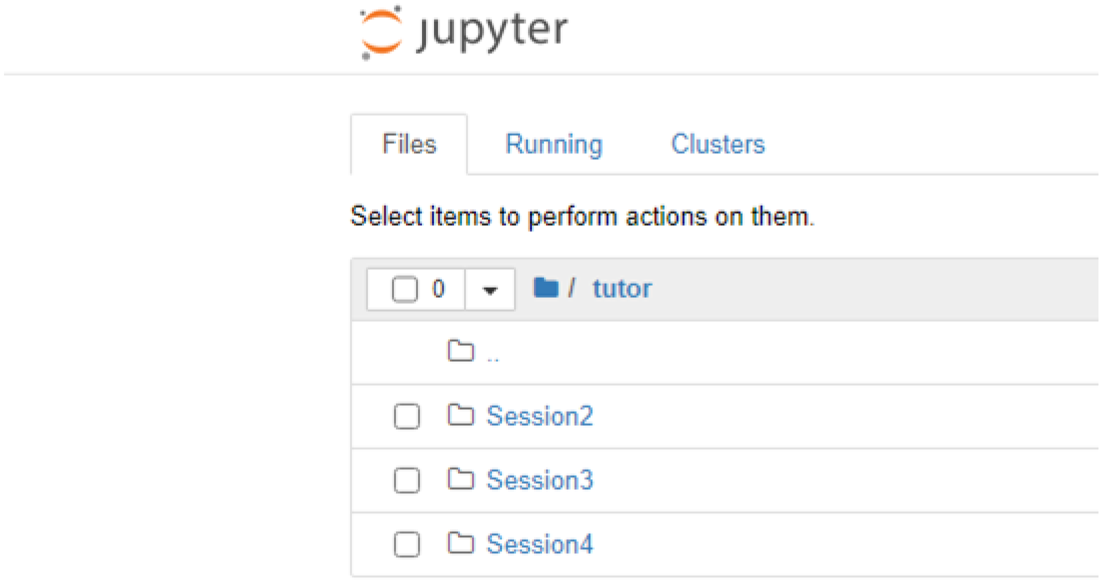
BIOINFORMATICS
SESSION 4. PRACTICE

2023-09-25

When DNA sequences are toxic

Contents

Make a Session4 directory



Select items to perform actions on them.

0 / tutor

..

Session2

Session3

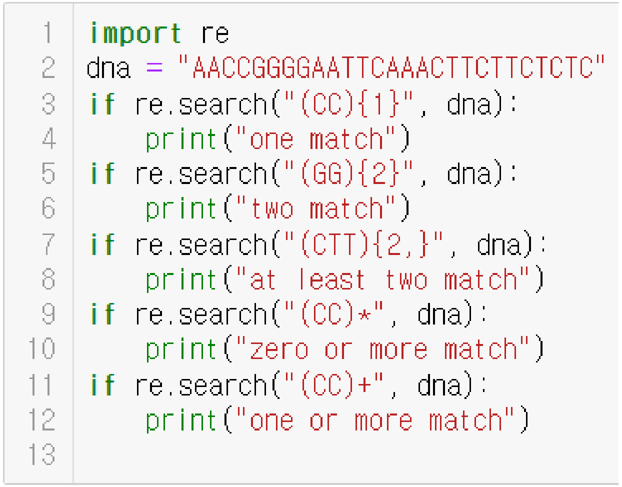
Session4

Basic Python – re module

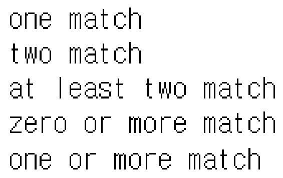
Method	Description
re.**search**(*pattern*, *string*, *flags=0*)	**Scan through *string* looking for <u>the first location where the regular expression *pattern* produces a match</u>**, and return a corresponding **MatchObject** instance. Return None if no position in the string matches the pattern; note that this is different from finding a zero-length match at some point in the string.
re.**match**(*pattern*, *string*, *flags=0*)	**If zero or more characters <u>at the beginning of *string*</u> match the regular expression *pattern***, return a corresponding **MatchObject** instance. Return None if the string does not match the pattern; note that this is different from a zero-length match.
re.**findall**(*pattern*, *string*, *flags=0*)	**Return <u>all non-overlapping matches of *pattern*</u> in *string*, <u>as a list of strings</u>**. The *string* is scanned left-to-right, and matches are returned in the order found. If one or more groups are present in the pattern, return a list of groups; this will be a list of tuples if the pattern has more than one group. Empty matches are included in the result unless they touch the beginning of another match.

Pattern matching – re module

```
rePattern1.ipynb
```

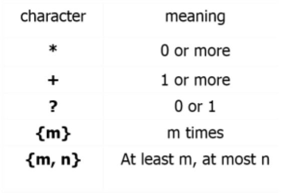


```python
import re
dna = "AACCGGGGAATTCAAACTTCTTCTCTC"
if re.search("(CC){1}", dna):
    print("one match")
if re.search("(GG){2}", dna):
    print("two match")
if re.search("(CTT){2,}", dna):
    print("at least two match")
if re.search("(CC)*", dna):
    print("zero or more match")
if re.search("(CC)+", dna):
    print("one or more match")
```



```
one match
two match
at least two match
zero or more match
one or more match
```

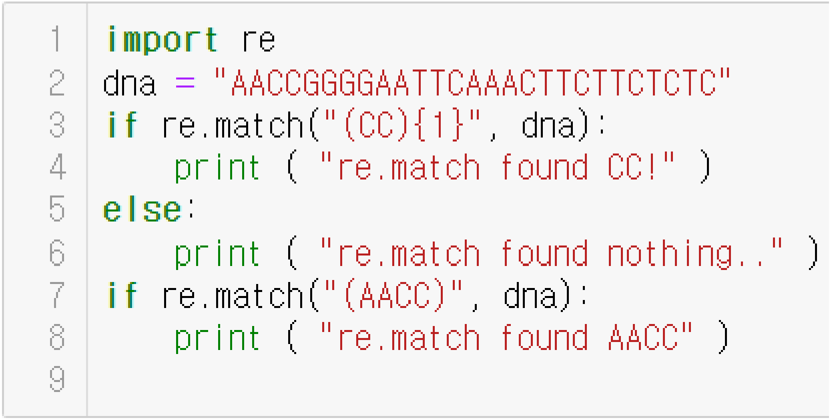
- [CT]: C or T
- [AB*]: A or AB or ABB or ABBB, …
- [AB+]: AB or ABB or ABBB,…
- [AB?]: A or AB
- A{6}: AAAAAA
- A{4,6}: AAAA, AAAAA, AAAAAA
- (CC) : a “group” or “capture”



character	meaning
*	0 or more
+	1 or more
?	0 or 1
{m}	m times
{m, n}	At least m, at most n

Pattern matching – re module

```
rePattern2.ipyb
```

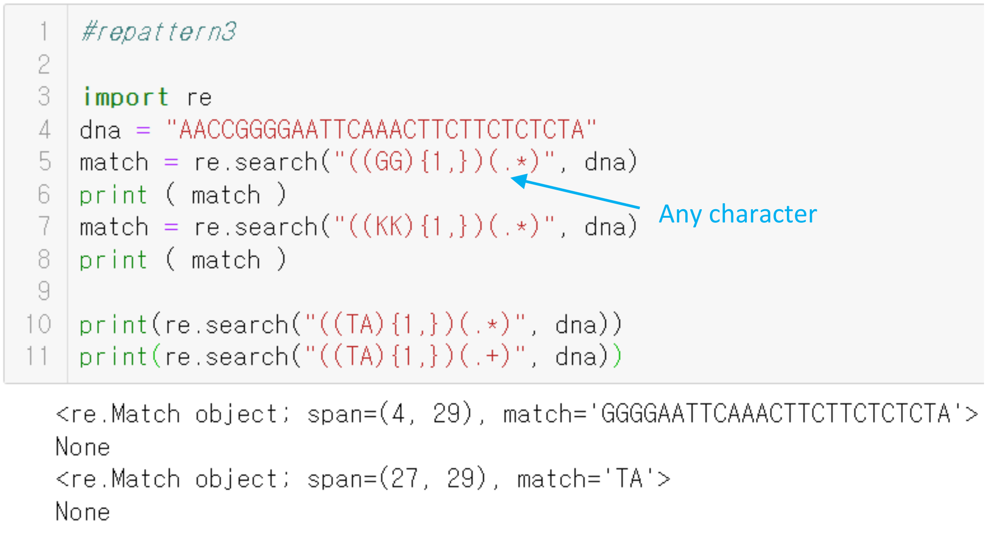


```python
import re
dna = "AACCGGGGAATTCAAACTTCTTCTCTC"
if re.match("(CC){1}", dna):
    print ( "re.match found CC!" )
else:
    print ( "re.match found nothing.." )
if re.match("(AACC)", dna):
    print ( "re.match found AACC" )

```

```
re.match found nothing..
re.match found AACC
```

Pattern matching – re module

```
rePattern3.ipynb
```



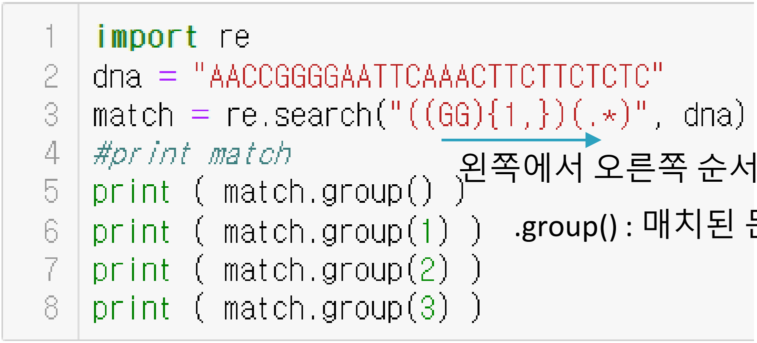
```
#repattern3

import re
dna = "AACCGGGGAATTCAAACTTCTTCTCTCTA"
match = re.search("((GG){1,})(.*)", dna)
print ( match )
match = re.search("((KK){1,})(.*)", dna)
print ( match )

print(re.search("((TA){1,})(.*)", dna))
print(re.search("((TA){1,})(.+)", dna))
```

Any character

```
<re.Match object; span=(4, 29), match='GGGGAATTCAAACTTCTTCTCTCTA'>
None
<re.Match object; span=(27, 29), match='TA'>
None
```

Pattern matching – re module

```
rePattern4.ipynb
```



```python
import re
dna = "AACCGGGGAATTCAAACTTCTTCTCTC"
match = re.search("((GG){1,})(.*)", dna)
#print match
print ( match.group() )
print ( match.group(1) )
print ( match.group(2) )
print ( match.group(3) )
```

왼쪽에서 오른쪽 순서로 배치 돼 있는 소괄호로 group구분

.group() : 매치된 문자열 반환

```
GGGGAATTCAAACTTCTTCTCTC
GGGG
GG
AATTCAAACTTCTTCTCTC
```

Match= re.search(“(pattern1)(pattern2)”, dna)
Match.group() -> pattern1+pattern2 #entire match
Match.group(1) -> pattern1 #1st pattern group match
Match.group(2) -> pattern2 #2nd pattern group match

Pattern matching – re module

```
rePattern5.ipynb
```

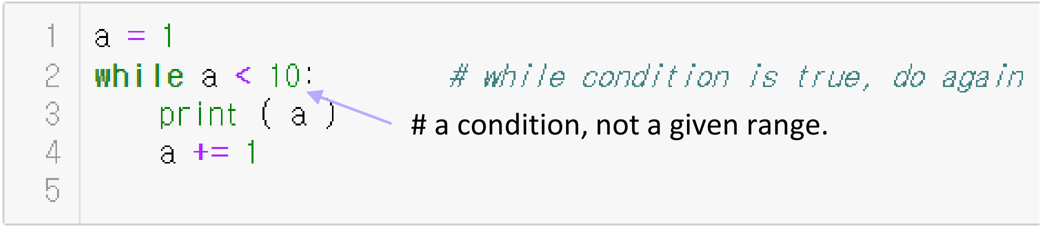
```python
import re
dna = "AACCGGGGAATTCAAACTTCTTCTCTC"
TC_all = re.findall("(TC)", dna)
TC_repeats = re.findall("((TC){2,})", dna)
TC_repeats2 = re.findall("(TT)(C){1,}", dna)
print ( TC_all )
print ( TC_repeats )
print ( TC_repeats2 )
```

```
['TC', 'TC', 'TC', 'TC', 'TC']
[('TCTCTC', 'TC')]
[('TT', 'C'), ('TT', 'C'), ('TT', 'C')]
```

re.search 처럼 왼쪽에서 오른쪽 소괄호에 해당하는 내용을 tuple에 저장

Conditional Expressions, While

```
whileloop.ipyb
```



```python
a = 1
while a < 10:           # while condition is true, do again
    print ( a )
    a += 1

```

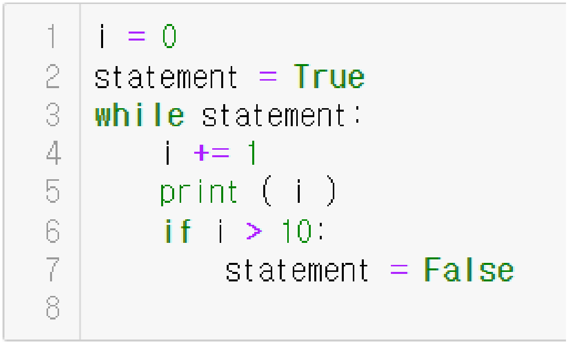
# a condition, not a given range.

```
1
2
3
4
5
6
7
8
9
```

Conditional Expressions, While

While loop

```
while_statement.ipynb
```

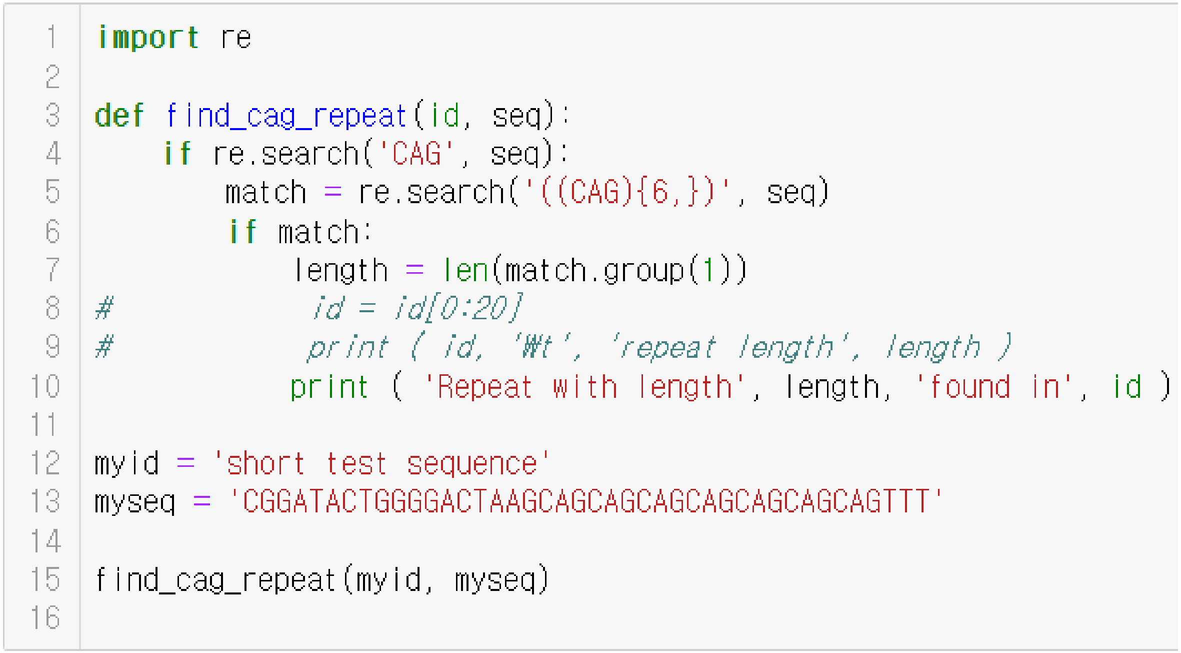


```python
i = 0
statement = True
while statement:
    i += 1
    print ( i )
    if i > 10:
        statement = False
```

```
1
2
3
4
5
6
7
8
9
10
11
```

Caution!
While loop을 잘못 만들면 무한 loop이 만들어 질 수 있음.
만약에 코드 실행이 종료 되지 않으면 강제로라도 코드 실행을 종료해야 함!
아니면 리소스를 무한정 사용하기 때문에 서버가 셧다운 될 수 있음

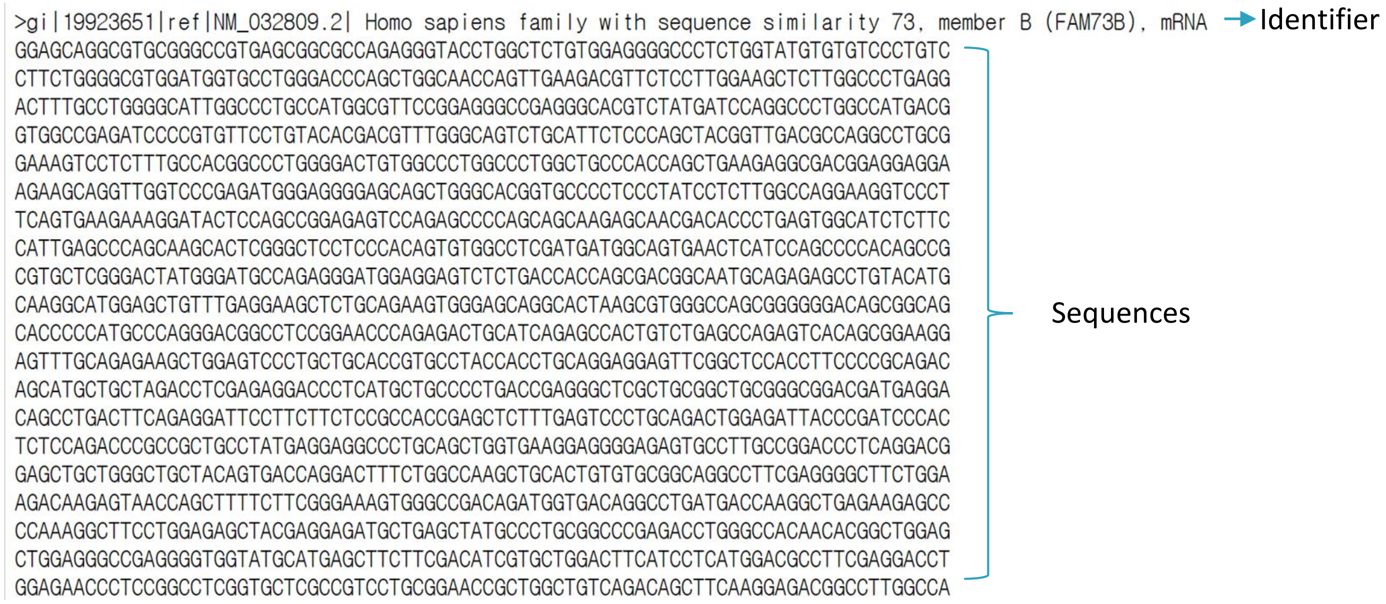
find_cag_short.ipynb



```python
import re

def find_cag_repeat(id, seq):
    if re.search('CAG', seq):
        match = re.search('((CAG){6,})', seq)
        if match:
            length = len(match.group(1))
#            id = id[0:20]
#            print ( id, 'Wt', 'repeat length', length )
            print ( 'Repeat with length', length, 'found in', id )

myid = 'short test sequence'
myseq = 'CGGATACTGGGGACTAAGCAGCAGCAGCAGCAGCAGCAGTTT'

find_cag_repeat(myid, myseq)

```

```
Repeat with length 21 found in short test sequence
```

refseq_human.txt



```
>gi|19923651|ref|NM_032809.2| Homo sapiens family with sequence similarity 73, member B (FAM73B), mRNA
GGAGCAGGCGTGCGGGCCGTGAGCGGCGCCAGAGGGTACCTGGCTCTGTGGAGGGGCCCTCTGGTATGTGTGTCCCTGTC
CTTCTGGGGCGTGGATGGTGCCTGGGACCCAGCTGGCAACCAGTTGAAGACGTTCTCCTTGGAAGCTCTTGGCCCTGAGG
ACTTTGCCTGGGGCATTGGCCCTGCCATGGCGTTCCGGAGGGCCGAGGGCACGTCTATGATCCAGGCCCTGGCCATGACG
GTGGCCGAGATCCCCGTGTTCCTGTACACGACGTTTGGGCAGTCTGCATTCTCCCAGCTACGGTTGACGCCAGGCCTGCG
GAAAGTCCTCTTTGCCACGGCCCTGGGGACTGTGGCCCTGGCCCTGGCTGCCCACCAGCTGAAGAGGCGACGGAGGAGGA
AGAAGCAGGTTGGTCCCGAGATGGGAGGGGAGCAGCTGGGCACGGTGCCCCTCCCTATCCTCTTGGCCAGGAAGGTCCCT
TCAGTGAAGAAAGGATACTCCAGCCGGAGAGTCCAGAGCCCCAGCAGCAAGAGCAACGACACCCTGAGTGGCATCTCTTC
CATTGAGCCCAGCAAGCACTCGGGCTCCTCCCACAGTGTGGCCTCGATGATGGCAGTGAACTCATCCAGCCCCACAGCCG
CGTGCTCGGGACTATGGGATGCCAGAGGGATGGAGGAGTCTCTGACCACCAGCGACGGCAATGCAGAGAGCCTGTACATG
CAAGGCATGGAGCTGTTTGAGGAAGCTCTGCAGAAGTGGGAGCAGGCACTAAGCGTGGGCCAGCGGGGGGACAGCGGCAG
CACCCCCATGCCCAGGGACGGCCTCCGGAACCCAGAGACTGCATCAGAGCCACTGTCTGAGCCAGAGTCACAGCGGAAGG
AGTTTGCAGAGAAGCTGGAGTCCCTGCTGCACCGTGCCTACCACCTGCAGGAGGAGTTCGGCTCCACCTTCCCCGCAGAC
AGCATGCTGCTAGACCTCGAGAGGACCCTCATGCTGCCCCTGACCGAGGGCTCGCTGCGGCTGCGGGCGGACGATGAGGA
CAGCCTGACTTCAGAGGATTCCTTCTTCTCCGCCACCGAGCTCTTTGAGTCCCTGCAGACTGGAGATTACCCGATCCCAC
TCTCCAGACCCGCCGCTGCCTATGAGGAGGCCCTGCAGCTGGTGAAGGAGGGGAGAGTGCCTTGCCGGACCCTCAGGACG
GAGCTGCTGGGCTGCTACAGTGACCAGGACTTTCTGGCCAAGCTGCACTGTGTGCGGCAGGCCTTCGAGGGGCTTCTGGA
AGACAAGAGTAACCAGCTTTTCTTCGGGAAAGTGGGCCGACAGATGGTGACAGGCCTGATGACCAAGGCTGAGAAGAGCC
CCAAAGGCTTCCTGGAGAGCTACGAGGAGATGCTGAGCTATGCCCTGCGGCCCGAGACCTGGGCCACAACACGGCTGGAG
CTGGAGGGCCGAGGGGTGGTATGCATGAGCTTCTTCGACATCGTGCTGGACTTCATCCTCATGGACGCCTTCGAGGACCT
GGAGAACCCTCCGGCCTCGGTGCTCGCCGTCCTGCGGAACCGCTGGCTGTCAGACAGCTTCAAGGAGACGGCCTTGGCCA
```

reading and writing files

```
#reading file
input_file = open(file_name, ‘r’)

#writing file
output_file = open(outfile_name, ‘w’)
output_file.write(xxx)
```

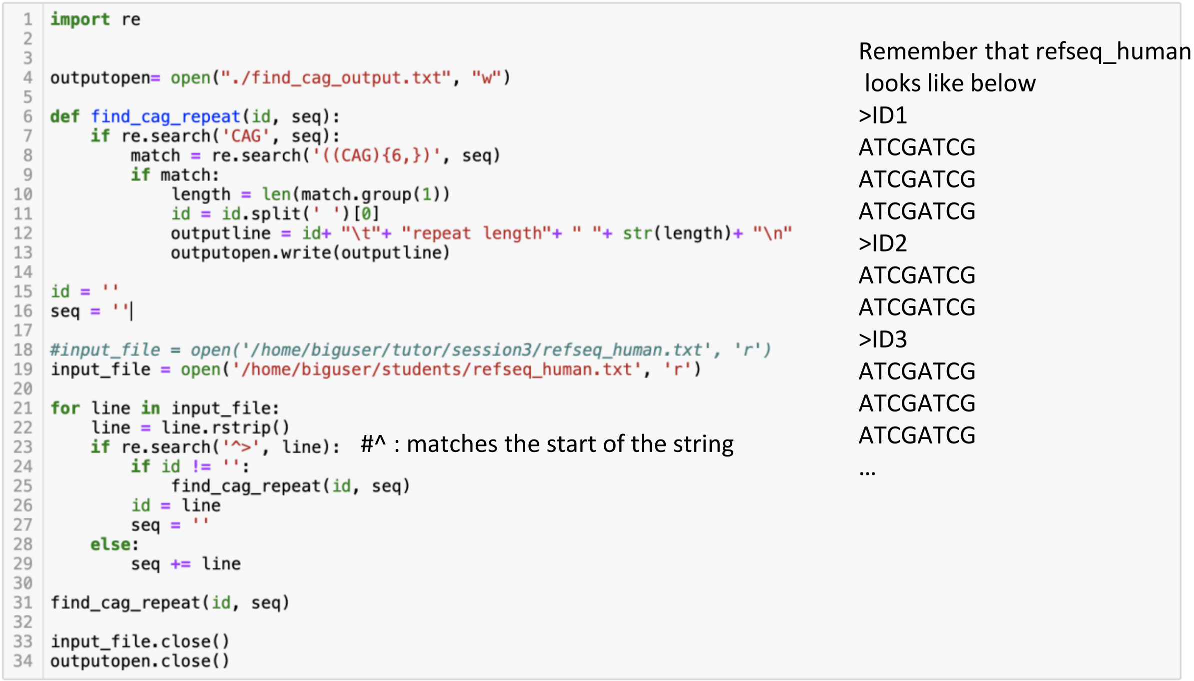
*caution
plese close the input and output file in your code

```
input_file.close()
output_file.close()
```

Remember that refseq_human.txt file looks like below

```
>ID1
ATCGATCG
ATCGATCG
ATCGATCG
>ID2
ATCGATCG
ATCGATCG
>ID3
ATCGATCG
ATCGATCG
ATCGATCG
…
```

find_cag.ipynb

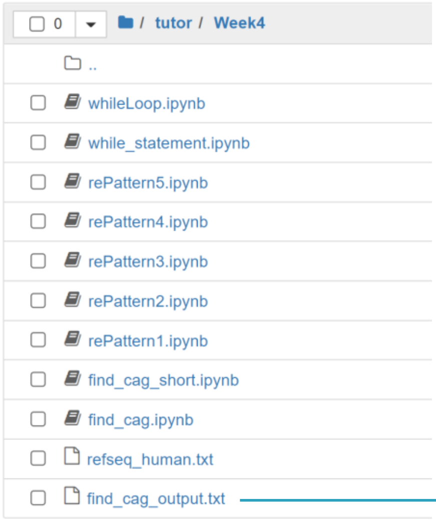


```python
import re


outputopen= open("./find_cag_output.txt", "w")

def find_cag_repeat(id, seq):
    if re.search('CAG', seq):
        match = re.search('((CAG){6,})', seq)
        if match:
            length = len(match.group(1))
            id = id.split(' ')[0]
            outputline = id+ "\t"+ "repeat length"+ " "+ str(length)+ "\n"
            outputopen.write(outputline)

id = ''
seq = ''

#input_file = open('/home/biguser/tutor/session3/refseq_human.txt', 'r')
input_file = open('/home/biguser/students/refseq_human.txt', 'r')

for line in input_file:
    line = line.rstrip()
    if re.search('^>', line):   #^ : matches the start of the string
        if id != '':
            find_cag_repeat(id, seq)
        id = line
        seq = ''
    else:
        seq += line

find_cag_repeat(id, seq)

input_file.close()
outputopen.close()
```

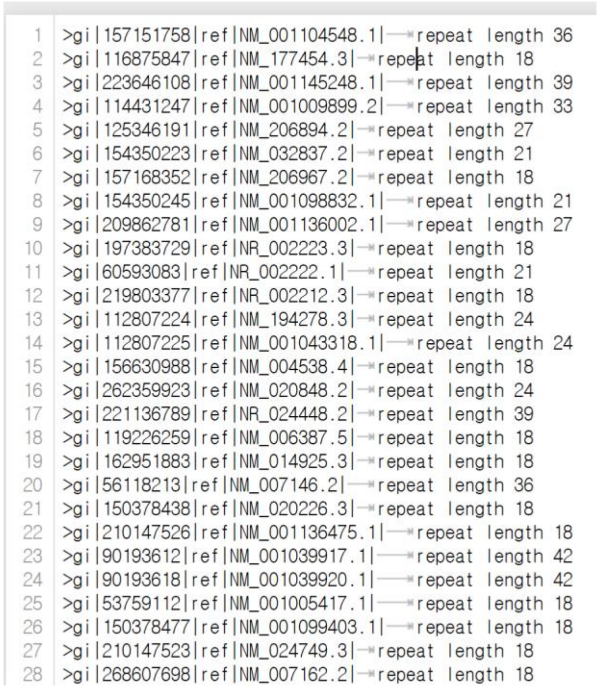
Remember that refseq_human.txt file looks like below

```
>ID1
ATCGATCG
ATCGATCG
ATCGATCG
>ID2
ATCGATCG
ATCGATCG
>ID3
ATCGATCG
ATCGATCG
ATCGATCG
…
```

find_cag_output.txt



```
whileLoop.ipynb
while_statement.ipynb
rePattern5.ipynb
rePattern4.ipynb
rePattern3.ipynb
rePattern2.ipynb
rePattern1.ipynb
find_cag_short.ipynb
find_cag.ipynb
refseq_human.txt
find_cag_output.txt
```



```
>gi|157151758|ref|NM_001104548.1|	repeat length 36
>gi|116875847|ref|NM_177454.3|	repeat length 18
>gi|223646108|ref|NM_001145248.1|	repeat length 39
>gi|114431247|ref|NM_001009899.2|	repeat length 33
>gi|125346191|ref|NM_206894.2|	repeat length 27
>gi|154350223|ref|NM_032837.2|	repeat length 21
>gi|157168352|ref|NM_206967.2|	repeat length 18
>gi|154350245|ref|NM_001098832.1|	repeat length 21
>gi|209862781|ref|NM_001136002.1|	repeat length 27
>gi|197383729|ref|NR_002223.3|	repeat length 18
>gi|60593083|ref|NR_002222.1|	repeat length 21
>gi|219803377|ref|NR_002212.3|	repeat length 18
>gi|112807224|ref|NM_194278.3|	repeat length 24
>gi|112807225|ref|NM_001043318.1|	repeat length 24
>gi|156630988|ref|NM_004538.4|	repeat length 18
>gi|262359923|ref|NM_020848.2|	repeat length 24
>gi|221136789|ref|NR_024448.2|	repeat length 39
>gi|119226259|ref|NM_006387.5|	repeat length 18
>gi|162951883|ref|NM_014925.3|	repeat length 18
>gi|56118213|ref|NM_007146.2|	repeat length 36
>gi|150378438|ref|NM_020226.3|	repeat length 18
>gi|210147526|ref|NM_001136475.1|	repeat length 18
>gi|90193612|ref|NM_001039917.1|	repeat length 42
>gi|90193618|ref|NM_001039920.1|	repeat length 42
>gi|53759112|ref|NM_001005417.1|	repeat length 18
>gi|150378477|ref|NM_001099403.1|	repeat length 18
>gi|210147523|ref|NM_024749.3|	repeat length 18
>gi|268607698|ref|NM_007162.2|	repeat length 18
```

Exercise

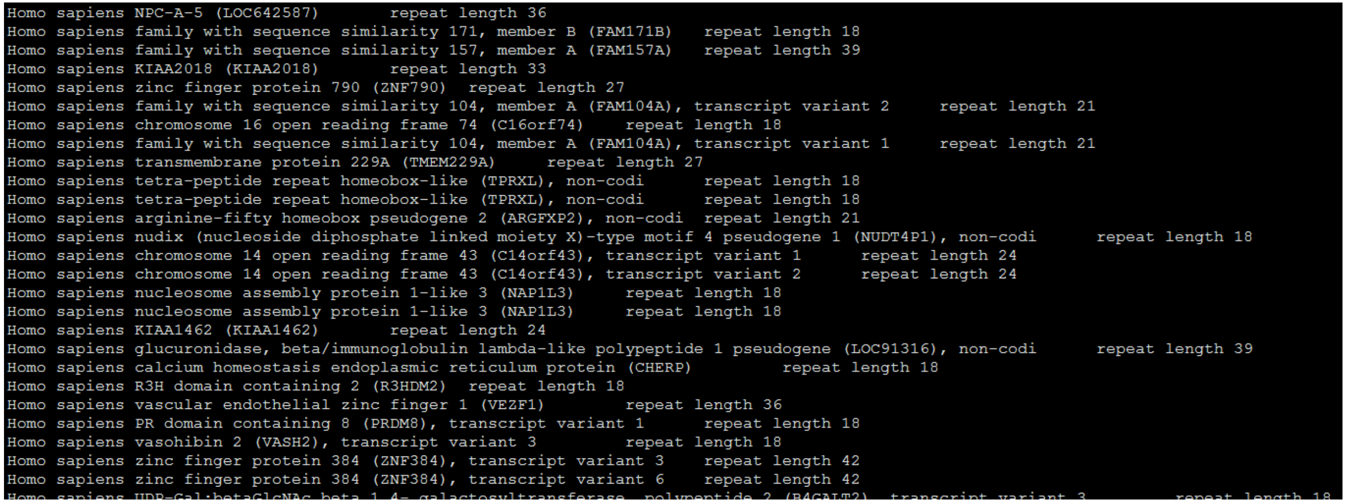
- In the collection of RefSeq sequences used in find_cag.py (refseq_human.txt), mRNAs have their functional description. Modify the code so that this information, instead of the identifier, is reported in your output.

- Are there proteins in the output that are described as ‘huntingtin’? If so, print out the corresponding protein ID

```
If Protein_id.find(“huntingtin”)>=0:
    print(Protein_id)
```

- Huntingtin 관련 protein들이 정상적으로 출력되는지 확인할 것 !
  - Homo sapiens huntingtin (HTT)

Example of correct output



```
Homo sapiens NPC-A-5 (LOC642587)       repeat length 36
Homo sapiens family with sequence similarity 171, member B (FAM171B)    repeat length 18
Homo sapiens family with sequence similarity 157, member A (FAM157A)    repeat length 39
Homo sapiens KIAA2018 (KIAA2018)       repeat length 33
Homo sapiens zinc finger protein 790 (ZNF790)  repeat length 27
Homo sapiens family with sequence similarity 104, member A (FAM104A), transcript variant 2     repeat length 21
Homo sapiens chromosome 16 open reading frame 74 (C16orf74)    repeat length 18
Homo sapiens family with sequence similarity 104, member A (FAM104A), transcript variant 1     repeat length 21
Homo sapiens transmembrane protein 229A (TMEM229A)     repeat length 27
Homo sapiens tetra-peptide repeat homeobox-like (TPRXL), non-codi      repeat length 18
Homo sapiens tetra-peptide repeat homeobox-like (TPRXL), non-codi      repeat length 18
Homo sapiens arginine-fifty homeobox pseudogene 2 (ARGFXP2), non-codi  repeat length 21
Homo sapiens nudix (nucleoside diphosphate linked moiety X)-type motif 4 pseudogene 1 (NUDT4P1), non-codi      repeat length 18
Homo sapiens chromosome 14 open reading frame 43 (C14orf43), transcript variant 1      repeat length 24
Homo sapiens chromosome 14 open reading frame 43 (C14orf43), transcript variant 2      repeat length 24
Homo sapiens nucleosome assembly protein 1-like 3 (NAP1L3)     repeat length 18
Homo sapiens nucleosome assembly protein 1-like 3 (NAP1L3)     repeat length 18
Homo sapiens KIAA1462 (KIAA1462)       repeat length 24
Homo sapiens glucuronidase, beta/immunoglobulin lambda-like polypeptide 1 pseudogene (LOC91316), non-codi      repeat length 39
Homo sapiens calcium homeostasis endoplasmic reticulum protein (CHERP)         repeat length 18
Homo sapiens R3H domain containing 2 (R3HDM2)  repeat length 18
Homo sapiens vascular endothelial zinc finger 1 (VEZF1)        repeat length 36
Homo sapiens PR domain containing 8 (PRDM8), transcript variant 1      repeat length 18
Homo sapiens vasohibin 2 (VASH2), transcript variant 3         repeat length 18
Homo sapiens zinc finger protein 384 (ZNF384), transcript variant 3    repeat length 42
Homo sapiens zinc finger protein 384 (ZNF384), transcript variant 6    repeat length 42
Homo sapiens UDP-Gal:betaGlcNAc beta 1,4- galactosyltransferase, polypeptide 2 (B4GALT2), transcript variant 3         repeat length 18
```