**python及其生物信息学应用**

**实践报告**

|  |  |
| --- | --- |
| **班级** | **生信1801** |
| **姓名** | **张彦哲** |
| **学号** | **8305180710** |
| **实验名称** | **程序控制结构和mRNA翻译** |
| **日期** | **1/8/2021** |

**实践四：程序控制结构和mRNA翻译**

**实践目的：**

1）理解和掌握程序的控制结构（if、for、while和random语句）及其实现方法。

2）理解和掌握文档读取和写入文件方法。

3）理解和编写程序进行简单的生物数据分析。

4）理解和编写程序将RNA序列翻译为相应的蛋白质序列。

**实践内容：**1）掌握if语句的使用方法。

2）掌握for语句的使用方法。

3）掌握while语句的使用方法。

4）掌握random库的使用方法。

5）使用python进行读取和写入文件。

6）使用python进行简单的生物数据分析。

7）使用python将RNA序列翻译为相应的蛋白质序列。

**1）掌握if语句的使用方法。**

**内容：**使用Python编写程序计算个人所得税。

|  |  |  |
| --- | --- | --- |
| **应纳税所得额** | **税率** | **速算扣除** |
| 1～3000 | 3% | 0 |
| 3001～12000 | 10% | 210 |
| 12001～25000 | 20% | 1410 |
| 25001～35000 | 25% | 2660 |
| 35001～55000 | 30% | 4410 |
| 55001～80000 | 35% | 7160 |
| 80000以上 | 45% | 15160 |

根据上述表格中的税率，编写如下程序计算个人所得税：

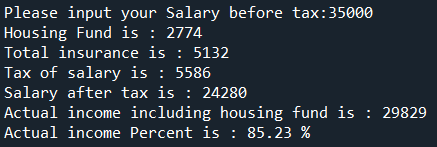


**注：请计算月薪35000元需要交多少个人所得税？（请附上计算结果）**

源码：

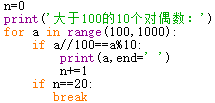
1. **def** taxRate(base):
2. **if** base < 0:
3. tax = 0
4. **elif** base <= 1500:
5. tax = base \* 0.03
6. **elif** base > 1500 **and** base <= 4500:
7. tax = base \* 0.1 - 105
8. **elif** base > 4500 **and** base <= 9000:
9. tax = base \* 0.2 - 555
10. **elif** base > 9000 **and** base <= 35000:
11. tax = base \* 0.25 - 1005
12. **elif** base > 35000 **and** base <= 55000:
13. tax = base \* 0.3 - 2755
14. **elif** base > 55000 **and** base <= 80000:
15. tax = base \* 0.35 - 5505
16. **elif** base > 80000:
17. tax = base \* 0.45 - 13505
18. **return** tax
19. **def** salaryAfterTax(salaryBeforeTax):
20. #免征点3500 ,个人五险一金比率，养老8%，医疗2%，失业0.2%，公积金12%
21. threshold = 3500
22. oldAgeRating = 0.08
23. medicalRating = 0.02
24. unemployRating = 0.002
25. housingFundRating = 0.12
26. # 2016年社平工资7706，五险一金上限是社评三倍工资
27. averageSalary = 7706
28. tripleAverageSalary = 3 \* averageSalary
29. **if** salaryBeforeTax < tripleAverageSalary:
30. totalInsurance = salaryBeforeTax \* (oldAgeRating + medicalRating + unemployRating + housingFundRating)
31. housingFund = salaryBeforeTax \* housingFundRating
32. **else**:
33. totalInsurance = tripleAverageSalary \* (
34. oldAgeRating + medicalRating + unemployRating + housingFundRating)
35. housingFund = tripleAverageSalary \* housingFundRating   #公积金封顶
36. #housingFund = salaryBeforeTax \* housingFundRating     #公司给补超额公积金部分
37. # 纳税额
38. payment = salaryBeforeTax - totalInsurance - threshold
39. tax = taxRate(payment)
40. # 税后工资
41. salaryAfterTax = salaryBeforeTax - totalInsurance - tax
42. actualIncome = salaryAfterTax + housingFund \* 2
43. **print**('Housing Fund is : %d' % housingFund)
44. **print**('Total insurance is : %d ' % totalInsurance)
45. **print**('Tax of salary is : %d' % tax)
46. **print**('Salary after tax is : %d' % salaryAfterTax)
47. **print**('Actual income including housing fund is : %d ' % actualIncome)
48. **print**('Actual income Percent is : %.2f %%' % float(actualIncome \* 100 / salaryBeforeTax))
49. **return** salaryAfterTax
50. **if** \_\_name\_\_ == '\_\_main\_\_':
51. salary = int(input('Please input your Salary before tax:'))
52. salaryAfterTax(salary)

结果：

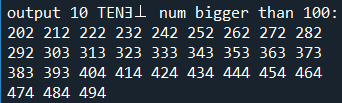


**2）掌握for语句的使用方法。**

**内容：**编写程序输出大于100的20个对偶数。具体程序如下：

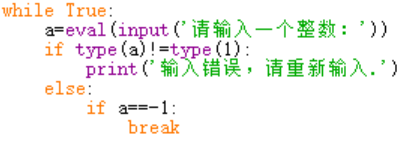


**注：请输出大于200-1000的对偶数30个。**

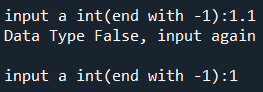


**3）掌握while语句的使用方法。**

**内容：**编写程序输入一个整数，输入非整数时给出错误或者终止运行。具体程序如下：



**注：请附上运行结果。**

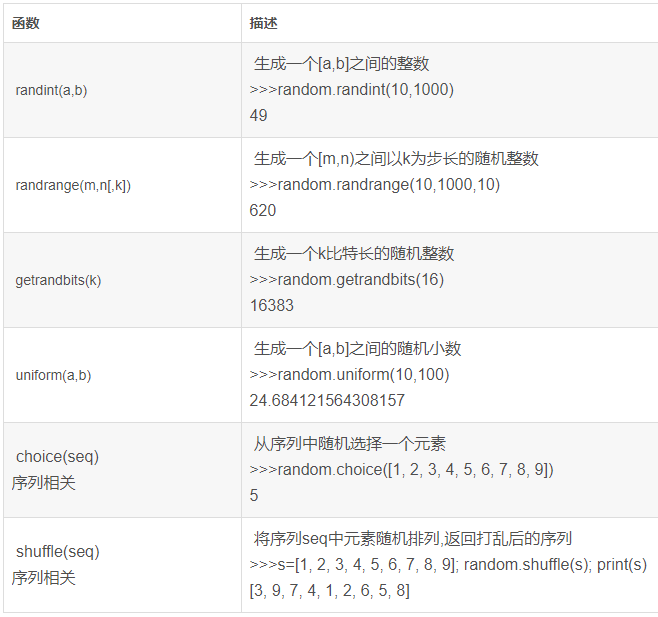


4）**掌握random库的使用方法**

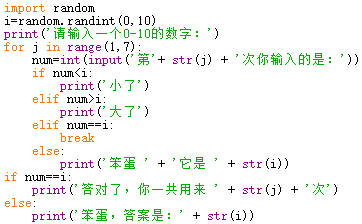
**内容：**使用random库编写一个猜数字的程序。

random库函数包括：基本随机函数和扩展随机函数等





具体程序如下：

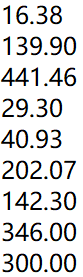
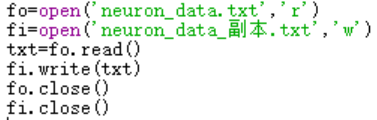


**注：请附上运行结果。**



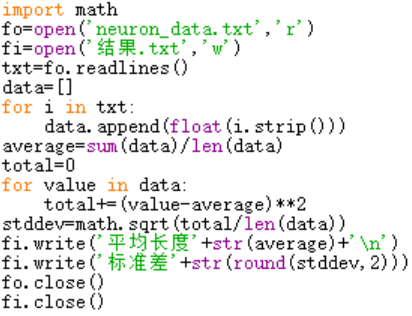
**5）使用Python进行读取和写入文件。**

内容：使用Python编写程序读取文件（**neuron\_data.txt**）中的内容（**文件内容如下**）写入文件（**neuron\_data副本.txt**）。具体程序如下：

**6）使用Python进行简单的生物数据分析。**

**内容：**使用Python编写程序求取神经树突长度数据的均值和标准差。具体程序如下：



**注：运行并输出结果。**

源码：

import numpy

with open('实践4\\data.txt', 'r') as f:

    list\_temp=list(map(float, f.readlines()))

    print('Std = ',numpy.std(list\_temp),'average=',numpy.mean(list\_temp))

MATLAB 2019b:

data=importdata('neuron\_data.txt');

avg=mean(data(:))

stdm=std(data)

结果：

avg =

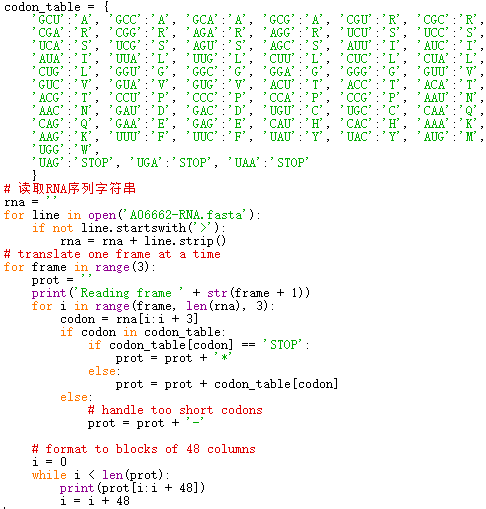
372.8037

stdm =

147.5180

**7）使用Python将RNA序列翻译为相应的蛋白质序列。**

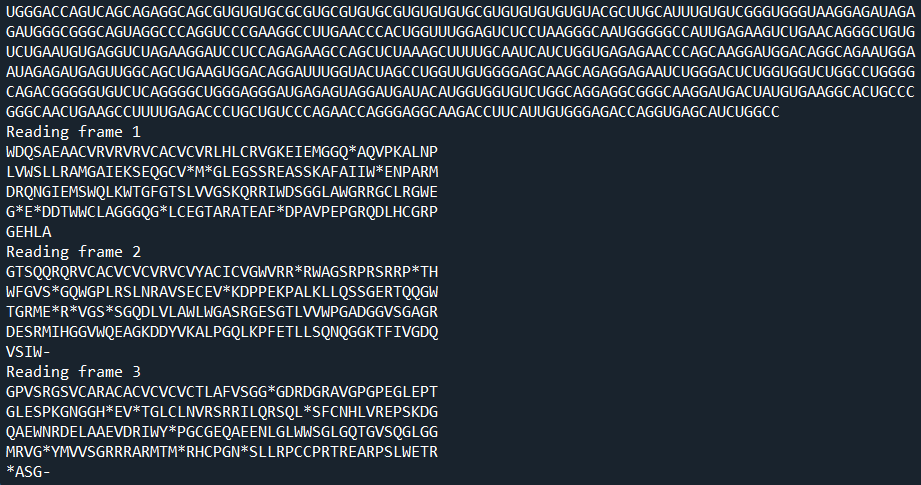
**内容：**将A06662-RNA.fasta中的RNA序列翻译为相应的蛋白质序列。具体程序如下：

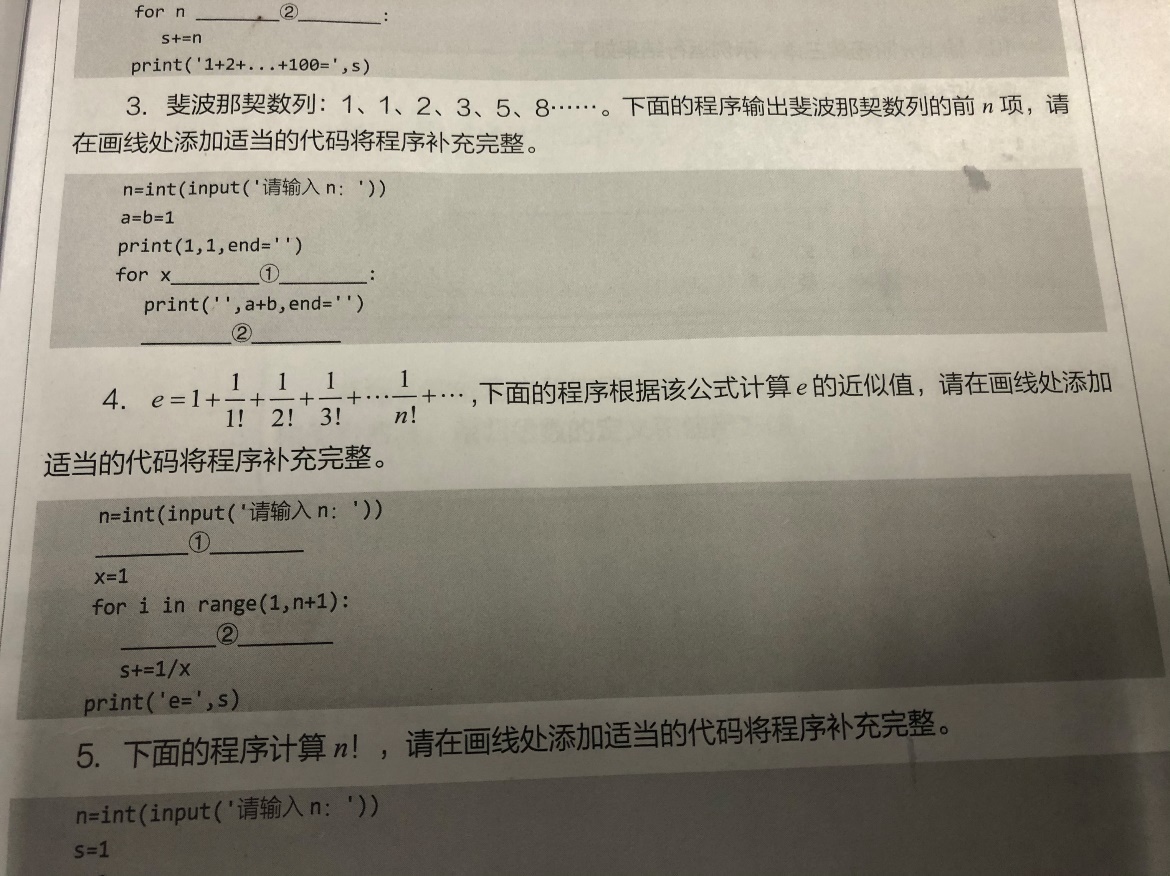


**注：运行并输出结果。**

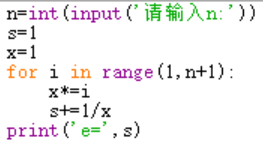
**添加一行：**

1. rna=rna.replace('T', 'U')



**8）练习**

（1）



**注：运行并输出结果。**

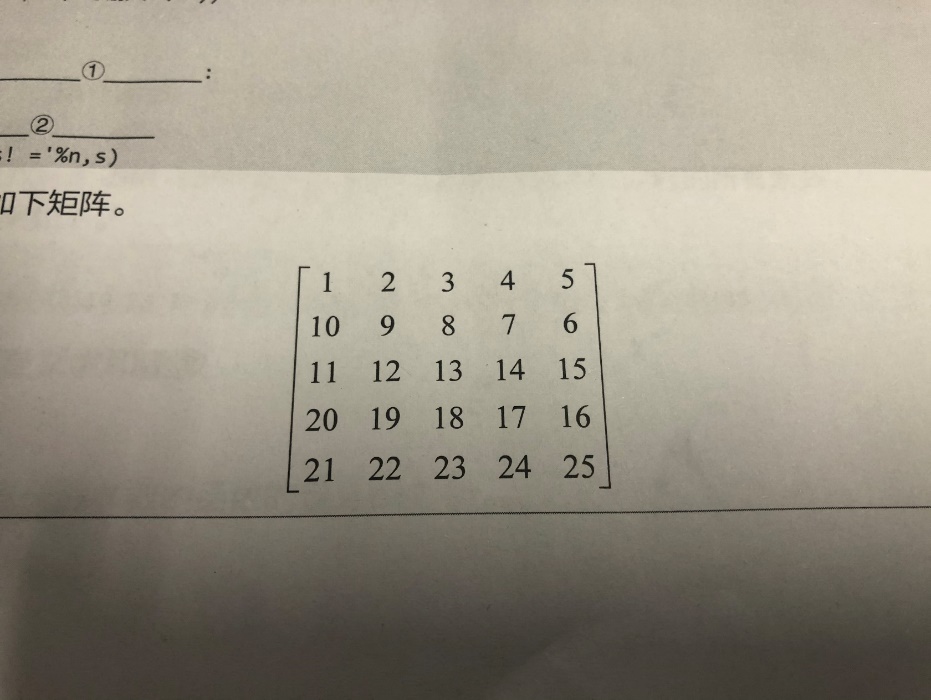
**源码：**

1. **def** factorial(number):
2. **if** number==1:
3. **return** 1
4. **return** number\*factorial(number-1)
6. **def** sum\_toE(n):
7. retv=2.0
8. **for** x **in** range(2,n+1):
9. retv += 1/float(factorial(x))
10. **return** retv
12. **if** \_\_name\_\_ == '\_\_main\_\_':
13. retv = sum\_toE(10)
14. **print**(retv)

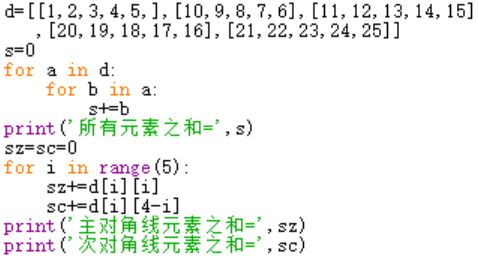
**结果：**



（2）有如下矩阵，请编写程序计算所有元素之和、主对角线元素及次对角线元素之和。



具体程序如下：



**注：运行并输出结果。**

源码：

A=zeros(5);

for i=1:5

for j=1:5

A(i,j)=(i-1)\*5+j;

end

end

sum(sum(A))

sum(diag(A))

sum(diag(A(1:end,end:-1:1)))

结果：

ans =

325

ans =

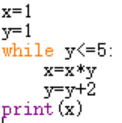
65

ans =

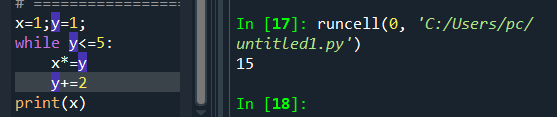
65

**9）作业**

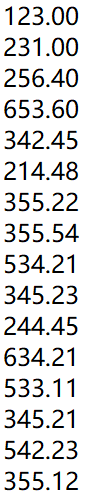
**题目一**：下面代码输出的结果是（C）.



A, 1 B, 10 C, 15 D, 20



**题目二：**新建一个txt文件，在文档中手动输入下边图片中的数值，将文档命名为“神经树突长度数据.txt”，请编写程序读取“神经树突长度数据.txt”中的数值写入到“树突长度统计.txt”文件中，并且计算这些数据的平均值和标准差。



**注：编写程序并运行输出结果。**

源码：（和上述题目一样）

data=importdata('data.txt');

avg=mean(data(:))

stdm=std(data)

结果：

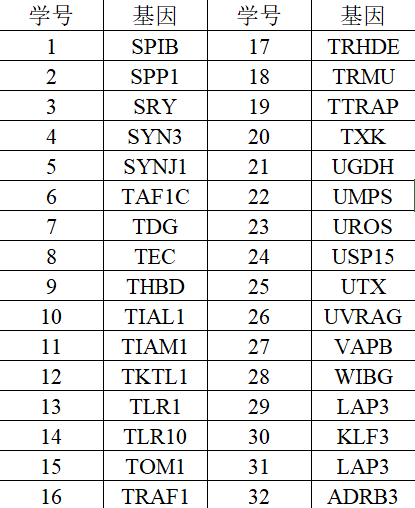
avg =

372.8037

stdm =

147.5180

**题目三：**编写程序将基因的对应RNA序列翻译为相应的蛋白质序列？**（按对应学号选取基因）**

****

**注：编写程序并运行输出结果。**

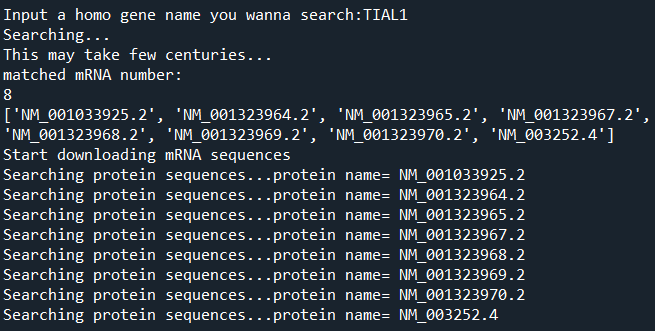
使用爬虫，爬取对应基因全部mRNA，自动下载其fasta格式文件，自动翻译成预测蛋白序列，保存为txt文件

源码：

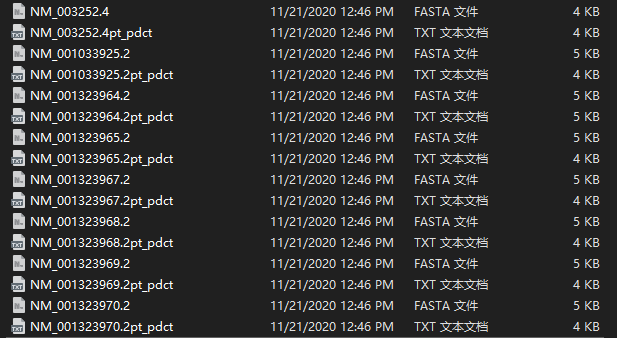
1. **import** requests
2. **import** re
3. **import** os
4. **from** bs4 **import** BeautifulSoup
6. #get mRNA list
7. sch\_gene\_name=input('Input a homo gene name you wanna search:')
8. sch\_url='https://www.ncbi.nlm.nih.gov/gene?term=(homo%5BOrganism%5D)%20AND%20'+sch\_gene\_name+'%5BGene%20Name%5D#reference-sequences'
9. **print**('Searching...\nThis may take few centuries...')
10. res = requests.get(sch\_url)
11. res.encoding='gbk'
12. soup = BeautifulSoup(res.text,"html.parser")
13. match\_mRNAs\_url = re.findall(r'/nuccore/NM\_[0-9]+.[0-9]?',res.text)
14. **print**('matched mRNA number: ')
15. **print**(len(match\_mRNAs\_url))
16. mRNA\_name=[]
17. **for** i **in** range(len(match\_mRNAs\_url)):
18. mRNA\_name.append(re.search(r'NM\_\S+', match\_mRNAs\_url[i]).group())
19. **print**(mRNA\_name)
20. mRNA\_uid=[]
22. codon\_table = {
23. 'GCU':'A', 'GCC':'A', 'GCA':'A', 'GCG':'A', 'CGU':'R', 'CGC':'R',
24. 'CGA':'R', 'CGG':'R', 'AGA':'R', 'AGG':'R', 'UCU':'S', 'UCC':'S',
25. 'UCA':'S', 'UCG':'S', 'AGU':'S', 'AGC':'S', 'AUU':'I', 'AUC':'I',
26. 'AUA':'I', 'UUA':'L', 'UUG':'L', 'CUU':'L', 'CUC':'L', 'CUA':'L',
27. 'CUG':'L', 'GGU':'G', 'GGC':'G', 'GGA':'G', 'GGG':'G', 'GUU':'V',
28. 'GUC':'V', 'GUA':'V', 'GUG':'V', 'ACU':'T', 'ACC':'T', 'ACA':'T',
29. 'ACG':'T', 'CCU':'P', 'CCC':'P', 'CCA':'P', 'CCG':'P', 'AAU':'N',
30. 'AAC':'N', 'GAU':'D', 'GAC':'D', 'UGU':'C', 'UGC':'C', 'CAA':'Q',
31. 'CAG':'Q', 'GAA':'E', 'GAG':'E', 'CAU':'H', 'CAC':'H', 'AAA':'K',
32. 'AAG':'K', 'UUU':'F', 'UUC':'F', 'UAU':'Y', 'UAC':'Y', 'AUG':'M',
33. 'UGG':'W',
34. 'UAG':'STOP', 'UGA':'STOP', 'UAA':'STOP'
35. }
37. #downloading mRNA fasta
38. **print**('Start downloading mRNA sequences')
39. **for** t **in** range(len(match\_mRNAs\_url)):
40. **print**('Searching protein sequences...protein name= '+mRNA\_name[t])
41. mRNA\_sch\_url='https://www.ncbi.nlm.nih.gov'+match\_mRNAs\_url[t]
42. mRNA\_res = requests.get(mRNA\_sch\_url)
43. mRNA\_res.encoding='gbk'
44. match\_mRNAs = re.findall(r'<span class="ff\_line" id="'+mRNA\_name[t]+'\_[0-9]+">\S+</span>',mRNA\_res.text)
45. match\_mRNAs=(re.search(r'(?<=<meta name="ncbi\_uidlist" content=")[0-9]+(?=" />)',mRNA\_res.text)).group()
46. mRNA\_uid.append(match\_mRNAs)
47. mRNA\_sch\_url='https://www.ncbi.nlm.nih.gov/sviewer/viewer.cgi?tool=portal&save=file&log$=seqview&db=nuccore&report=fasta&id='+mRNA\_uid[t]+'&conwithfeat=on&withparts=on&hide-cdd=on'
48. mRNA\_fasta = requests.get(mRNA\_sch\_url).text
50. #storing mRNA sequences
51. fh = open(os.getcwd()+'\\'+mRNA\_name[t]+'.fasta', 'w', encoding='utf-8')
52. fh.write(mRNA\_fasta)
53. fh.close()
54. mRNA\_seq=''
55. mRNA\_trans\_predict=[]
57. #predict translated sequences and store it
58. **for** line **in** open(os.getcwd()+'\\'+mRNA\_name[t]+'.fasta'):
59. **if** **not** line.startswith('>'):
60. mRNA\_seq = mRNA\_seq + line.strip()
61. mRNA\_seq=mRNA\_seq.replace('T', 'U')
62. **for** frame **in** range(3):
63. prot = ''
64. **for** i **in** range(frame, len(mRNA\_seq), 3):
65. codon = mRNA\_seq[i:(i + 3)]
66. **if** codon **in** codon\_table:
67. **if** codon\_table[codon] == 'STOP':
68. prot = prot + '\*'
69. **else**:
70. prot = prot + codon\_table[codon]
71. **else**:
72. prot = prot + '-'
73. fp=open(os.getcwd()+'\\'+mRNA\_name[t]+'pt\_pdct.txt', 'a')
74. **if**(fp.writable()):
75. fp.write('\nReading frame ' + str(frame + 1)+'\n'+prot)
76. fp.close()

结果：

Console：



文件管理器：



由于有8组mRNA，篇幅有限，此处仅以NM\_001323970.2为例：

mRNA序列：

1. >NM\_001323970.2 Homo sapiens TIA1 cytotoxic granule associated RNA binding protein like 1 (TIAL1), transcript variant 8, mRNA
2. GCAGAAGTGCTTCCCTCTCTCATTTCCAGGACCACAATTCCCAGAGACTTCGGCTTCACGACGTTTCTCT
3. TTTTGCCCGATCTCTCCCGGAGCTGGCTGGGCTTCGGCCGGCCAGAGGCCCACAGCGACGACGTGATCCG
4. TCGTGAGCGGGTCCCAGGGTTTCCTCGGCGGCCCTTTTTTCTCTCCCCTCGGTCGTCCCCCTTGCAGGCT
5. GTCGCGGCTGGCCTGAGCGGTGACCTGGCGGGTCGCGCCTGCGCTCTGCCCTGTTTCCTGCCTGGCTGGT
6. GGCGGCGGCCATTTTGTTCATCCTCCTCCTCCTCCTGCTCCTCCTGGTTGGAGCGCAGTGTCCGGAGCGG
7. GCTGGGGGGAGAGAGCCCGAGAGCAGGGTTCGGTGCCTTTTCCTCTGTCCCCAGCCGGTGCCCAGAGCCC
8. CCCTCCCCTTCCTCCCCACCCCCTCCCCTCCCCAACCCTGCCCTCCCCCTTGTCCCGGGATCGCTCCGTC
9. GCACCCACCATGATGGAAGACGACGGGCAGCCCCGGACTCTACACACACAAGCTAAGACATATGGTTCTA
10. TCTTCAGAAAACAATATGGTGTTTAGAGCCTAGATACGTAGGTAACCTTTCCAGAGATGTGACAGAAGTC
11. CTTATACTTCAGTTGTTCAGTCAGATTGGACCCTGTAAAAGCTGTAAAATGATAACAGAGCAACCCGATA
12. GCAGAAGGGTCAACTCTTCTGTTGGATTTTCTGTTTTGCAGCATACAAGCAATGACCCATATTGCTTTGT
13. GGAATTTTATGAACACAGAGATGCAGCTGCTGCATTAGCTGCTATGAATGGGAGAAAAATTTTGGGAAAG
14. GAGGTCAAAGTAAACTGGGCAACCACACCAAGTAGCCAGAAAAAAGATACTTCCAATCACTTCCATGTGT
15. TTGTTGGGGATTTGAGTCCAGAAATTACAACAGAAGATATCAAATCAGCATTTGCCCCCTTTGGTAAAAT
16. ATCGGATGCCCGGGTAGTTAAAGACATGGCAACTGGAAAATCCAAAGGCTATGGTTTTGTATCTTTTTAT
17. AACAAACTGGATGCAGAAAATGCGATTGTGCATATGGGCGGTCAGTGGTTGGGTGGTCGTCAAATCCGAA
18. CCAATTGGGCCACTCGTAAACCACCTGCACCTAAAAGTACACAAGAAAACAACACTAAGCAGTTGAGATT
19. TGAAGATGTAGTAAACCAGTCAAGTCCAAAAAATTGTACTGTGTACTGTGGAGGAATTGCGTCTGGGTTA
20. ACAGATCAGCTTATGAGACAGACATTCTCACCATTTGGACAAATTATGGAAATAAGAGTTTTCCCAGAAA
21. AGGGCTATTCATTTGTCAGATTTTCAACCCATGAAAGTGCAGCCCATGCCATTGTTTCGGTGAACGGTAC
22. TACGATTGAAGGACATGTGGTTAAATGCTATTGGGGTAAAGAATCTCCTGATATGACTAAAAACTTCCAA
23. CAGGTTGACTATAGTCAATGGGGCCAATGGAGCCAAGTGTATGGAAACCCACAACAGTATGGACAGTATA
24. TGGCAAATGGGTGGCAAGTACCGCCTTATGGAGTATACGGGCAACCATGGAATCAACAAGGATTTGGAGT
25. AGATCAATCACCTTCTGCTGCTTGGATGGGTGGATTTGGTGCTCAGCCTCCCCAAGGACAAGCTCCTCCC
26. CCTGTAATACCTCCTCCTAACCAAGCCGGATATGGTATGGCAAGTTACCAAACACAGTGAGCCGGGACTC
27. TAAAAAAAAATTGTAATTCATGATAGGCTTCGATTTCCTGTGACACTCTGAAGACATGAAAGTAGACATC
28. GGAAAATGAAAATATTTATTTTAAAAATTGAAATGTTTGGAACCTTTAGCACAGATTTGCTTTGGTGAAG
29. GACACGTGTCTTCTAGTTCTGCCTTTTTAAGTTTTTGTTCATGATGGATATGAACATGATTTTTCTTTAT
30. GTACAAAAACTAAAATAAAGTCAATAAAGACAATTCTGACTACAAATTTTGATATAATAGGAAAAATGGC
31. TAATACATTTTGATTCTTAGATACTATTCCATTTTTATCTTGCTGTTCAGTATTTTAACTCACTGTGTTT
32. TTAAAAGAGCAAAAAAGGGAGGATCGTGAAAACCTGGGAATCACATATAAGTTCATCCTGAATCCTGATA
33. CTCCCCTCCCCTTCCCTGAGGTGGACCACATTTGAAGTCAGCAGAGAAAAAGTGTGATATTCAGAAGAAA
34. TGCGTGATTTTGGAGTCGCTTTGGAGGAAATATTTTCTTTCTCTATGCCTAAAGAAACTGAAGCCAGACT
35. GAAGTTTTGCACCCTAAAAAAGGAACAGCATTGTTTGAGTTACTTGAGCAAATGTTGGTGGTCCACGTTA
36. AGACATATTTTTAAAACTTCCAAAAGTGTCGATTATTAAAATTGTAGTATTTTACATTTCATTTTGGGGG
37. GAAATCCAAGTATGGTGTTTGTATTGAAGTCAGACAGTCATACTTGTGCTTTTACATGAAGTTTAAATGA
38. TACATATTGTAAATATTCAATAACTACAGTGTTTAAAAAGCATGCTTCAACATAGAAGTAGCAGCAATGT
39. AATTATTTGAAGTAACACTTAACACACTCCGCTGCATTGAATGCAGTGGATTGATCAGAATGTTAAGACT
40. GACATTTCCAAGGTTGGCTACTATGTAAAATTAAAATTACACAAATTGTGCAGAAAAAGCCTTAATTTTA
41. ATTTCATACAATCTTTGATGCATTAGTATGTTCTAAAATGTCATTGGGAATTAGTTTTTTGTTTTTGTTT
42. TTTTTTTTTTTTTTTTTTGCTTTACATTACTTGGTATGTAAATACCTTGATTAAAACCTTGTAAACCAAT
43. TTCAAGGTTACTATAAGTTGTATAGTACAAGTGTTTTTTAAAAATCTTGGGGTGTTTTTAAAAATTAAGA
44. TATATTTTGCCCAAGAATTTTTTTAACAAGATTGCTAAAAACATCTTATTTAGACACTTCAATGTACCAA
45. TTTATAATTGGATATTCAGTTTAAATAGTACACAGAGTTGTGGCTTTTATTTTCAATTAATTTTTTTCCT
46. TGTGGGCAGTGTGCATGGTATAATAAGCCTGAGCAGAGGCTTAAGTTGTATGTGTGCAGAGTTTGTAAAG
47. GAATCAATTGGAAGATGCAGAAGACCGAGGTTTGCTTTCAAGGTATTTTTCAGGCTGTGTGGGTAAAATT
48. TGCCTCAAATTTCTATCAAACAGGAATGTAAAATAGATAAAATCCTATGTATTTGAATTGTCAGAGCTAG
49. GGAGTGCAAATGTTTTGGCAATGTATTCAAAATGCTGGCCTGGGCACCAAAGAGAAAATAGCCTTTTACA
50. GTTACATAGTAAGATGCGATTAGTACCCACAAATTACTGTTTTCTAAACATTTGAAGTTTTACGATTAGC
51. TTTAAAATAATGATTTTATAAATTGGTGGTCACAATAATTTTGGTATTACTTTCCTCCTTTTCCCACTTA
52. GCAATATAGCCAAATGTATTCAACATAAAAATTCATAGGGTCTGAAATTCATAGCTGGGCCAAATTTTTT
53. ATGGCACCTTAGTTTTACCATAATGGTCATCTATTACACTCTTCTGTTATAAAATATACCCTTATTTCTT
54. TTGTTTATAGTATCTTTGAGGAATGTTTTTGGAAAAGTTAATTTATATTTTATAGGGAGAACACTCAATA
55. AATTATGTTAACTGTGCCCCCGAGTTAAAAATTTTATGAGTATATGTGAAACTTGAACAACTGAAGACTT
56. TTTTTAATTGATAAAAATGCTTAGTATGCCTGTTTTGGTCTGCCAGTAAATTAAGTAGCTTATTGAGATA
57. ACTAACAGCTAAATATAGCTGTAGTGTTTCCTGACTGTATATTCTATGATTTAATAAAATTATCCAGACT
58. AGTTATATTGCCACAGTAAA

顺式翻译蛋白预测的3种结果：

1. **Reading frame 1**
2. AEVLPSLISRTTIPRDFGFTTFLFLPDLSRSWLGFGRPEAHSDDVIRRERVPGFPRRPFFLSPRSSPLQAVAAGLSGDLAGRACALPCFLPGWWRRPFCSSSSSSCSSWLERSVRSGLGGESPRAGFGAFSSVPSRCPEPPSPSSPPPPLPNPALPLVPGSLRRTHHDGRRRAAPDSTHTS\*DIWFYLQKTIWCLEPRYVGNLSRDVTEVLILQLFSQIGPCKSCKMITEQPDSRRVNSSVGFSVLQHTSNDPYCFVEFYEHRDAAAALAAMNGRKILGKEVKVNWATTPSSQKKDTSNHFHVFVGDLSPEITTEDIKSAFAPFGKISDARVVKDMATGKSKGYGFVSFYNKLDAENAIVHMGGQWLGGRQIRTNWATRKPPAPKSTQENNTKQLRFEDVVNQSSPKNCTVYCGGIASGLTDQLMRQTFSPFGQIMEIRVFPEKGYSFVRFSTHESAAHAIVSVNGTTIEGHVVKCYWGKESPDMTKNFQQVDYSQWGQWSQVYGNPQQYGQYMANGWQVPPYGVYGQPWNQQGFGVDQSPSAAWMGGFGAQPPQGQAPPPVIPPPNQAGYGMASYQTQ\*AGTLKKNCNS\*\*ASISCDTLKT\*K\*TSENENIYFKN\*NVWNL\*HRFALVKDTCLLVLPF\*VFVHDGYEHDFSLCTKTKIKSIKTILTTNFDIIGKMANTF\*FLDTIPFLSCCSVF\*LTVFLKEQKREDRENLGITYKFILNPDTPLPFPEVDHI\*SQQRKSVIFRRNA\*FWSRFGGNIFFLYA\*RN\*SQTEVLHPKKGTALFELLEQMLVVHVKTYF\*NFQKCRLLKL\*YFTFHFGGKSKYGVCIEVRQSYLCFYMKFK\*YIL\*IFNNYSV\*KACFNIEVAAM\*LFEVTLNTLRCIECSGLIRMLRLTFPRLATM\*N\*NYTNCAEKALILISYNL\*CISMF\*NVIGN\*FFVFVFFFFFFCFTLLGM\*IP\*LKPCKPISRLL\*VV\*YKCFLKILGCF\*KLRYILPKNFFNKIAKNILFRHFNVPIYNWIFSLNSTQSCGFYFQLIFFLVGSVHGIISLSRGLSCMCAEFVKESIGRCRRPRFAFKVFFRLCG\*NLPQISIKQECKIDKILCI\*IVRARECKCFGNVFKMLAWAPKRK\*PFTVT\*\*DAISTHKLLFSKHLKFYD\*L\*NNDFINWWSQ\*FWYYFPPFPT\*QYSQMYST\*KFIGSEIHSWAKFFMAP\*FYHNGHLLHSSVIKYTLISFVYSIFEECFWKS\*FIFYRENTQ\*IMLTVPPS\*KFYEYM\*NLNN\*RLFLIDKNA\*YACFGLPVN\*VAY\*DN\*QLNIAVVFPDCIFYDLIKLSRLVILPQ\*-
3. **Reading frame 2**
4. QKCFPLSFPGPQFPETSASRRFSFCPISPGAGWASAGQRPTATT\*SVVSGSQGFLGGPFFSPLGRPPCRLSRLA\*AVTWRVAPALCPVSCLAGGGGHFVHPPPPPAPPGWSAVSGAGWGERAREQGSVPFPLSPAGAQSPPPLPPHPLPSPTLPSPLSRDRSVAPTMMEDDGQPRTLHTQAKTYGSIFRKQYGV\*SLDT\*VTFPEM\*QKSLYFSCSVRLDPVKAVK\*\*QSNPIAEGSTLLLDFLFCSIQAMTHIALWNFMNTEMQLLH\*LL\*MGEKFWERRSK\*TGQPHQVARKKILPITSMCLLGI\*VQKLQQKISNQHLPPLVKYRMPG\*LKTWQLENPKAMVLYLFITNWMQKMRLCIWAVSGWVVVKSEPIGPLVNHLHLKVHKKTTLSS\*DLKM\*\*TSQVQKIVLCTVEELRLG\*QISL\*DRHSHHLDKLWK\*EFSQKRAIHLSDFQPMKVQPMPLFR\*TVLRLKDMWLNAIGVKNLLI\*LKTSNRLTIVNGANGAKCMETHNSMDSIWQMGGKYRLMEYTGNHGINKDLE\*INHLLLLGWVDLVLSLPKDKLLPL\*YLLLTKPDMVWQVTKHSEPGL\*KKIVIHDRLRFPVTL\*RHESRHRKMKIFILKIEMFGTFSTDLLW\*RTRVF\*FCLFKFLFMMDMNMIFLYVQKLK\*SQ\*RQF\*LQILI\*\*EKWLIHFDS\*ILFHFYLAVQYFNSLCF\*KSKKGRIVKTWESHISSS\*ILILPSPSLRWTTFEVSREKV\*YSEEMRDFGVALEEIFSFSMPKETEARLKFCTLKKEQHCLSYLSKCWWSTLRHIFKTSKSVDY\*NCSILHFILGGNPSMVFVLKSDSHTCAFT\*SLNDTYCKYSITTVFKKHAST\*K\*QQCNYLK\*HLTHSAALNAVD\*SEC\*D\*HFQGWLLCKIKITQIVQKKP\*F\*FHTIFDALVCSKMSLGISFLFLFFFFFFFALHYLVCKYLD\*NLVNQFQGYYKLYSTSVF\*KSWGVFKN\*DIFCPRIFLTRLLKTSYLDTSMYQFIIGYSV\*IVHRVVAFIFN\*FFSLWAVCMV\*\*A\*AEA\*VVCVQSL\*RNQLEDAEDRGLLSRYFSGCVGKICLKFLSNRNVK\*IKSYVFELSELGSANVLAMYSKCWPGHQRENSLLQLHSKMRLVPTNYCFLNI\*SFTISFKIMIL\*IGGHNNFGITFLLFPLSNIAKCIQHKNS\*GLKFIAGPNFLWHLSFTIMVIYYTLLL\*NIPLFLLFIVSLRNVFGKVNLYFIGRTLNKLC\*LCPRVKNFMSICET\*TTEDFF\*LIKMLSMPVLVCQ\*IK\*LIEITNS\*I\*L\*CFLTVYSMI\*\*NYPD\*LYCHSK
5. **Reading frame 3**
6. RSASLSHFQDHNSQRLRLHDVSLFARSLPELAGLRPARGPQRRRDPS\*AGPRVSSAALFSLPSVVPLAGCRGWPER\*PGGSRLRSALFPAWLVAAAILFILLLLLLLLVGAQCPERAGGREPESRVRCLFLCPQPVPRAPLPFLPTPSPPQPCPPPCPGIAPSHPP\*WKTTGSPGLYTHKLRHMVLSSENNMVFRA\*IRR\*PFQRCDRSPYTSVVQSDWTL\*KL\*NDNRATR\*QKGQLFCWIFCFAAYKQ\*PILLCGIL\*TQRCSCCISCYEWEKNFGKGGQSKLGNHTK\*PEKRYFQSLPCVCWGFESRNYNRRYQISICPLW\*NIGCPGS\*RHGNWKIQRLWFCIFL\*QTGCRKCDCAYGRSVVGWSSNPNQLGHS\*TTCT\*KYTRKQH\*AVEI\*RCSKPVKSKKLYCVLWRNCVWVNRSAYETDILTIWTNYGNKSFPRKGLFICQIFNP\*KCSPCHCFGERYYD\*RTCG\*MLLG\*RIS\*YD\*KLPTG\*L\*SMGPMEPSVWKPTTVWTVYGKWVASTALWSIRATMESTRIWSRSITFCCLDGWIWCSASPRTSSSPCNTSS\*PSRIWYGKLPNTVSRDSKKKL\*FMIGFDFL\*HSEDMKVDIGK\*KYLF\*KLKCLEPLAQICFGEGHVSSSSAFLSFCS\*WI\*T\*FFFMYKN\*NKVNKDNSDYKF\*YNRKNG\*YILILRYYSIFILLFSILTHCVFKRAKKGGS\*KPGNHI\*VHPES\*YSPPLP\*GGPHLKSAEKKCDIQKKCVILESLWRKYFLSLCLKKLKPD\*SFAP\*KRNSIV\*VT\*ANVGGPR\*DIFLKLPKVSIIKIVVFYISFWGEIQVWCLY\*SQTVILVLLHEV\*MIHIVNIQ\*LQCLKSMLQHRSSSNVII\*SNT\*HTPLH\*MQWIDQNVKTDISKVGYYVKLKLHKLCRKSLNFNFIQSLMH\*YVLKCHWELVFCFCFFFFFFLLYITWYVNTLIKTL\*TNFKVTISCIVQVFFKNLGVFLKIKIYFAQEFF\*QDC\*KHLI\*TLQCTNL\*LDIQFK\*YTELWLLFSINFFPCGQCAWYNKPEQRLKLYVCRVCKGINWKMQKTEVCFQGIFQAVWVKFASNFYQTGM\*NR\*NPMYLNCQS\*GVQMFWQCIQNAGLGTKEKIAFYSYIVRCD\*YPQITVF\*TFEVLRLALK\*\*FYKLVVTIILVLLSSFSHLAI\*PNVFNIKIHRV\*NS\*LGQIFYGTLVLP\*WSSITLFCYKIYPYFFCL\*YL\*GMFLEKLIYIL\*GEHSINYVNCAPELKIL\*VYVKLEQLKTFFN\*\*KCLVCLFWSASKLSSLLR\*LTAKYSCSVS\*LYIL\*FNKIIQTSYIATV-

（实际上只需写一个字典和循环，就可以把30个基因全部爬取完）