Jacek Radajewski
Student number: 43612772
BIOL3014 – Advanced Bioinformatics
Practical 0

Table of Contents

Exercise 1 2
- Code 2
- Output 3

Exercise 2 4
- Part 1 4
- Part 2 4
  - Code 4
  - Output 5
- Part 3 5
  - Code 5
  - Output 6

Exercise 3 7
- Output 8

Exercise 4 9
- Code 9
- Output 11

Exercise 5 12
- Code 12
- Output 13

Exercise 6 18
- Full Code 18
- Output 20

Exercise 7 21
- Source Code 21
- Output 23

Exercise 1

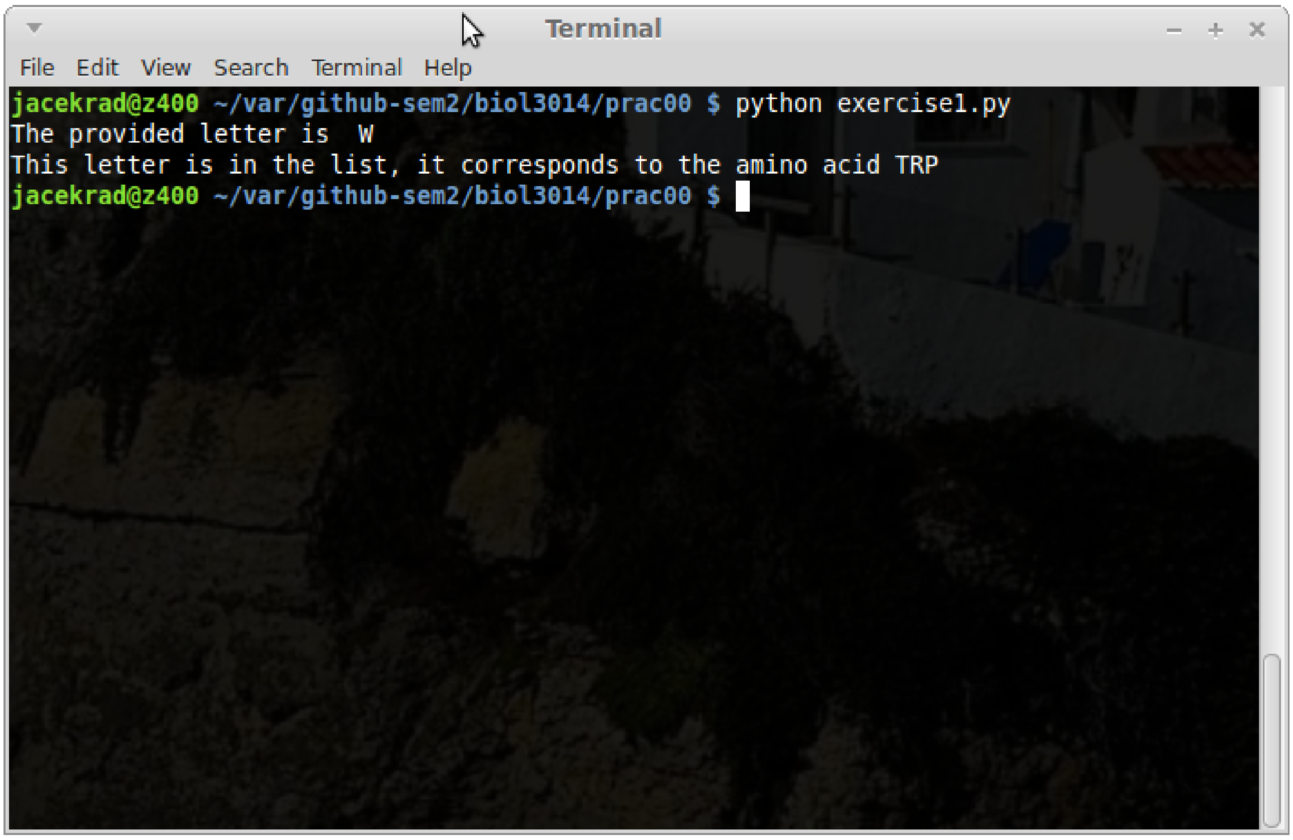
Source code shown below. Output section shows the execution of the program with my_aa='W' and the second with my_aa='Z'.

Code

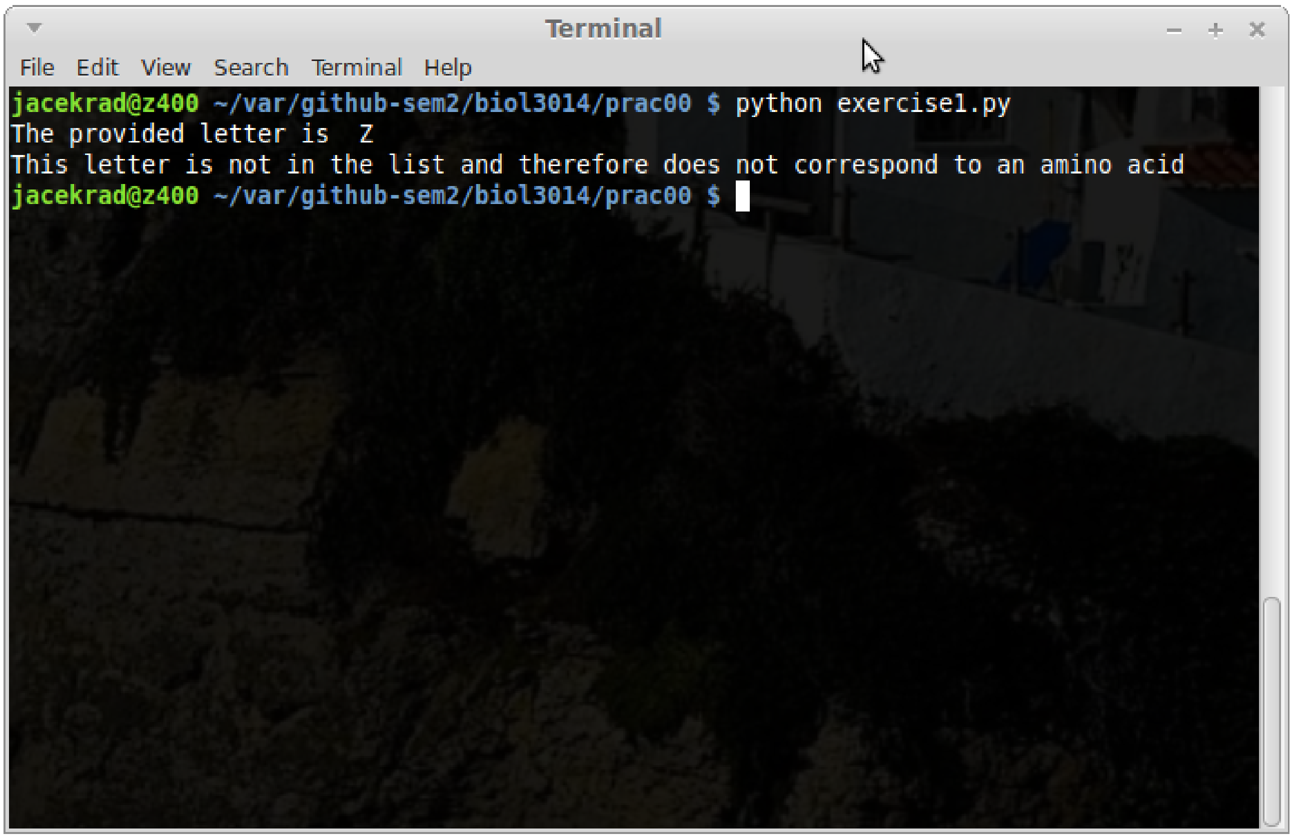
```
'''
Created on 30/07/2014

@author: s4361277
'''

my_aa = "Z"
#my_aa = "W"
my_aacodes = {'Y':'TYR', 'F':'PHE', 'C':'CYS', 'V':'VAL', 'H':'HIS', 'L':'LEU', \
              'D':'ASP', 'A':'ALA', 'S':'SER', 'E':'GLU', 'W':'TRP', 'P':'PRO', \
              'I':'ILE', 'R':'ARG', 'K':'LYS', 'N':'ASN', 'M':'MET', 'T':'THR', \
              'G':'GLY', 'Q':'GLN'}


print "The provided letter is ", my_aa
if my_aa in my_aacodes:
    print "This letter is in the list, it corresponds to the amino acid", \
        my_aacodes.get(my_aa)
else:
    print "This letter is not in the list and therefore does not correspond "\
        + "to an amino acid"
```

Output

Output shown for my_aa set to 'W' and 'Z'



```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ python exercise1.py
The provided letter is  W
This letter is in the list, it corresponds to the amino acid TRP
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $
```



```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ python exercise1.py
The provided letter is  Z
This letter is not in the list and therefore does not correspond to an amino acid
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $
```

Exercise 2

Part 1

When `mySequence = ["M","H","K","L"]` is replaced with `mySequence = "MHKL"` the program executes the same as before because in the first case the line

`for myaa in mySequence:`

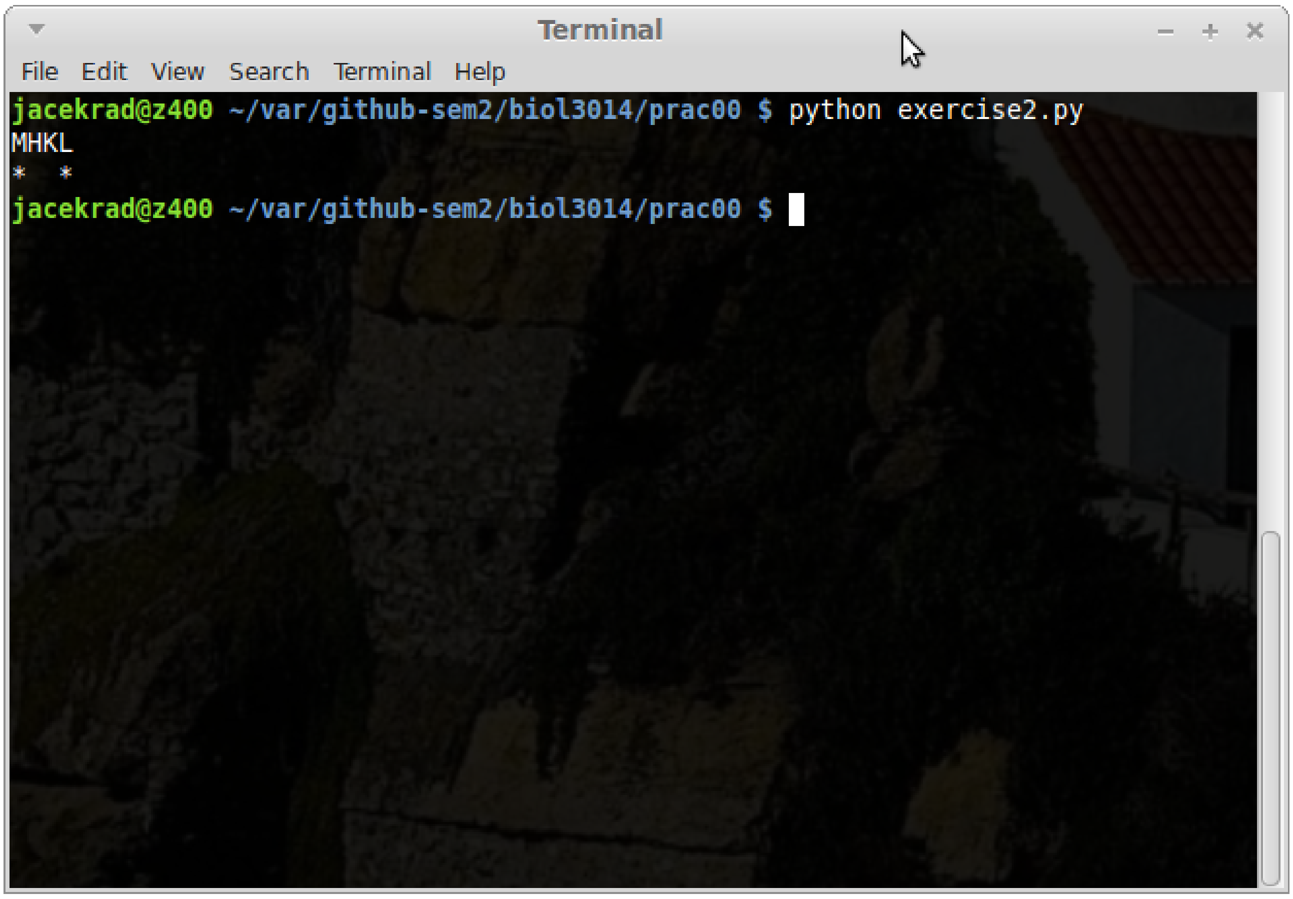
iterates over characters in a list and in the latter case over the same characters in a string.

Part 2

Code

```
'''
Created on 30/07/2014

@author: s4361277
'''
myHydrophobics = "FLIMVPAWG"
mySequence = "MHKL"
mySequenceHydro = ""
for myaa in mySequence:
    #print "The amino acids is", myaa
    if myaa in myHydrophobics:
        mySequenceHydro += "*"
        # print " It is hydrophobic"
    else:
        mySequenceHydro += " "
print mySequence
print mySequenceHydro
```

Output



```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ python exercise2.py
MHKL
*  *
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $
```

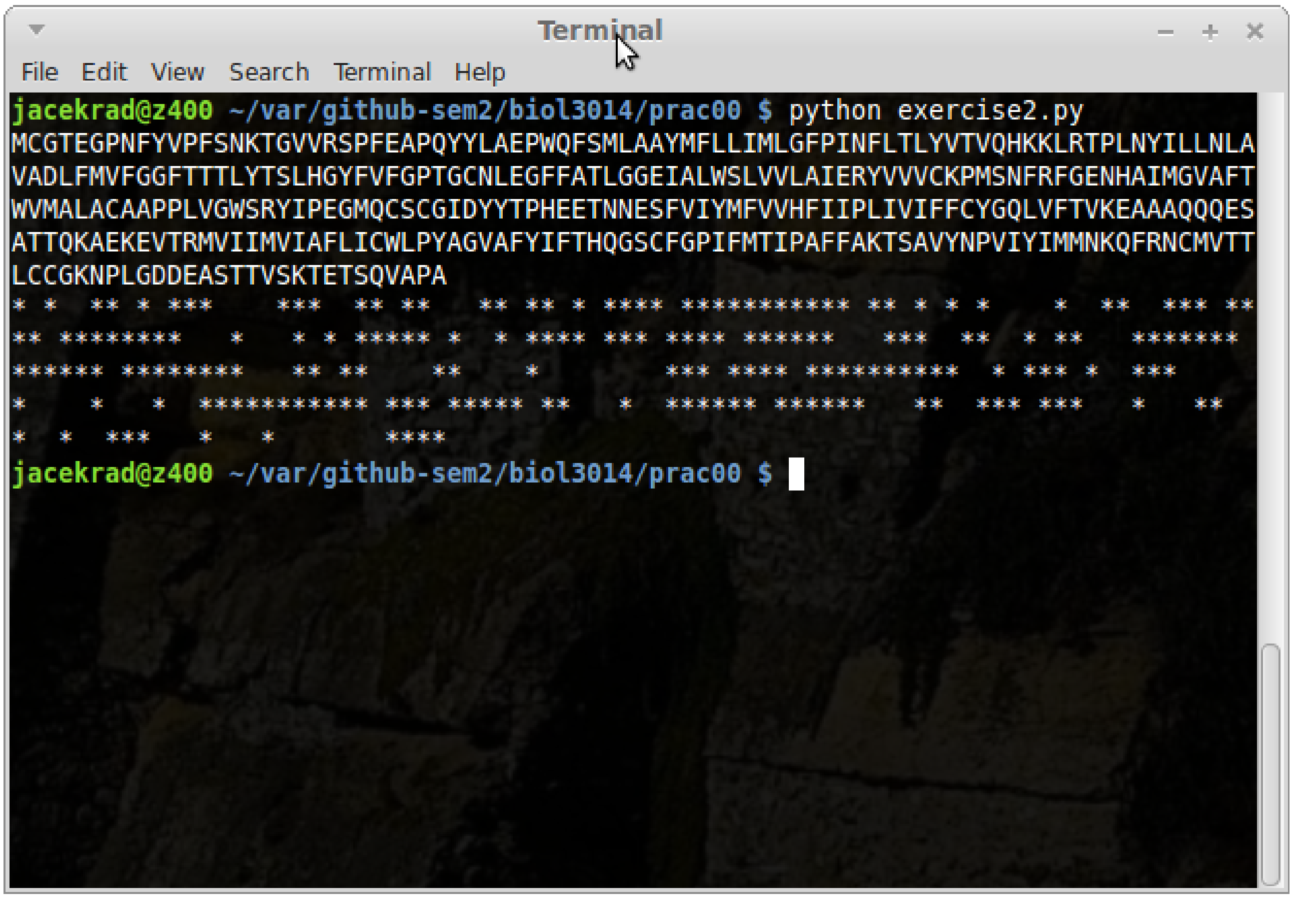
Part 3

Looking at the output it can be seen that the hydrophobic proteins are clustered. This is consistent with a trans membrane protein. Indeed UniProt BLAST search confirms that the residue sequence belongs to Rhodopsin, a multi-pass membrane protein. There are 12 clusters of 5 or more hydrophobic residues visible in the output.

Code

```
'''
Created on 30/07/2014

@author: s4361277
'''
myHydrophobics = "FLIMVPAWG"
mySequence = "MCGTEGPNFYVPFSNKTGVVRSPFEAPQYYLAEPWQFSMLAAYMFLLIMLGFPIN"\
    + "FLTLYVTVQHKKLRTPLNYILLNLAVADLFMVFGGFTTTLYTSLHGYFVFGPTGCNLEGFFA"\
    + "TLGGEIALWSLVVLAIERYVVVCKPMSNFRFGENHAIMGVAFTWVMALACAAPPLVGWSRYI"\
    + "PEGMQCSCGIDYYTPHEETNNESFVIYMFVVHFIIPLIVIFFCYGQLVFTVKEAAAQQQESA"\
    + "TTQKAEKEVTRMVIIMVIAFLICWLPYAGVAFYIFTHQGSCFGPIFMTIPAFFAKTSAVYNP"\
    + "VIYIMMNKQFRNCMVTTLCCGKNPLGDDEASTTVSKTETSQVAPA"
mySequenceHydro = ""
for myaa in mySequence:
    #print "The amino acids is", myaa
    if myaa in myHydrophobics:
        mySequenceHydro += "*"
    else:
        mySequenceHydro += " "
print mySequence
print mySequenceHydro
```

Output



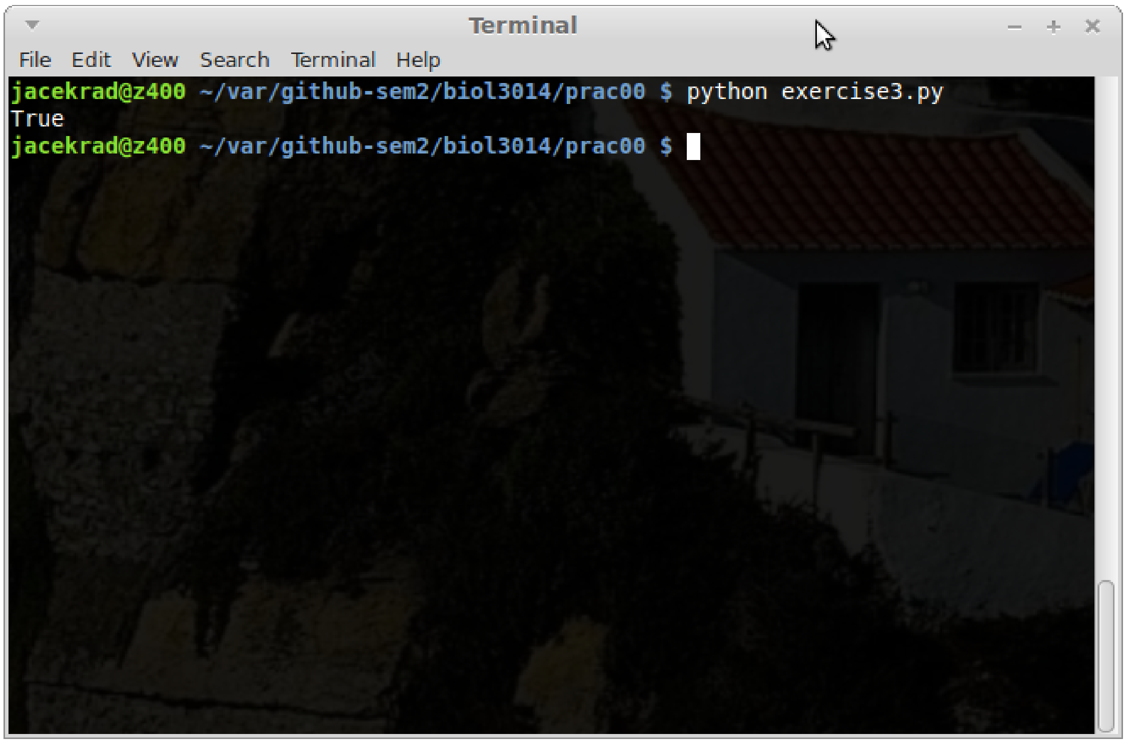
```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ python exercise2.py
MCGTEGPNFYVPFSNKTGVVRSPFEAPQYYLAEPWQFSMLAAYMFLLIMLGFPINFLTLYVTVQHKKLRTPLNYILLNLA
VADLFMVFGGFTTTLYTSLHGYFVFGPTGCNLEGFFATLGGEIALWSLVVLAIERYVVVCKPMSNFRFGENHAIMGVAFT
WVMALACAAPPLVGWSRYIPEGMQCSCGIDYYTPHEETNNESFVIYMFVVHFIIPLIVIFFCYGQLVFTVKEAAAQQQES
ATTQKAEKEVTRMVIIMVIAFLICWLPYAGVAFYIFTHQGSCFGPIFMTIPAFFAKTSAVYNPVIYIMMNKQFRNCMVTT
LCCGKNPLGDDEASTTVSKTETSQVAPA
* *  ** * ***    ***  ** **   ** ** * **** *********** ** * * *    *  **  *** **
** ********   *   * * ***** *  * **** *** **** ******   ***  **  * **   ******* 
****** ********   ** **    **    *        *** **** **********  * *** *  ***
*    *   *  *********** *** ***** **   *  ****** ******   **  *** ***   *   **
*  *  ***   *   *       ****
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ 
```

Exercise 3

The purpose of the myIsDNA(mySeq) function is to determine if mySeq that is passed to the function is a DNA sequence or not. If the sequence is determined to be a DNA sequence then boolean True is returned and boolean False is returned otherwise. The function works by defining a sequence of letters (myNonDNA) which should not appear in a DNA sequence. The function iterates over this myNonDNA sequence and for each letter found it is checked if it appears in mySeq. If a defined nonDNA letter is found in mySeq then the function returns False. If whole sequence was processed and no non-DNA letters matched then True is returned. The two possible outputs are True or False. Documented function shown below:

```
'''
Created on 30/07/2014

@author: s4361277
'''
def myIsDNA (mySeq):
    '''
    This function returns True iff mySeq consists entirely of DNA letters 
or
    more specifically mySeq does not contain any of the following letters:
    QWERYUIOPSDFHJKLZXVBNM.  Function returns False otherwise.
    '''
    # list of non-DNA letters
    myNonDNA = "QWERYUIOPSDFHJKLZXVBNM"

    # iterate over the non-DNA letters
    for myAA in myNonDNA:

        # check if the non-DNA letter is contained in our sequence
        # and if it is then return False
        if myAA in mySeq:
            return False

    # finished iterating over the non-DNA letter and have not returned 
False yet so return True as mySeq IS DNA
    return True

# call the function
print myIsDNA("GTTCGACCA")
```

Output



```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ python exercise3.py
True
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ 
```

Exercise 4

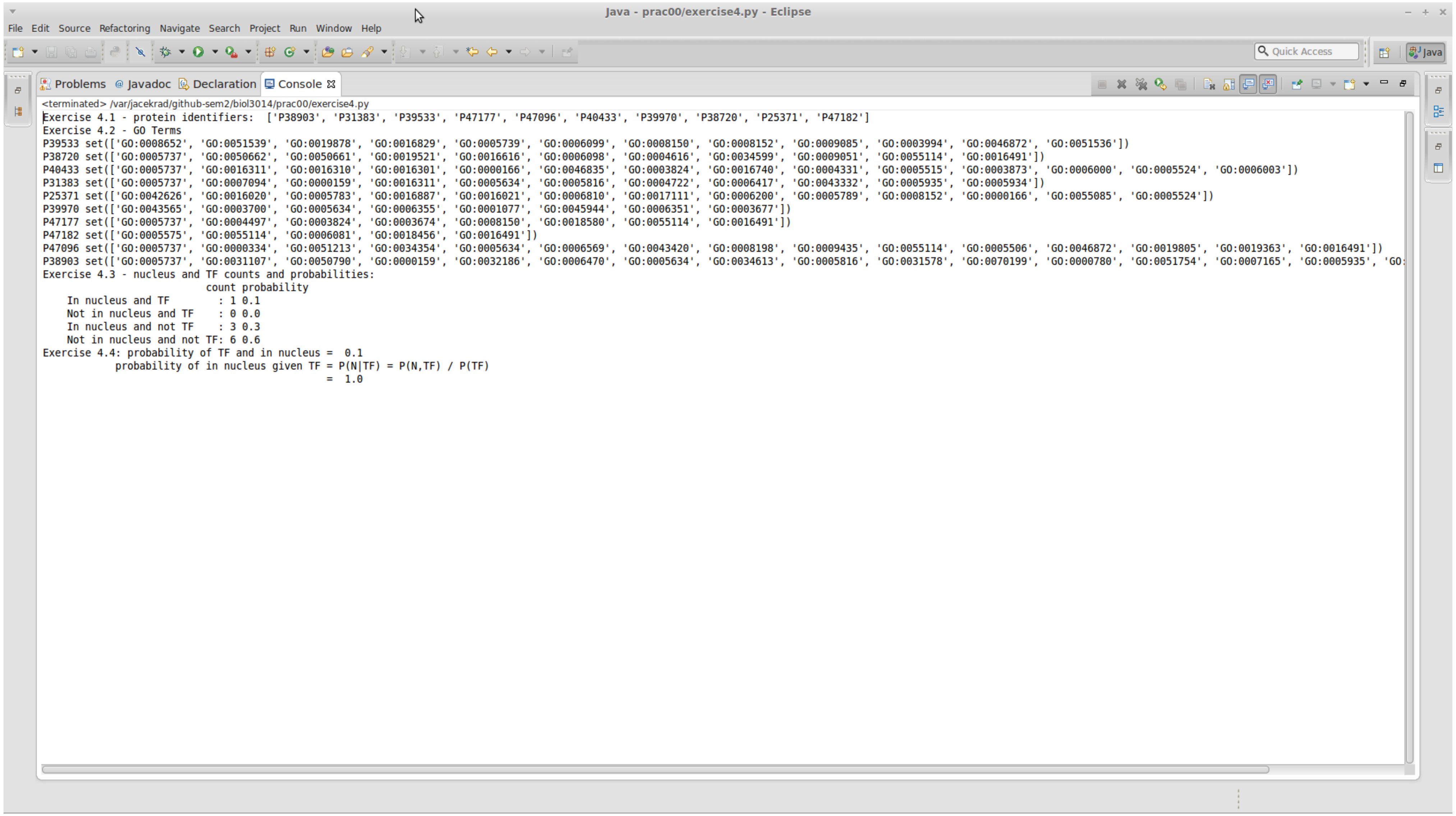
Code is shown in the following section and the output in the one after that. Each part of the question is marked in the output.

Code

```
'''
Created on 02/08/2014

@author: jacekrad
'''
from webservice import getGOTerms

yeast_file = open("yeast_transcriptome_10.txt")
protein_identifiers = [] # list of protein IDs
go_terms_dict = {} # dictionary of go_terms with the gene ID as the key

# total number of proteins read from file
protein_count = 0

# counters for TF and nucleus proteins
tf_nucleus_count = 0
not_tf_nucleus_count = 0
tf_not_nucleus_count = 0
not_tf_not_nucleus_count = 0

for line in yeast_file:
    protein_count += 1
    fields = line.strip().split("\t")
    protein_id = fields[0]
    protein_identifiers.append(protein_id)
    go_terms = getGOTerms(fields[0])
    go_terms_dict[protein_id] = go_terms
    if "GO:0005634" in go_terms: # check if nucleus
        if "GO:0003700" in go_terms: # check if TF
            tf_nucleus_count += 1
        else:
            not_tf_nucleus_count += 1
    else:
        if "GO:0003700" in go_terms: # check if TF
            tf_not_nucleus_count += 1
        else:
            not_tf_not_nucleus_count += 1
yeast_file.close()

print "Exercise 4.1 - protein identifiers: ", protein_identifiers
print "Exercise 4.2 - GO Terms"
for protein_id in go_terms_dict:
    print protein_id, go_terms_dict[protein_id]
print "Exercise 4.3 - nucleus and TF counts and probabilities:"
print "                           count probability"
print "    In nucleus and TF        :", tf_nucleus_count, \
    float(tf_nucleus_count) / protein_count
print "    Not in nucleus and TF    :", tf_not_nucleus_count, \
    float(tf_not_nucleus_count) / protein_count
print "    In nucleus and not TF    :", not_tf_nucleus_count, \
    float(not_tf_nucleus_count) / protein_count
print "    Not in nucleus and not TF:", not_tf_not_nucleus_count, \
    float(not_tf_not_nucleus_count) / protein_count
```

```
print "Exercise 4.4: probability of TF and in nucleus = ", \
    float(tf_nucleus_count) / protein_count
print "            probability of in nucleus given TF = P(N|TF) = P(N,TF) /
P(TF)"
print "                                                 = ", \
    (float(tf_nucleus_count) / protein_count) / (float(tf_nucleus_count +
tf_not_nucleus_count) / protein_count)
```

Output



Java - prac00/exercise4.py - Eclipse
File Edit Source Refactoring Navigate Search Project Run Window Help
Quick Access
Java
Problems
Javadoc
Declaration
Console

```
<terminated> /var/jacekrad/github-sem2/biol3014/prac00/exercise4.py
Exercise 4.1 - protein identifiers:  ['P38903', 'P31383', 'P39533', 'P47177', 'P47096', 'P40433', 'P39970', 'P38720', 'P25371', 'P47182']
Exercise 4.2 - GO Terms
P39533 set(['GO:0008652', 'GO:0051539', 'GO:0019878', 'GO:0016829', 'GO:0005739', 'GO:0006099', 'GO:0008150', 'GO:0008152', 'GO:0009085', 'GO:0003994', 'GO:0046872', 'GO:0051536'])
P38720 set(['GO:0005737', 'GO:0050662', 'GO:0050661', 'GO:0019521', 'GO:0016616', 'GO:0006098', 'GO:0004616', 'GO:0034599', 'GO:0009051', 'GO:0055114', 'GO:0016491'])
P40433 set(['GO:0005737', 'GO:0016311', 'GO:0016310', 'GO:0016301', 'GO:0000166', 'GO:0046835', 'GO:0003824', 'GO:0016740', 'GO:0004331', 'GO:0005515', 'GO:0003873', 'GO:0006000', 'GO:0005524', 'GO:0006003'])
P31383 set(['GO:0005737', 'GO:0007094', 'GO:0000159', 'GO:0016311', 'GO:0005634', 'GO:0005816', 'GO:0004722', 'GO:0006417', 'GO:0043332', 'GO:0005935', 'GO:0005934'])
P25371 set(['GO:0042626', 'GO:0016020', 'GO:0005783', 'GO:0016887', 'GO:0016021', 'GO:0006810', 'GO:0017111', 'GO:0006200', 'GO:0005789', 'GO:0008152', 'GO:0000166', 'GO:0055085', 'GO:0005524'])
P39970 set(['GO:0043565', 'GO:0003700', 'GO:0005634', 'GO:0006355', 'GO:0001077', 'GO:0045944', 'GO:0006351', 'GO:0003677'])
P47177 set(['GO:0005737', 'GO:0004497', 'GO:0003824', 'GO:0003674', 'GO:0008150', 'GO:0018580', 'GO:0055114', 'GO:0016491'])
P47182 set(['GO:0005575', 'GO:0055114', 'GO:0006081', 'GO:0018456', 'GO:0016491'])
P47096 set(['GO:0005737', 'GO:0000334', 'GO:0051213', 'GO:0034354', 'GO:0005634', 'GO:0006569', 'GO:0043420', 'GO:0008198', 'GO:0009435', 'GO:0055114', 'GO:0005506', 'GO:0046872', 'GO:0019805', 'GO:0019363', 'GO:0016491'])
P38903 set(['GO:0005737', 'GO:0031107', 'GO:0050790', 'GO:0000159', 'GO:0032186', 'GO:0006470', 'GO:0005634', 'GO:0034613', 'GO:0005816', 'GO:0031578', 'GO:0070199', 'GO:0000780', 'GO:0051754', 'GO:0007165', 'GO:0005935', 'GO:
Exercise 4.3 - nucleus and TF counts and probabilities:
                           count probability
    In nucleus and TF        : 1 0.1
    Not in nucleus and TF    : 0 0.0
    In nucleus and not TF    : 3 0.3
    Not in nucleus and not TF: 6 0.6
Exercise 4.4: probability of TF and in nucleus =  0.1
            probability of in nucleus given TF = P(N|TF) = P(N,TF) / P(TF)
                                               =  1.0
```

Exercise 5

As specified, this question is a modified version of exercise 4. Both code and full output are provided below. Note that results for 5.3 and 5.4 are at the very end of the output.

Code

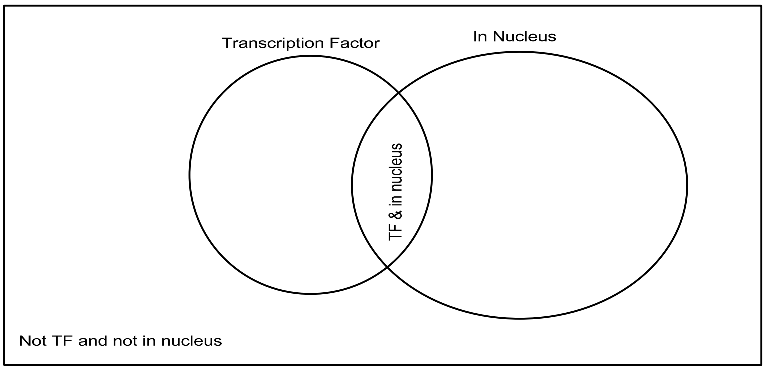
```
'''
Created on 02/08/2014

@author: jacekrad
'''
from webservice import getGOTerms

yeast_file = open("yeast_transcriptome_10.txt")
protein_identifiers = [] # list of protein IDs

# total number of proteins read from file
protein_count = 0

# counters for TF and nucleus proteins
tf_nucleus_count = 0
not_tf_nucleus_count = 0
tf_not_nucleus_count = 0
not_tf_not_nucleus_count = 0

for line in yeast_file:
    protein_count += 1
    fields = line.strip().split("\t")
    protein_id = fields[0]
    protein_identifiers.append(protein_id) # ex 5.1
yeast_file.close()

# dictionary of go_terms with the gene ID as the key
go_terms_dict = getGOTerms(protein_identifiers)

for protein_id in go_terms_dict:
    go_terms = go_terms_dict.get(protein_id)
    if "GO:0005634" in go_terms: # check if nucleus
        if "GO:0003700" in go_terms: # check if TF
            tf_nucleus_count += 1
        else:
            not_tf_nucleus_count += 1
    else:
        if "GO:0003700" in go_terms: # check if TF
            tf_not_nucleus_count += 1
        else:
            not_tf_not_nucleus_count += 1


print "Exercise 5.1 - protein identifiers: ", protein_identifiers
print "Exercise 5.2 - GO Terms"
print go_terms_dict
print "Exercise 5.3 - nucleus and TF counts and probabilities:"
print "                             count probability"
print "    In nucleus and TF          :", tf_nucleus_count, \
    float(tf_nucleus_count) / protein_count
print "    Not in nucleus and TF      :", tf_not_nucleus_count, \
    float(tf_not_nucleus_count) / protein_count
print "    In nucleus and not TF      :", not_tf_nucleus_count, \
    float(not_tf_nucleus_count) / protein_count
print "    Not in nucleus and not TF:", not_tf_not_nucleus_count, \
    float(not_tf_not_nucleus_count) / protein_count

print "Exercise 5.4: probability of TF and in nucleus = ", \
    float(tf_nucleus_count) / protein_count
print "            probability of in nucleus given TF = P(N|TF) = P(N,TF) / 
P(TF)"
print "                                                = ", \
    (float(tf_nucleus_count) / protein_count) / (float(tf_nucleus_count + 
tf_not_nucleus_count) / protein_count)
```

Output

```
Exercise 5.1 - protein identifiers:  ['P38903', 'P31383', 'Q00362', 
'P47177', 'P47096', 'P40433', 'Q12471', 'P38720', 'P53319', 'P47182', 
'P42884', 'Q08361', 'P43546', 'P25612', 'Q07747', 'P43547', 'P12904', 
'P37898', 'P08521', 'P32357', 'P23542', 'Q01802', 'Q08641', 'P14164', 
'Q02486', 'P47146', 'P15891', 'P39970', 'P40535', 'Q00955', 'P31787', 
'P21192', 'P28240', 'Q12031', 'P47129', 'P32316', 'Q07622', 'Q08981', 
'Q04401', 'P21147', 'P39533', 'P19414', 'P13711', 'P32463', 'Q01574', 
'P52910', 'P60010', 'Q02336', 'Q03233', 'P53909', 'Q2V2Q1', 'P00330', 
'P00331', 'P07246', 'P10127', 'P38113', 'Q04894', 'P25377', 'P47143', 
'P25371', 'Q01976', 'P07248', 'P48360', 'Q12184', 'P04710', 'P18239', 
'P18238', 'P38872', 'P25613', 'Q05955', 'Q07732', 'P32493', 'P22136', 
'Q12089', 'P53930', 'P32317', 'P32794', 'Q99222', 'P39925', 'P33304', 
'P22149', 'Q08957', 'P32323', 'P32781', 'Q12482', 'Q04412', 'P40529', 
'P38628', 'P25376', 'P38090', 'P43548', 'P43567', 'Q12449', 'P25649', 
'Q12433', 'P38013', 'P29589', 'P03875', 'P03876', 'Q12152']
Exercise 5.2 - GO Terms
{'P40433': set(['GO:0005737', 'GO:0016311', 'GO:0016310', 'GO:0016301', 
'GO:0000166', 'GO:0046835', 'GO:0003824', 'GO:0016740', 'GO:0004331', 
'GO:0005515', 'GO:0003873', 'GO:0006000', 'GO:0005524', 'GO:0006003']), 
'Q12184': set(['GO:0046872', 'GO:0006784', 'GO:0051536', 'GO:0051537', 
'GO:0005739', 'GO:0016226', 'GO:0016653', 'GO:0005759', 'GO:0006744', 
'GO:0055114', 'GO:0009055']), 'P18238': set(['GO:0016020', 'GO:0016021', 
'GO:0015866', 'GO:0005215', 'GO:0015886', 'GO:0006810', 'GO:0005739', 
'GO:0055085', 'GO:0005743', 'GO:0009061', 'GO:0015867', 'GO:0005471']), 
'P18239': set(['GO:0016020', 'GO:0006915', 'GO:0016021', 'GO:0015866', 
'GO:0005215', 'GO:0015886', 'GO:0006810', 'GO:0005739', 'GO:0055085', 
'GO:0005743', 'GO:0009061', 'GO:0009060', 'GO:0015867', 'GO:0006839', 
'GO:0005471']), 'P15891': set(['GO:0005737', 'GO:0005856', 'GO:0008104', 
'GO:0000147', 'GO:0030479', 'GO:0051016', 'GO:0051015', 'GO:0005622', 
'GO:0003779', 'GO:0005938', 'GO:0005515']), 'P32317': set(['GO:0016558', 
'GO:0000166', 'GO:0005743', 'GO:0003674', 'GO:0006515', 'GO:0034599', 
'GO:0005524']), 'P32316': set(['GO:0005737', 'GO:0006084', 'GO:0008775', 
'GO:0006083', 'GO:0016787', 'GO:0005829', 'GO:0005739', 'GO:0003824', 
'GO:0003986']), 'P32493': set(['GO:0006417', 'GO:0045182', 'GO:0005739']), 
'P12904': set(['GO:0005737', 'GO:0006468', 'GO:0007031', 'GO:0001302', 
'GO:0006355', 'GO:0045722', 'GO:0005634', 'GO:0030554', 'GO:0043539', 
'GO:0000166', 'GO:0003824', 'GO:0031588', 'GO:0006357', 'GO:0006351', 
'GO:0005515', 'GO:0005886', 'GO:0071902', 'GO:0005975', 'GO:0004679', 
'GO:0005641', 'GO:0005524']), 'Q08641': set(['GO:0005737', 'GO:0005856', 
'GO:0030674', 'GO:0032432', 'GO:0030488', 'GO:0030479', 'GO:0051017', 
'GO:0051015', 'GO:0003779', 'GO:0016740', 'GO:0005884', 'GO:0008168',
'GO:0052735', 'GO:0032259']), 'P47129': set(['GO:0005575', 'GO:0008150',
'GO:0003674', 'GO:0005515']), 'P25377': set(['GO:0006066', 'GO:0016616',
'GO:0008106', 'GO:0005575', 'GO:0048037', 'GO:0008270', 'GO:0046872',
'GO:0055114', 'GO:0016491']), 'P25376': set(['GO:0016020', 'GO:0006810',
'GO:0016021', 'GO:0055085', 'GO:0003333', 'GO:0006865', 'GO:0015193',
'GO:0035524', 'GO:0005886', 'GO:0015171']), 'P25371': set(['GO:0042626',
'GO:0016020', 'GO:0005783', 'GO:0016887', 'GO:0016021', 'GO:0006810',
'GO:0017111', 'GO:0006200', 'GO:0005789', 'GO:0008152', 'GO:0000166',
'GO:0055085', 'GO:0005524']), 'P10127': set(['GO:0006113', 'GO:0000947',
'GO:0005739', 'GO:0004022', 'GO:0046872', 'GO:0055114', 'GO:0016491']),
'P38113': set(['GO:0000947', 'GO:0043458', 'GO:0004022', 'GO:0006116',
'GO:0008270', 'GO:0046872', 'GO:0055114', 'GO:0016491']), 'P32794':
set(['GO:0042273', 'GO:0016887', 'GO:0042493', 'GO:0030687', 'GO:0017111',
'GO:0006200', 'GO:0005622', 'GO:0008152', 'GO:0000166', 'GO:0005524']),
'P43567': set(['GO:0008453', 'GO:0008483', 'GO:0006545', 'GO:0003824',
'GO:0016740', 'GO:0019265', 'GO:0008152', 'GO:0030170']), 'Q01976':
set(['GO:0016787', 'GO:0046872', 'GO:0019693', 'GO:0005739',
'GO:0047631']), 'Q99222': set(['GO:0005737', 'GO:0008105', 'GO:0005634',
'GO:0000282', 'GO:0003674', 'GO:0005938', 'GO:0051666', 'GO:0005886',
'GO:0005935', 'GO:0048471']), 'Q07747': set(['GO:0006081', 'GO:0005575',
'GO:0006950', 'GO:0018456', 'GO:0055114', 'GO:0016491']), 'Q12449':
set(['GO:0005737', 'GO:0006457', 'GO:0032781', 'GO:0034605', 'GO:0051087',
'GO:0006950', 'GO:0005515', 'GO:0001671']), 'Q04412': set(['GO:0005737',
'GO:0006891', 'GO:0006810', 'GO:0006888', 'GO:0043547', 'GO:0032312',
'GO:0005543', 'GO:0005096', 'GO:0008060', 'GO:0008270', 'GO:0015031',
'GO:0046872', 'GO:0005768', 'GO:0005802']), 'Q08981': set(['GO:0005737',
'GO:0051301', 'GO:0005634', 'GO:0006355', 'GO:0006351', 'GO:0005515',
'GO:0007067', 'GO:0051436', 'GO:0055105', 'GO:0007049']), 'P22149':
set(['GO:0005737', 'GO:0034087', 'GO:0034758', 'GO:0005634', 'GO:0045132',
'GO:0036086', 'GO:0000778', 'GO:0006355', 'GO:0045944', 'GO:0006351',
'GO:0005515', 'GO:0006366', 'GO:0000987', 'GO:0007059', 'GO:0046872',
'GO:0000982']), 'P39970': set(['GO:0043565', 'GO:0003700', 'GO:0005634',
'GO:0006355', 'GO:0001077', 'GO:0045944', 'GO:0006351', 'GO:0003677']),
'Q02336': set(['GO:0000124', 'GO:0006348', 'GO:0000183', 'GO:0016573',
'GO:0035066', 'GO:0006357', 'GO:0046872', 'GO:0004402', 'GO:0005634',
'GO:0003713', 'GO:0006355', 'GO:0005671', 'GO:0006351', 'GO:0005515',
'GO:0003677', 'GO:0046695', 'GO:0003682', 'GO:0001786', 'GO:0016568',
'GO:0008270']), 'Q01802': set(['GO:0019266', 'GO:0001302', 'GO:0006532',
'GO:0001300', 'GO:0004069', 'GO:0008483', 'GO:0080130', 'GO:0006520',
'GO:0009058', 'GO:0005739', 'GO:0003824', 'GO:0006536', 'GO:0016740',
'GO:0006531', 'GO:0006103', 'GO:0005759', 'GO:0030170']), 'Q12433':
set(['GO:0005737', 'GO:0016573', 'GO:0005634', 'GO:0004402', 'GO:0006355',
'GO:0005671', 'GO:0006351', 'GO:0005515']), 'Q03233': set(['GO:0005737',
'GO:0003674', 'GO:0030433']), 'P33304': set(['GO:0000753', 'GO:0000750',
'GO:0001400', 'GO:0008277', 'GO:0005057', 'GO:0000767', 'GO:0035556']),
'Q07732': set(['GO:0005737', 'GO:0005856', 'GO:0007126', 'GO:0005628',
'GO:0030476', 'GO:0000001', 'GO:0016020', 'GO:0005816', 'GO:0030435',
'GO:0003674', 'GO:0051301', 'GO:0005515', 'GO:0007049']), 'P14164':
set(['GO:0030466', 'GO:0006338', 'GO:0000122', 'GO:0044374', 'GO:0006355',
'GO:0005634', 'GO:0006281', 'GO:0043565', 'GO:0006974', 'GO:0003688',
'GO:0001077', 'GO:0006260', 'GO:0006261', 'GO:0070911', 'GO:0003677',
'GO:0045944', 'GO:0000978', 'GO:0000113', 'GO:0006351']), 'Q12089':
set(['GO:0016071', 'GO:0070124', 'GO:0016020', 'GO:0005739', 'GO:0019898',
'GO:0005743', 'GO:0003674']), 'P03876': set(['GO:0004129', 'GO:0006278',
'GO:0020037', 'GO:1902600', 'GO:0004519', 'GO:0016021', 'GO:0005739',
'GO:0090305', 'GO:0009060', 'GO:0003723', 'GO:0003676', 'GO:0003964',
'GO:0005506', 'GO:0009055', 'GO:0008380', 'GO:0055114', 'GO:0006397']),
'P03875': set(['GO:0004129', 'GO:0006278', 'GO:0006315', 'GO:1902600',
'GO:0004519', 'GO:0005739', 'GO:0090305', 'GO:0045333', 'GO:0003723',
'GO:0003676', 'GO:0003964', 'GO:0055114', 'GO:0006397']), 'P53319':
set(['GO:0050662', 'GO:0050661', 'GO:0019521', 'GO:0016616', 'GO:0005829',
'GO:0006098', 'GO:0004616', 'GO:0009051', 'GO:0055114', 'GO:0016491']),
'Q04401': set(['GO:0034553', 'GO:0005739', 'GO:0015976', 'GO:0003674',
'GO:0006111', 'GO:0005758', 'GO:0006094']), 'P40535': set(['GO:0043565',
'GO:0003700', 'GO:0061429', 'GO:0005634', 'GO:0071400', 'GO:0006355',
'GO:0001077', 'GO:0045944', 'GO:0006351', 'GO:0003677', 'GO:0006259',
'GO:0005829', 'GO:0071244']), 'P04710': set(['GO:0016020', 'GO:0015886',
'GO:0016021', 'GO:0006783', 'GO:0005215', 'GO:0005829', 'GO:0006810',
'GO:0005739', 'GO:0055085', 'GO:0005743', 'GO:0009060', 'GO:0015866',
'GO:0015867', 'GO:0006839', 'GO:0005758', 'GO:0005471']), 'P47143':
set(['GO:0016301', 'GO:0044209', 'GO:0016773', 'GO:0016310', 'GO:0004001',
'GO:0000166', 'GO:0016740', 'GO:0006166', 'GO:0006144', 'GO:0005524']),
'P22136': set(['GO:0006417', 'GO:0070124', 'GO:0003674', 'GO:0003723',
'GO:0005739']), 'P60010': set(['GO:0032432', 'GO:0005200', 'GO:0031011',
'GO:0000011', 'GO:0035267', 'GO:0030010', 'GO:0030050', 'GO:0005737',
'GO:0000142', 'GO:0000001', 'GO:0006887', 'GO:0004402', 'GO:0000166',
'GO:0016573', 'GO:0005524', 'GO:0005856', 'GO:0030476', 'GO:0006281',
'GO:0009306', 'GO:0030479', 'GO:0031505', 'GO:0005884', 'GO:0006897',
'GO:0034599', 'GO:0000132', 'GO:0000916', 'GO:0001300', 'GO:0007119',
'GO:0000812', 'GO:0005515']), 'P47146': set(['GO:0005575', 'GO:0003674',
'GO:0000226']), 'Q07622': set(['GO:0009967', 'GO:0003674', 'GO:0031505',
'GO:0005515']), 'P37898': set(['GO:0004177', 'GO:0005977', 'GO:0016787',
'GO:0008270', 'GO:0006508', 'GO:0008233', 'GO:0046872', 'GO:0008237']),
'P38090': set(['GO:0016020', 'GO:0006865', 'GO:0016021', 'GO:0005515',
'GO:0055085', 'GO:1902274', 'GO:0003333', 'GO:0006810', 'GO:0015203',
'GO:0000329', 'GO:0005789', 'GO:0003674', 'GO:0005887', 'GO:1902269',
'GO:0015171']), 'P53930': set(['GO:0005737', 'GO:0006348', 'GO:0006338',
'GO:0006281', 'GO:0043486', 'GO:0005634', 'GO:0000812', 'GO:0006974',
'GO:0006355', 'GO:0003674', 'GO:0006351', 'GO:0005515', 'GO:0016568',
'GO:0035267']), 'P47096': set(['GO:0005737', 'GO:0000334', 'GO:0051213',
'GO:0034354', 'GO:0005634', 'GO:0006569', 'GO:0043420', 'GO:0008198',
'GO:0009435', 'GO:0055114', 'GO:0005506', 'GO:0046872', 'GO:0019805',
'GO:0019363', 'GO:0016491']), 'P29589': set(['GO:0005575', 'GO:0008150',
'GO:0003674']), 'Q12152': set(['GO:0006468', 'GO:0016310', 'GO:0016301',
'GO:0000166', 'GO:0005575', 'GO:0004674', 'GO:0016740', 'GO:0004672',
'GO:0016772', 'GO:0005524']), 'P23542': set(['GO:0005737', 'GO:0006532',
'GO:0004069', 'GO:0008483', 'GO:0005829', 'GO:0006520', 'GO:0005777',
'GO:0009058', 'GO:0080130', 'GO:0003824', 'GO:0006536', 'GO:0016740',
'GO:0006531', 'GO:0006103', 'GO:0030170']), 'P40529': set(['GO:0005737',
'GO:0005794', 'GO:0006891', 'GO:0006888', 'GO:0043547', 'GO:0032312',
'GO:0005096', 'GO:0008060', 'GO:0008270', 'GO:0046872']), 'Q00955':
set(['GO:0005737', 'GO:0006633', 'GO:0006631', 'GO:0046872', 'GO:0005783',
'GO:0016020', 'GO:0006606', 'GO:0006998', 'GO:0000166', 'GO:0003824',
'GO:2001295', 'GO:0008152', 'GO:0003989', 'GO:0042759', 'GO:0004075',
'GO:0005789', 'GO:0016874', 'GO:0005524', 'GO:0006629']), 'P21192':
set(['GO:0005829', 'GO:0005634', 'GO:0006366', 'GO:2001043', 'GO:0000083',
'GO:0006355', 'GO:0006351', 'GO:0003676', 'GO:0003677', 'GO:0000987',
'GO:0046872', 'GO:0000982']), 'P31383': set(['GO:0005737', 'GO:0007094',
'GO:0000159', 'GO:0016311', 'GO:0005634', 'GO:0005816', 'GO:0004722',
'GO:0006417', 'GO:0043332', 'GO:0005935', 'GO:0005934']), 'P19414':
set(['GO:0005737', 'GO:0046872', 'GO:0000002', 'GO:0051539', 'GO:0051536',
'GO:0005829', 'GO:0016829', 'GO:0005739', 'GO:0006099', 'GO:0008152',
'GO:0005758', 'GO:0005759', 'GO:0003994', 'GO:0003690', 'GO:0042645',
'GO:0003697']), 'Q01574': set(['GO:0005737', 'GO:0006085', 'GO:0005783',
'GO:0016880',
 'GO:0005634', 'GO:0016208', 'GO:0019654', 'GO:0005739', 'GO:0000166',
'GO:0003824', 'GO:0019427', 'GO:0008152', 'GO:0043231', 'GO:0003987',
'GO:0016874', 'GO:0005829', 'GO:0005524', 'GO:0016573']), 'P25649':
set(['GO:0005737', 'GO:0005634', 'GO:0005671', 'GO:0016573',
'GO:0005515']), 'Q02486': set(['GO:0000002', 'GO:0000001', 'GO:0005634',
'GO:0000262', 'GO:0005739', 'GO:0008301', 'GO:0090139', 'GO:0003677',
'GO:0042645']), 'Q12471': set(['GO:0005737', 'GO:0016310', 'GO:0016301',
'GO:0000166', 'GO:0046835', 'GO:0003824', 'GO:0016740', 'GO:0006110',
'GO:0003873', 'GO:0006000', 'GO:0005524', 'GO:0006003']), 'P31787':
set(['GO:0015909', 'GO:0001300', 'GO:0005324', 'GO:0006810', 'GO:0008289',
'GO:0000062', 'GO:0005576']), 'P32357': set(['GO:0005737', 'GO:0000244',
'GO:0005634', 'GO:0008380', 'GO:0003674', 'GO:0005682', 'GO:0006397']),
'P32463': set(['GO:0006633', 'GO:0006631', 'GO:0006629', 'GO:0005739',
'GO:0000036', 'GO:0070469', 'GO:0009107', 'GO:0055114']), 'P00330':
set(['GO:0005737', 'GO:0043458', 'GO:0000947', 'GO:0004022', 'GO:0006116',
'GO:0008270', 'GO:0019170', 'GO:0046872', 'GO:0055114', 'GO:0016491']),
'P00331': set(['GO:0005737', 'GO:0000947', 'GO:0004022', 'GO:0006116',
'GO:0006067', 'GO:0008270', 'GO:0046872', 'GO:0055114', 'GO:0016491']),
'P13711': set(['GO:0006631', 'GO:0050660', 'GO:0005782', 'GO:0006629',
'GO:0006635', 'GO:0016627', 'GO:0005777', 'GO:0033540', 'GO:0008152',
'GO:0003995', 'GO:0003997', 'GO:0055114', 'GO:0016491']), 'P38013':
set(['GO:0005737', 'GO:0016209', 'GO:0045454', 'GO:0051920', 'GO:0008379',
'GO:0010038', 'GO:0034599', 'GO:0004601', 'GO:0055114', 'GO:0016491']),
'Q08361': set(['GO:0005575', 'GO:0055114', 'GO:0006081', 'GO:0018456',
'GO:0016491']), 'P48360': set(['GO:0006879', 'GO:0005739', 'GO:0015039',
'GO:0005743', 'GO:0005759', 'GO:0006744', 'GO:0055114', 'GO:0016491']),
'P42884': set(['GO:0006081', 'GO:0005575', 'GO:0005515', 'GO:0018456',
'GO:0055114', 'GO:0016491']), 'Q00362': set(['GO:0005737', 'GO:0034504',
'GO:0050790', 'GO:0000159', 'GO:0006470', 'GO:0005634', 'GO:0000078',
'GO:0061586', 'GO:0010971', 'GO:0001100', 'GO:0007165', 'GO:0005515',
'GO:0008601', 'GO:0005935', 'GO:0005934', 'GO:0007049']), 'Q12031':
set(['GO:0046421', 'GO:0019629', 'GO:0016829', 'GO:0005739', 'GO:0004451',
'GO:0003824', 'GO:0019752', 'GO:0008152', 'GO:0005759']), 'P47177':
set(['GO:0005737', 'GO:0004497', 'GO:0003824', 'GO:0003674', 'GO:0008150',
'GO:0018580', 'GO:0055114', 'GO:0016491']), 'Q2V2Q1': set(['GO:0000750',
'GO:0000122', 'GO:0005634', 'GO:0001012', 'GO:0006355', 'GO:0006351']),
'P08521': set(['GO:0006417', 'GO:0005829', 'GO:0045182']), 'P39533':
set(['GO:0008652', 'GO:0051539', 'GO:0019878', 'GO:0016829', 'GO:0005739',
'GO:0006099', 'GO:0008150', 'GO:0008152', 'GO:0009085', 'GO:0003994',
'GO:0046872', 'GO:0051536']), 'P39925': set(['GO:0000329', 'GO:0006200',
'GO:0016887', 'GO:0008233', 'GO:0008237', 'GO:0016021', 'GO:0016020',
'GO:0005739', 'GO:0000166', 'GO:0097002', 'GO:0002181', 'GO:0006508',
'GO:0046872', 'GO:0005524', 'GO:0016787', 'GO:0004222', 'GO:0006461',
'GO:0017111', 'GO:0031966', 'GO:0006465', 'GO:0001302', 'GO:0005743',
'GO:0005515', 'GO:0005745', 'GO:0008270']), 'P32323': set(['GO:0000752',
'GO:0005618', 'GO:0016020', 'GO:0005576', 'GO:0009277', 'GO:0007155',
'GO:0050839', 'GO:0019236', 'GO:0031225']), 'P38720': set(['GO:0005737',
'GO:0050662', 'GO:0050661', 'GO:0019521', 'GO:0016616', 'GO:0006098',
'GO:0004616', 'GO:0034599', 'GO:0009051', 'GO:0055114', 'GO:0016491']),
'Q12482': set(['GO:0016020', 'GO:0015297', 'GO:0006810', 'GO:0044271',
'GO:0015292', 'GO:0015813', 'GO:0005313', 'GO:0006865', 'GO:0005739',
'GO:0055085', 'GO:0005743', 'GO:0015810', 'GO:0016021', 'GO:0015183']),
'Q05955': set(['GO:0005737', 'GO:0005856', 'GO:0007126', 'GO:0005816',
'GO:0005198', 'GO:0030435', 'GO:0051301', 'GO:0007049']), 'P32781':
set(['GO:0007155', 'GO:0050839', 'GO:0000752', 'GO:0009277']), 'Q08957':
set(['GO:0006879', 'GO:0005634', 'GO:0036086', 'GO:0006355', 'GO:0001077',
'GO:0045944', 'GO:0006351', 'GO:0000978', 'GO:0034599']), 'P38872':
set(['GO:0005737', 'GO:0005856', 'GO:0030476', 'GO:0005634', 'GO:0005816',
'GO:0005575', 'GO:0030435', 'GO:0003674', 'GO:0005515']), 'P47182':
set(['GO:0005575', 'GO:0055114', 'GO:0006081', 'GO:0018456',
'GO:0016491']), 'P38903': set(['GO:0005737', 'GO:0031107', 'GO:0050790',
'GO:0000159', 'GO:0032186', 'GO:0006470', 'GO:0005634', 'GO:0034613',
'GO:0005816', 'GO:0031578', 'GO:0070199', 'GO:0000780', 'GO:0051754',
'GO:0007165', 'GO:0005935', 'GO:0008601', 'GO:0000776']), 'P21147':
```

```
set(['GO:0006633', 'GO:0006631', 'GO:0046872', 'GO:0030176', 'GO:0006629',
'GO:0020037', 'GO:0016021', 'GO:0016020', 'GO:0005789', 'GO:0016491',
'GO:0006636', 'GO:0009055', 'GO:0005506', 'GO:0004768', 'GO:0016717',
'GO:0055114', 'GO:0005783']), 'P43546': set(['GO:0005575', 'GO:0055114',
'GO:0006081', 'GO:0018456', 'GO:0016491']), 'P43547': set(['GO:0005575',
'GO:0055114', 'GO:0006081', 'GO:0018456', 'GO:0016491']), 'P43548':
set(['GO:0016020', 'GO:0016021', 'GO:0003333', 'GO:0006865', 'GO:0055085',
'GO:0006810', 'GO:0005886', 'GO:0015171']), 'P25612': set(['GO:0005575',
'GO:0055114', 'GO:0006081', 'GO:0018456', 'GO:0016491']), 'P25613':
set(['GO:0007126', 'GO:0016020', 'GO:0008519', 'GO:0006846', 'GO:0005773',
'GO:0035433', 'GO:0006811', 'GO:0006810', 'GO:0015123', 'GO:0005774',
'GO:0019740', 'GO:0005886', 'GO:0015696', 'GO:0055085', 'GO:0072488',
'GO:0016021']), 'Q04894': set(['GO:0006066', 'GO:0006081', 'GO:0033833',
'GO:0008106', 'GO:0033845', 'GO:0033859', 'GO:0005575', 'GO:0008270',
'GO:0046872', 'GO:0055114', 'GO:0016491']), 'P07246': set(['GO:0008270',
'GO:0046872', 'GO:0000947', 'GO:0005739', 'GO:0006116', 'GO:0004022',
'GO:0005759', 'GO:0016491', 'GO:0055114', 'GO:0006113']), 'P52910':
set(['GO:0005737', 'GO:0006085', 'GO:0001302', 'GO:0016573', 'GO:0005730',
'GO:0016880', 'GO:0005634', 'GO:0016208', 'GO:0000166', 'GO:0003824',
'GO:0019427', 'GO:0008152', 'GO:0003987', 'GO:0006090', 'GO:0016874',
'GO:0005829', 'GO:0005524']), 'P38628': set(['GO:0006048', 'GO:0005975',
'GO:0016853', 'GO:0000287', 'GO:0034221', 'GO:0004610', 'GO:0005515',
'GO:0016868', 'GO:0046872']), 'P07248': set(['GO:0031936', 'GO:0061410',
'GO:0003676', 'GO:0003677', 'GO:0001094', 'GO:0001190', 'GO:0001093',
'GO:0006325', 'GO:0005634', 'GO:0006366', 'GO:0046872', 'GO:0000982',
'GO:0061425', 'GO:0061424', 'GO:0001102', 'GO:0061429', 'GO:0007031',
'GO:0097235', 'GO:0071400', 'GO:0006355', 'GO:0006351', 'GO:0000978',
'GO:0034401']), 'P28240': set(['GO:0005737', 'GO:0005575', 'GO:0016829',
'GO:0019752', 'GO:0004451', 'GO:0003824', 'GO:0006099', 'GO:0008152',
'GO:0006097']), 'P53909': set(['GO:0005737', 'GO:0016787', 'GO:0043101',
'GO:0005634', 'GO:0046872', 'GO:0019239', 'GO:0009117', 'GO:0008270',
'GO:0043103', 'GO:0000034', 'GO:0009168', 'GO:0006146'])}
Exercise 5.3 - nucleus and TF counts and probabilities:
                           count probability
    In nucleus and TF        : 2 0.02
    Not in nucleus and TF    : 0 0.0
    In nucleus and not TF    : 23 0.23
    Not in nucleus and not TF: 75 0.75
Exercise 5.4: probability of TF and in nucleus =  0.02
            probability of in nucleus given TF = P(N|TF) = P(N,TF) / P(TF)
                                               =  1.0
```

Exercise 6



Lets denote the set of Transcription Factor proteins as T and the set of in nucleus proteins as N. Let U represent the universal set of all proteins in the file.

Now,

T, N and T∩N are calculated using the filter python function as shown in the following code snippet

```
in_nucleus_proteins = filter(lambda x: "GO:0005634" in x[1], \
                             go_terms_dict.iteritems())
tf_proteins = filter(lambda x: "GO:0003700" in x[1], \
                     go_terms_dict.iteritems())
tf_in_nucleus_proteins = filter(lambda x: "GO:0003700" in x[1] \
                                and "GO:0005634" in x[1], \
                                go_terms_dict.iteritems())
```

Now

$T\setminus N = T\cap N' = T - T\cap N$

$N\setminus T = N\cap T' = N - T\cap N$

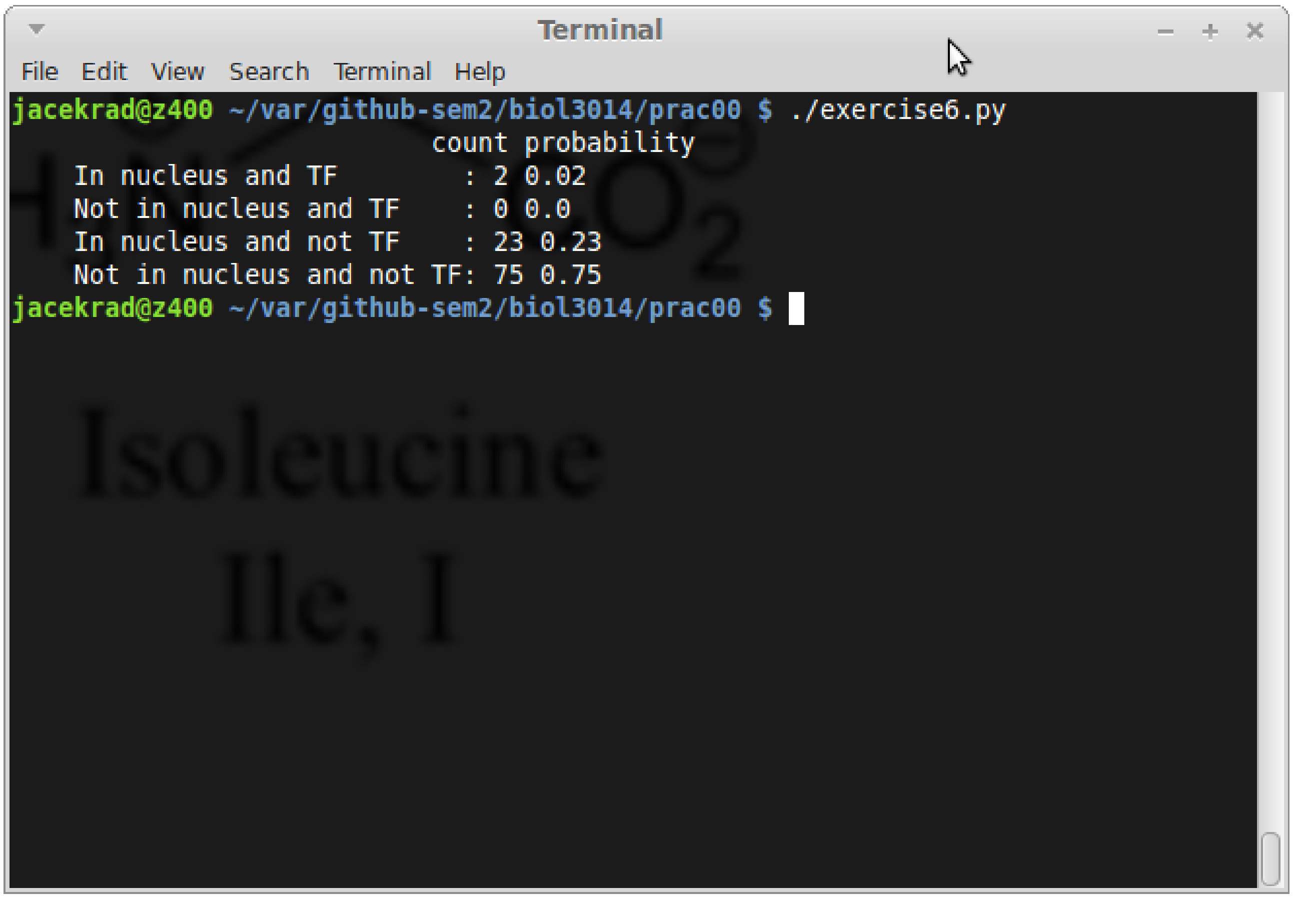
$T`\cap N' = (T\cup N)' = U - (T\cup N) = U - T - N + T\cap N$

Full Code

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
'''
Created on 02/08/2014

@author: jacekrad
'''
from webservice import getGOTerms

yeast_file = open("yeast_transcriptome_100.txt")
```

```python
protein_identifiers = []  # list of protein IDs

# total number of proteins read from file
protein_count = 0



for line in yeast_file:
    protein_count += 1
    fields = line.strip().split("\t")
    protein_id = fields[0]
    protein_identifiers.append(protein_id)
yeast_file.close()

# dictionary of go_terms with the gene ID as the key
go_terms_dict = getGOTerms(protein_identifiers)

in_nucleus_proteins = filter(lambda x: "GO:0005634" in x[1], \
                             go_terms_dict.iteritems())
tf_proteins = filter(lambda x: "GO:0003700" in x[1], \
                     go_terms_dict.iteritems())
tf_in_nucleus_proteins = filter(lambda x: "GO:0003700" in x[1] \
                                and "GO:0005634" in x[1], \
                                go_terms_dict.iteritems())

in_nucleus_count = len(in_nucleus_proteins)
tf_count = len(tf_proteins)
tf_in_nucleus_count = len(tf_in_nucleus_proteins)

# U - universal set (all proteins from file)
# T - set of transcription factor proteins
# N - set of in nucleus proteins
# T\N = T∩N' = T – T∩N
# N\T = N∩T' = N – T∩N
# T`∩N' = (T∪N)' = U - (T∪N) = U – T – N + T∩N
not_tf_nucleus_count = in_nucleus_count - tf_in_nucleus_count
tf_not_nucleus_count = tf_count - tf_in_nucleus_count
not_tf_not_nucleus_count = protein_count - tf_count - in_nucleus_count +
tf_count

print "                           count probability"
print "    In nucleus and TF        :", tf_in_nucleus_count, \
    float(tf_in_nucleus_count) / protein_count
print "    Not in nucleus and TF    :", tf_not_nucleus_count, \
    float(tf_not_nucleus_count) / protein_count
print "    In nucleus and not TF    :", not_tf_nucleus_count, \
    float(not_tf_nucleus_count) / protein_count
print "    Not in nucleus and not TF:", not_tf_not_nucleus_count, \
    float(not_tf_not_nucleus_count) / protein_count
```

Output



```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ ./exercise6.py
                           count probability
    In nucleus and TF         : 2 0.02
    Not in nucleus and TF     : 0 0.0
    In nucleus and not TF     : 23 0.23
    Not in nucleus and not TF: 75 0.75
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ 
```

Exercise 7

As specified, this exercise retrieves proteins from UniProtKB by GO terms and then filters proteins which are also contained in our file. The results of doing this differ to the results of processing **`yeast_transcriptome.txt`** (same file) using the code from question 6. The summary of the results is shown in the table below and also available in the output section. Let T be the of transcription factor proteins and N be the set in nucleus proteins. Table below show the counts

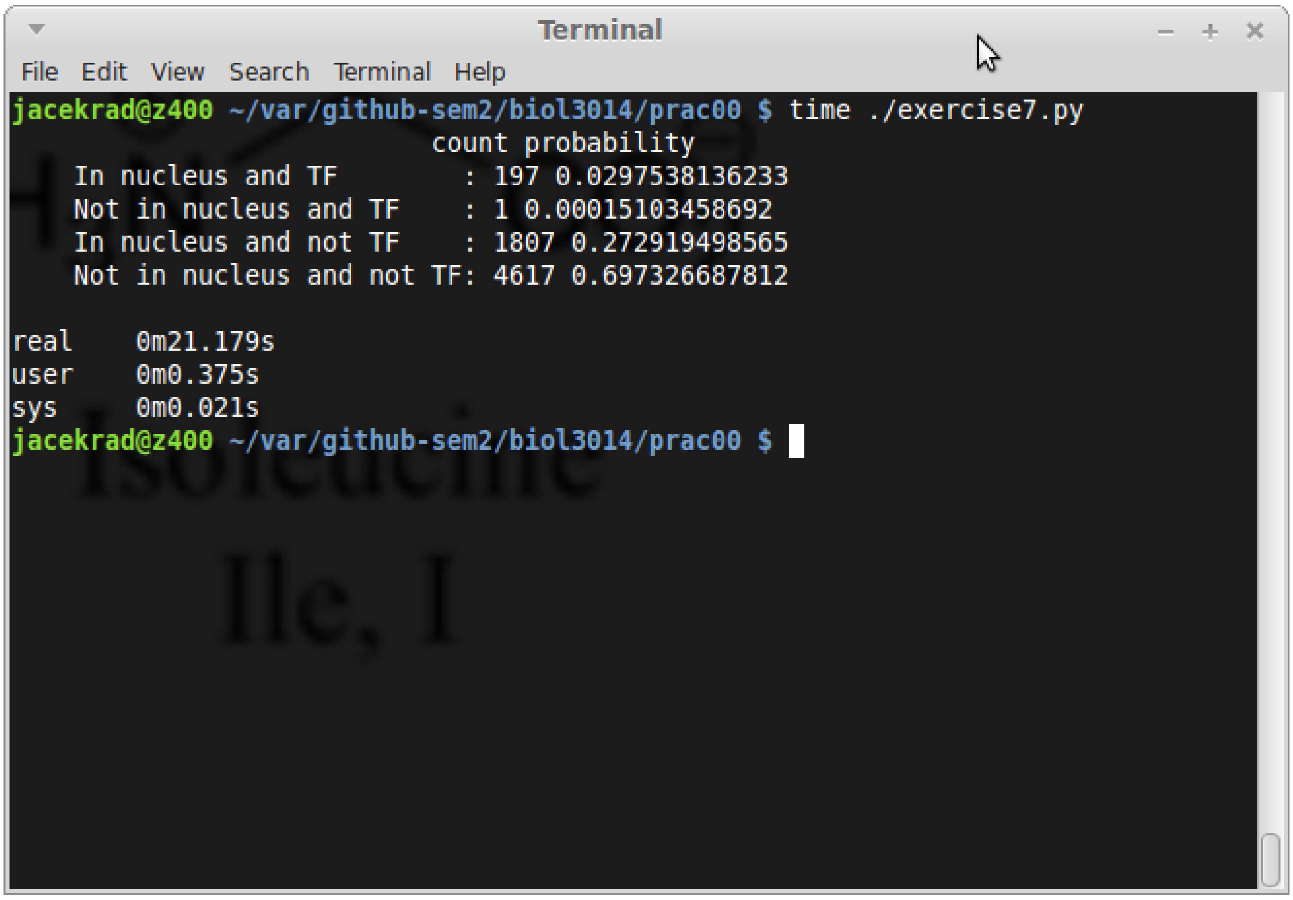
	Question 6 code	Question 7 code
T∩N	71	197
N\T	2	1
T\N	1816	1807
T'∩N'	4734	4617

An interesting to note is that processing the same data, exercise 6 code ran 66 times longer than exercise 7 code.

Source Code

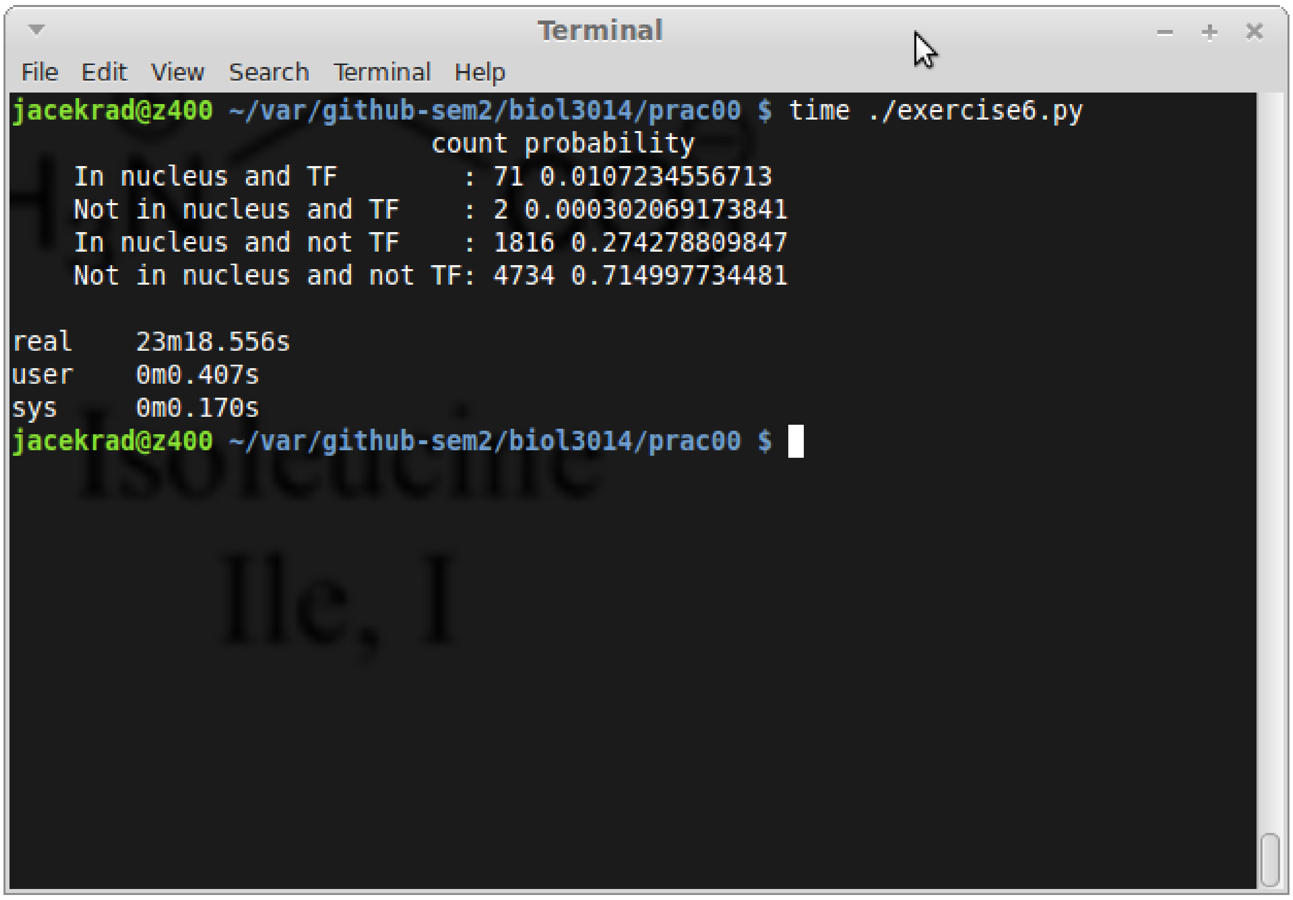
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
'''
Created on 03/08/2014

@author: jacekrad
'''
from webservice import getGenes

yeast_file = open("yeast_transcriptome.txt")
protein_identifiers = []  # list of protein IDs

# total number of proteins read from file
protein_count = 0

for line in yeast_file:
    protein_count += 1
    fields = line.strip().split("\t")
    protein_id = fields[0]
    protein_identifiers.append(protein_id)
yeast_file.close()

# get both GO terms so we only need to make single call to UniProt
genes = getGenes(["GO:0005634", "GO:0003700"], "UniProtKB", "559292")

all_in_nucleus_proteins = genes.get("GO:0005634")
all_tf_proteins = genes.get("GO:0003700")

# filter for entries which are both in the result genes from UniProtKB
# as well as in out file (protein_identifiers)
yeast_in_nucleus_proteins = [prot_id for prot_id in all_in_nucleus_proteins
\
                             if prot_id in protein_identifiers]

yeast_tf_proteins = [prot_id for prot_id in all_tf_proteins if prot_id \
                     in protein_identifiers]

yeast_in_nucleus_tf_proteins = [prot_id for prot_id in
protein_identifiers \
                                if prot_id in all_in_nucleus_proteins \
                                and prot_id in all_tf_proteins]

# get number of entries for each list
in_nucleus_count = len(yeast_in_nucleus_proteins)
tf_count = len(yeast_tf_proteins)
tf_in_nucleus_count = len(yeast_in_nucleus_tf_proteins)

# U - universal set (all proteins from file)
# T - set of transcription factor proteins
# N - set of in nucleus proteins
# T\N = T∩N' = T – T∩N
# N\T = N∩T' = N – T∩N
# T`∩N' = (T∪N)' = U - (T∪N) = U – T – N + T∩N
not_tf_nucleus_count = in_nucleus_count - tf_in_nucleus_count
tf_not_nucleus_count = tf_count - tf_in_nucleus_count
not_tf_not_nucleus_count = protein_count - tf_count - in_nucleus_count +
tf_count

print "                           count probability"
print "    In nucleus and TF        :", tf_in_nucleus_count, \
    float(tf_in_nucleus_count) / protein_count
print "    Not in nucleus and TF    :", tf_not_nucleus_count, \
    float(tf_not_nucleus_count) / protein_count
print "    In nucleus and not TF    :", not_tf_nucleus_count, \
    float(not_tf_nucleus_count) / protein_count
print "    Not in nucleus and not TF:", not_tf_not_nucleus_count, \
    float(not_tf_not_nucleus_count) / protein_count
```

Output



```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ time ./exercise7.py
                           count probability
    In nucleus and TF        : 197 0.0297538136233
    Not in nucleus and TF    : 1 0.00015103458692
    In nucleus and not TF    : 1807 0.272919498565
    Not in nucleus and not TF: 4617 0.697326687812

real    0m21.179s
user    0m0.375s
sys     0m0.021s
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $
```

Further to running code in exercise 7, I also modified exercise 6 to process the same, full, yeast_transcriptome.txt file. Result show below:



```
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $ time ./exercise6.py
                           count probability
    In nucleus and TF        : 71 0.0107234556713
    Not in nucleus and TF    : 2 0.000302069173841
    In nucleus and not TF    : 1816 0.274278809847
    Not in nucleus and not TF: 4734 0.714997734481

real    23m18.556s
user    0m0.407s
sys     0m0.170s
jacekrad@z400 ~/var/github-sem2/biol3014/prac00 $
```