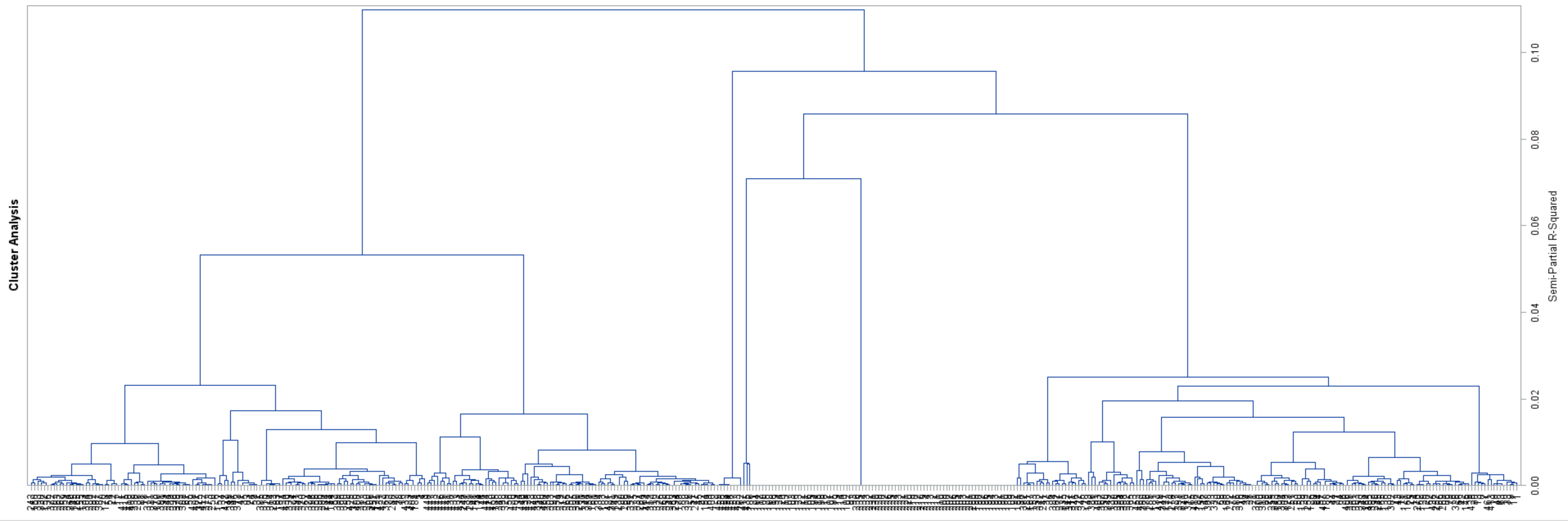


Cluster Analysis Results: Values of CCC, PST2 and PSF

Cluster History

Number of Clusters	Clusters Joined		Freq	Semipartial R-Square	R-Square	Approximate Expected R-Square	Cubic Clustering Criterion	Pseudo F Statistic	Pseudo t-Squared	Tie
461	OB21	OB25	2	0.0000	1.00	.	.	.	.	T
460	OB93	OB97	2	0.0000	1.00	.	.	.	.	T
9	CL12	CL17	119	0.0195	.487	.440	8.63	53.7	14.0	
8	CL48	CL9	134	0.0229	.464	.426	6.47	56.1	15.4	
7	CL10	CL19	123	0.0232	.441	.409	5.46	59.7	18.1	
6	CL8	CL23	156	0.0250	.416	.389	4.72	64.8	15.5	
5	CL11	CL7	213	0.0532	.362	.363	-.14	**64.9**	**37.7**	
4	CL357	CL26	85	0.0708	.291	.329	-5.4	62.8	574	
3	CL6	CL4	241	0.0859	.206	.280	-9.7	59.4	57.8	
2	CL3	CL86	249	0.0957	.110	.177	-9.4	56.8	53.3	
1	CL2	CL5	462	0.1099	.000	.000	0.00	.	56.8	

The Dendrogram


Cluster Analysis
Semi-Partial R-Squared