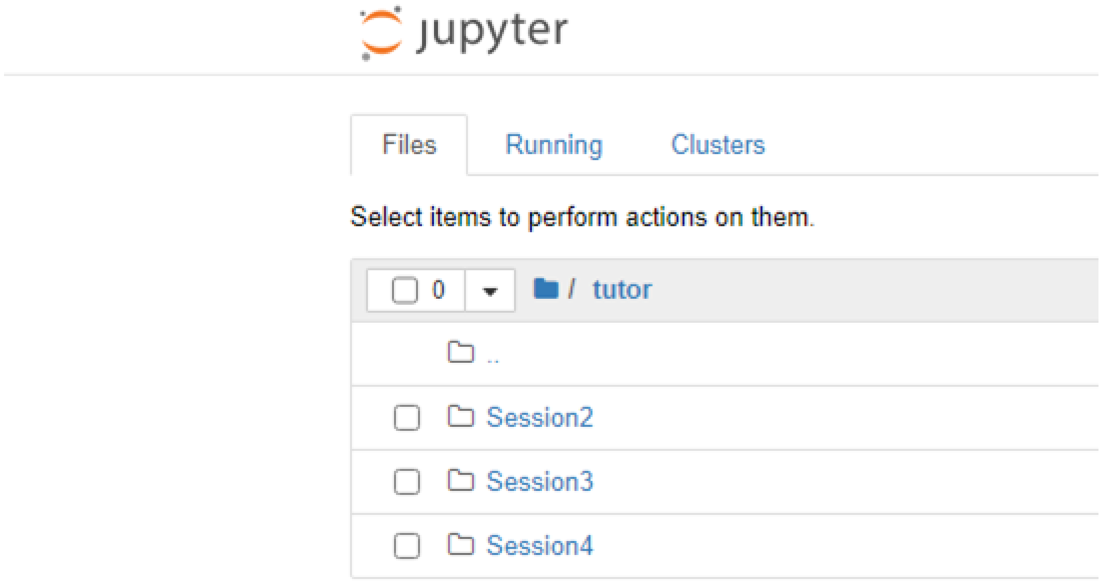
BIOINFORMATICS
SESSION 5. PRACTICE

2023-10-09

Iron imbalance and the iron responsive element

Make a Session5 directory

http://166.104.118.163:20002/



Select items to perform actions on them.

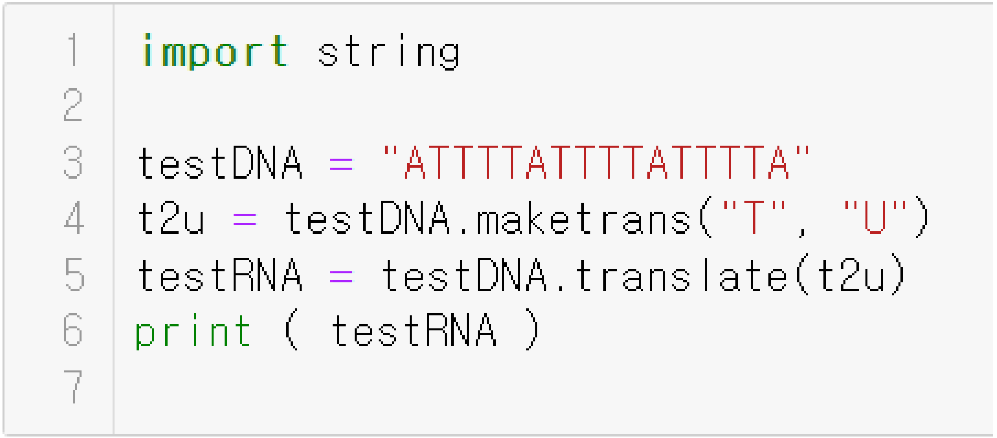
Session2

Session3

Session4

Replace Character(s) in String

stringModule.ipynb



```python
import string

testDNA = "ATTTTATTTTATTTTA"
t2u = testDNA.maketrans("T", "U")
testRNA = testDNA.translate(t2u)
print ( testRNA )

```

```
AUUUUAUUUUAUUUUA
```

Continued from stringModule.ipynb

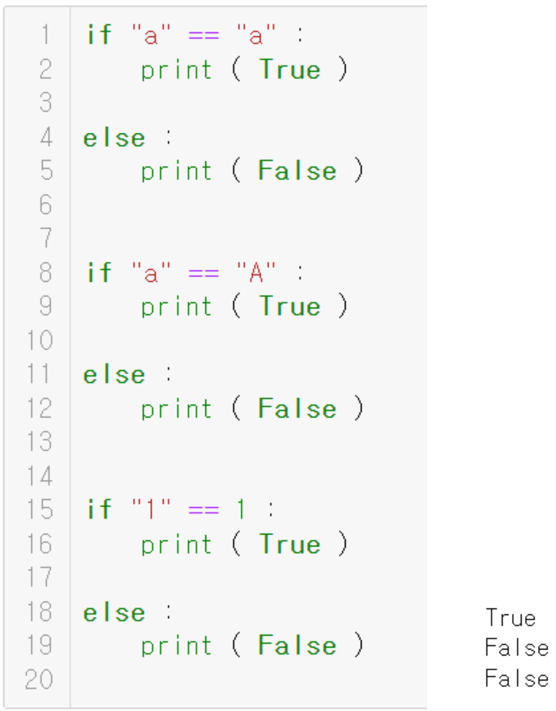
```python
testRNA2 = testDNA.replace('T', 'U')
print ( testRNA2 )

testRNA3 = testDNA.replace('T', 'U', 4)
print ( testRNA3 )

```

count

```
AUUUUAUUUUAUUUUA
AUUUUATTTTATTTTA
```

String comparison

strcmp.ipynb



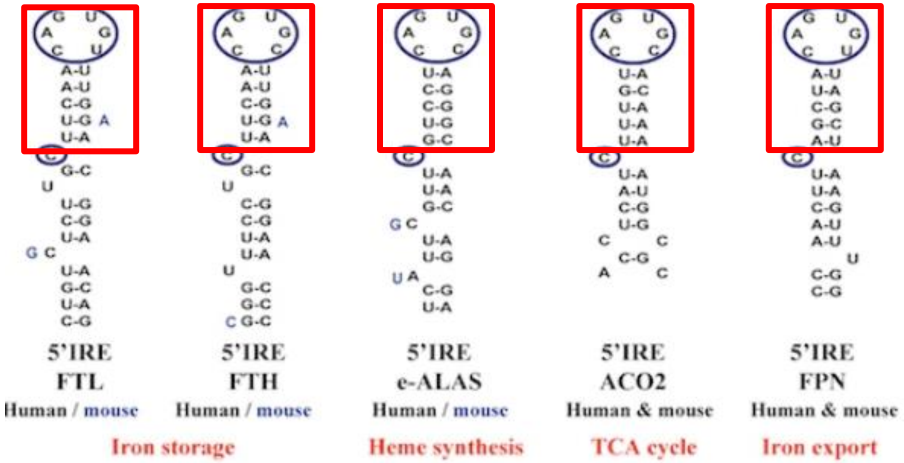
```
if "a" == "a" :
    print ( True )

else :
    print ( False )


if "a" == "A" :
    print ( True )

else :
    print ( False )


if "1" == 1 :
    print ( True )

else :
    print ( False )

```

```
True
False
False
```

ire.py

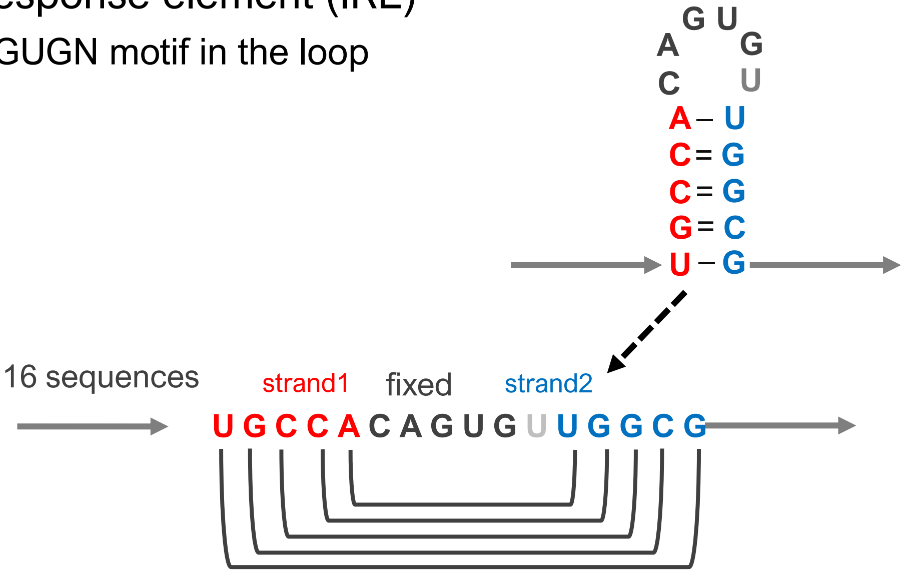
- Iron response element (IRE)
  - CAGUGN motif in the loop

checking 16 nucleotide sequences for today's practice

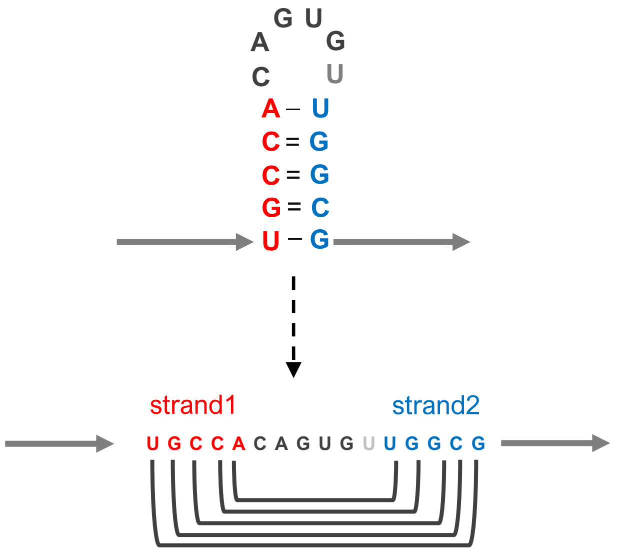


ire.py

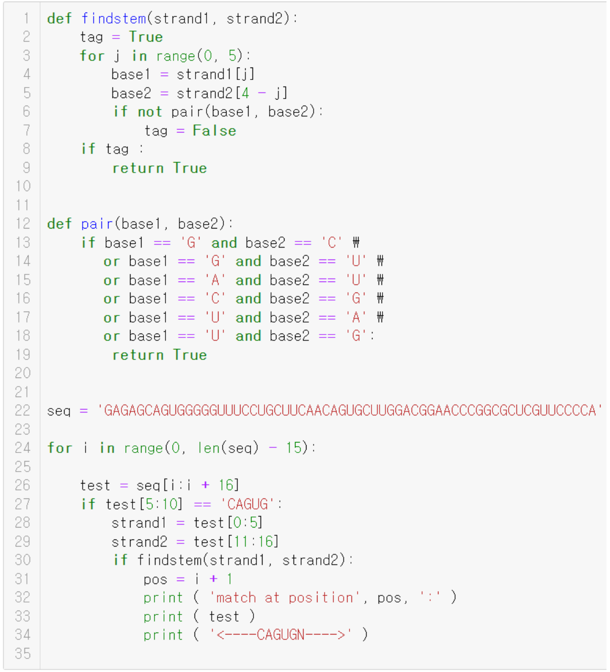
- Iron response element (IRE)
  - CAGUGN motif in the loop



ire.py



16 sequences

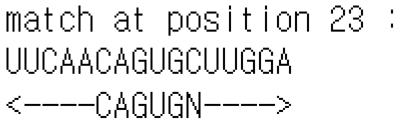


```
def findstem(strand1, strand2):
    tag = True
    for j in range(0, 5):
        base1 = strand1[j]
        base2 = strand2[4 - j]
        if not pair(base1, base2):
            tag = False
    if tag :
        return True


def pair(base1, base2):
    if base1 == 'G' and base2 == 'C' ₩
       or base1 == 'G' and base2 == 'U' ₩
       or base1 == 'A' and base2 == 'U' ₩
       or base1 == 'C' and base2 == 'G' ₩
       or base1 == 'U' and base2 == 'A' ₩
       or base1 == 'U' and base2 == 'G':
        return True


seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA'

for i in range(0, len(seq) - 15):

    test = seq[i:i + 16]
    if test[5:10] == 'CAGUG':
        strand1 = test[0:5]
        strand2 = test[11:16]
        if findstem(strand1, strand2):
            pos = i + 1
            print ( 'match at position', pos, ':' )
            print ( test )
            print ( '<----CAGUGN---->' )

```

```
match at position 23 :
UUCAACAGUGCUUGGA
<----CAGUGN---->
```

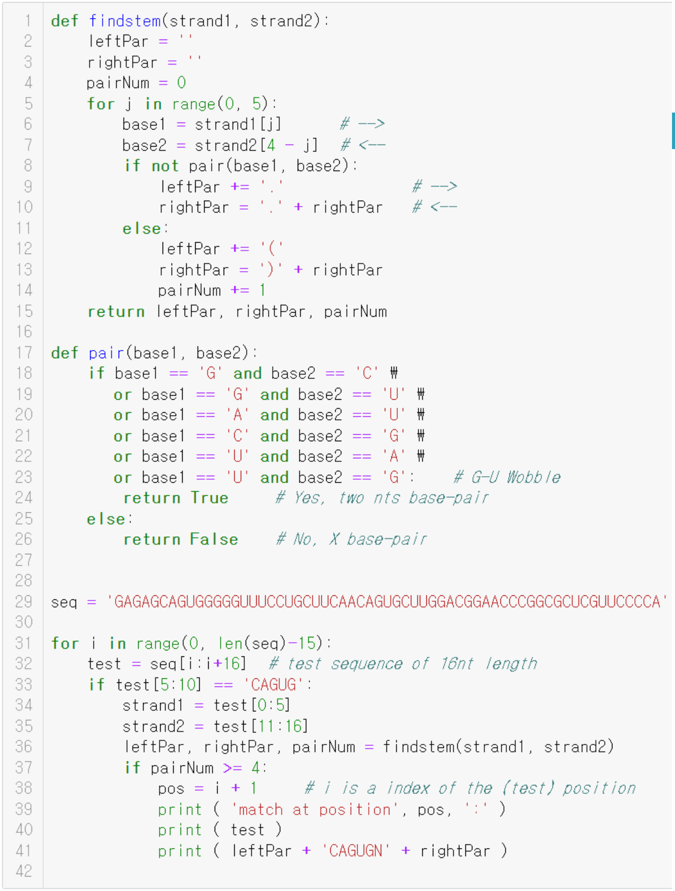
Seq= “GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA”

ire.py result



```
match at position 23 :
UUCAACAGUGCUUGGA
<----CAGUGN---->
```

ire2.ipynb



```python
def findstem(strand1, strand2):
    leftPar = ''
    rightPar = ''
    pairNum = 0
    for j in range(0, 5):
        base1 = strand1[j]       # -->
        base2 = strand2[4 - j]   # <--
        if not pair(base1, base2):
            leftPar += '.'              # -->
            rightPar = '.' + rightPar   # <--
        else:
            leftPar += '('
            rightPar = ')' + rightPar
            pairNum += 1
    return leftPar, rightPar, pairNum

def pair(base1, base2):
    if base1 == 'G' and base2 == 'C' ₩
       or base1 == 'G' and base2 == 'U' ₩
       or base1 == 'A' and base2 == 'U' ₩
       or base1 == 'C' and base2 == 'G' ₩
       or base1 == 'U' and base2 == 'A' ₩
       or base1 == 'U' and base2 == 'G':    # G-U Wobble
        return True      # Yes, two nts base-pair
    else:
        return False     # No, X base-pair


seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA'

for i in range(0, len(seq)-15):
    test = seq[i:i+16]   # test sequence of 16nt length
    if test[5:10] == 'CAGUG':
        strand1 = test[0:5]
        strand2 = test[11:16]
        leftPar, rightPar, pairNum = findstem(strand1, strand2)
        if pairNum >= 4:
            pos = i + 1      # i is a index of the (test) position
            print ( 'match at position', pos, ':' )
            print ( test )
            print ( leftPar + 'CAGUGN' + rightPar )
```

```
match at position 23 :
UUCAACAGUGCUUGGA
(((((CAGUGN)))))
```

Exercise 1

- There are many ways to solve a problem in programming. Consider the construction in ‘ire2.py’ with a number of operations with substring functions, such as:

```
        if test[5:10] == ‘CAGUG’:
```

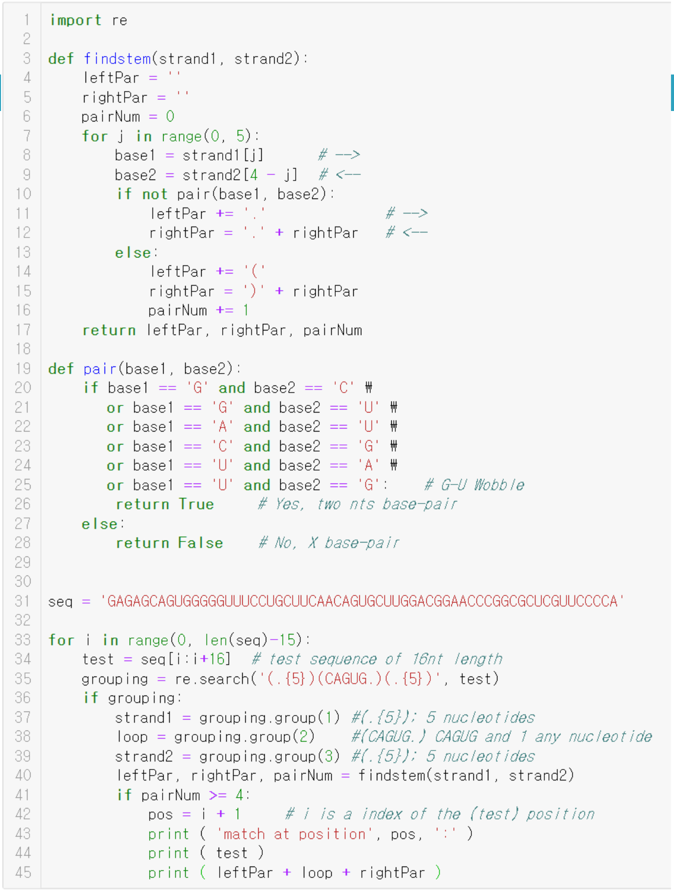
However, we could instead make use of a regular expression:

```
        if re.search(‘(     )(              )(     )’, test)
```

In this expression, we may capture not only **a loop sequence** as group(2), but also **strand1** and **strand2** variables as group(1) and group(3), respectively.

Modify ‘ire2.py’ to use this type of regular expression

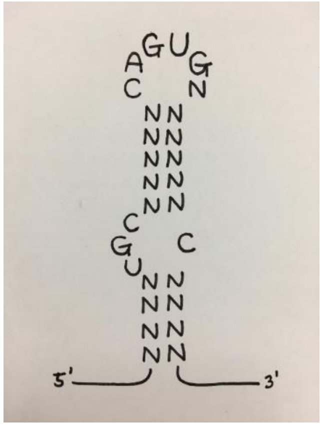
Exercise 1answer



```python
import re

def findstem(strand1, strand2):
    leftPar = ''
    rightPar = ''
    pairNum = 0
    for j in range(0, 5):
        base1 = strand1[j]      # -->
        base2 = strand2[4 - j]  # <--
        if not pair(base1, base2):
            leftPar += '.'              # -->
            rightPar = '.' + rightPar   # <--
        else:
            leftPar += '('
            rightPar = ')' + rightPar
            pairNum += 1
    return leftPar, rightPar, pairNum

def pair(base1, base2):
    if base1 == 'G' and base2 == 'C' ₩
       or base1 == 'G' and base2 == 'U' ₩
       or base1 == 'A' and base2 == 'U' ₩
       or base1 == 'C' and base2 == 'G' ₩
       or base1 == 'U' and base2 == 'A' ₩
       or base1 == 'U' and base2 == 'G':    # G-U Wobble
        return True      # Yes, two nts base-pair
    else:
        return False     # No, X base-pair


seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA'

for i in range(0, len(seq)-15):
    test = seq[i:i+16]  # test sequence of 16nt length
    grouping = re.search('(.{5})(CAGUG.)(.{5})', test)
    if grouping:
        strand1 = grouping.group(1) #(.{5}): 5 nucleotides
        loop = grouping.group(2)    #(CAGUG.) CAGUG and 1 any nucleotide
        strand2 = grouping.group(3) #(.{5}): 5 nucleotides
        leftPar, rightPar, pairNum = findstem(strand1, strand2)
        if pairNum >= 4:
            pos = i + 1      # i is a index of the (test) position
            print ( 'match at position', pos, ':' )
            print ( test )
            print ( leftPar + loop + rightPar )
```

Assignment

- Modify your ire.ipynb to search the complete iron response elements (IREs) in the 'refseq_human.txt' using regular expressions.
- Note that sequences in the 'refseq_human.txt' is DNA sequences ("T" in 'refseq_human.txt" should be replaced by "U")



- 오른쪽 그림과 같이 28nt(16nt 아님!) IRE구조를 가진 gene id, sequence를 refseq_human.txt에서 regular expression (re.search)를 이용하여 찾아보기.
- !주의! refseq_human.txt는 'T'로 되어있으니 'U'로 변환해야함.

- 과제 제출 기한: 10/15 Sunday 23:59 @ LMS
- 작성한 코드와 해당 코드의 결과를 캡처한 뒤 워드에 첨부(코드만 긁어와서 붙여넣지 말기), 코드에 대한 설명 간략히 작성
  워드 파일명은 n주차_학번_이름 형식으로 제출(e.g. 5주차_2023123456_김현우)