객체(object)란?

- 파이썬: object-oriented Programming
- 파이썬의 클래스(class):
  - 실세계의 어떤 사물과 상황도 클래스를 이용하여 모델링할 수 있음
- 파이썬의 객체(object):
  - 위에서 정의된 클래스에 근거하여 객체를 생성

모듈(modules) vs. 클래스(classes): from module import class1, class2, class3…

- Importing a Single Class
- Storing Multiple Classes in a Module
- Importing Multiple Classes from a Module
- Importing an Entire Module
- Importing all classes from a Module

클래스가 있어야 객체가 존재한다.

From module import *: 모듈내의 모든 클래스를 불러온다.

The python standard library

- A set of modules included with every Python installation.
- We can start to use modules that other programmers have written
- We can use any function or class in the standard library by including a simple import statement at the top of your file.
- One resource for exploring the python standard library is a site called python module of the week. Goto ….pymotw.com

Sequence Objects

- The most important difference between Seq objects and standard Python strings is they have different methods
- Sequences act like strings:
  - Enumerate, Index, Count, Len, Slicing a sequence

- Sequences act like strings:
    - Concatenating or adding sequences
    - Changing case
- Nucleotide sequences and (reverse) complements
- Transcription
    - Dna sequence 를 RNA sequence로 전사해줌. transcribe()
    - (T->U)
    - Back transcription: mrna -> the coding strand of the dna로 전사(u->t)
- Translation
    - Dna sequence or Rna sequence를 단백질 시퀀스로 번역해줌. Translate()

```
from Bio.Seq import Seq

my_seq = Seq("GATCG")

for index, letter in enumerate(my_seq):

    print("%i %s" %(index, letter))

print(my_seq[0]) >> G

print(my_seq[-1]) >> G

print(my_seq[-3]) >> T

print(my_seq[-5]) >> G
```

```
from Bio.Seq import Seq

print("AAAA".count("AA")) # AA가 몇 개인지? >> 2

print(Seq("AAAA").count("AA")) >> 2
```

```
from Bio.Seq import Seq

my_seq = Seq(“GATCGTACGATCGAT”)

print(len(my_seq))

print(my_seq.count(“G”)) # how many Gs

print(my_seq.count(“C”)) # how many Cs

print(100* float(my_seq.count(“G”) + my_seq.count(“C”)) / len(my_seq))
```

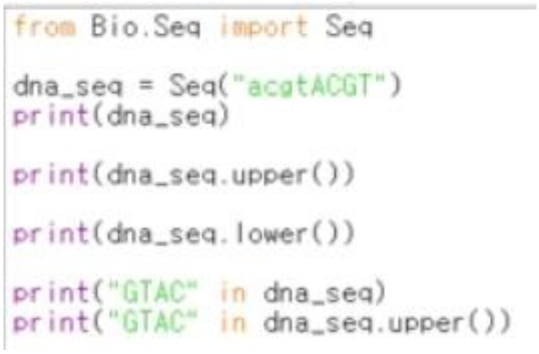
```
from Bio.Seq import Seq

from Bio.SeqUtils import GC

my_seq = Seq(“GACTAGCTAGCGGGATTTCGAGAAC”)

print(GC(my_Seq))# (G개수+C 개수) / 길이
```

<슬라이싱>

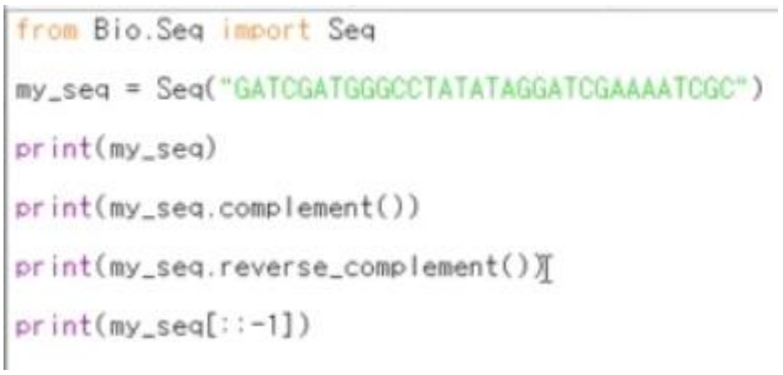
```
From Bio.Seq import Seq

My_seq = Seq(“GACTATTTAGCTTGGACAACCCGTA”)

Print(my_seq[4:12])#4부터 12

Print(my_seq[0::3])# 0부터 시작해서 3칸씩 띄고 출력

Print(my_seq[::-1]) # reverse 거꾸로 출력
```

<concatenating>

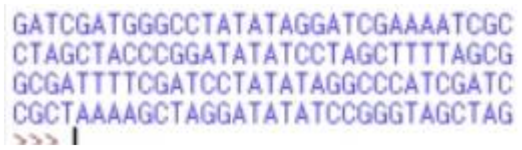
```
Pro_seq   = Seq(“ASGDF”)

Seq = Seq(“dkagjfd”)

Print(Pro_Seq + Seq) -> “ASGDFdkagjfd
```



```python
from Bio.Seq import Seq

dna_seq = Seq("acgtACGT")
print(dna_seq)

print(dna_seq.upper())

print(dna_seq.lower())

print("GTAC" in dna_seq)
print("GTAC" in dna_seq.upper())
```



```python
from Bio.Seq import Seq

my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC")

print(my_seq)

print(my_seq.complement())

print(my_seq.reverse_complement())

print(my_seq[::-1])
```

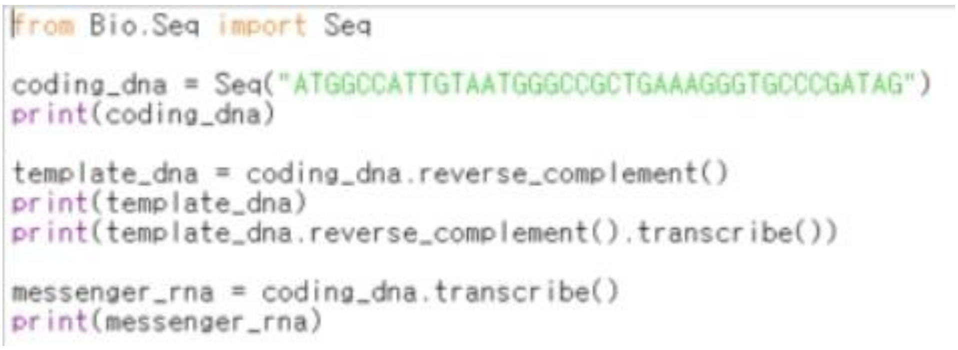


```
GATCGATGGGCCTATATAGGATCGAAAATCGC
CTAGCTACCCGGATATATCCTAGCTTTTAGCG
GCGATTTTCGATCCTATATAGGCCCATCGATC
CGCTAAAAGCTAGGATATATCCGGGTAGCTAG
>>> 
```

Complement(): g->c, a->t

Reverse_comoplement(): complement-> reverse

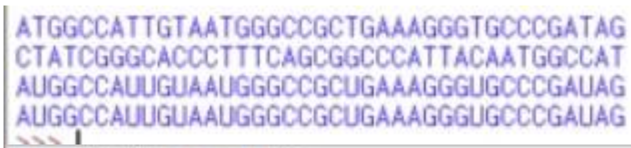
[::-1]: reverse print



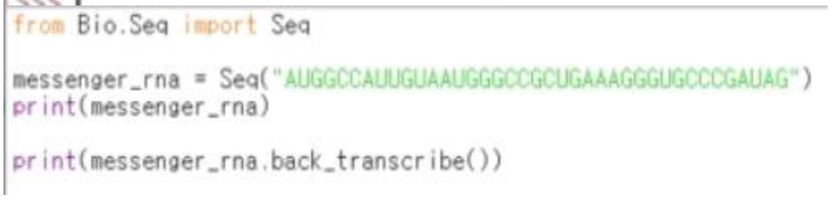
```python
from Bio.Seq import Seq

coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
print(coding_dna)

template_dna = coding_dna.reverse_complement()
print(template_dna)
print(template_dna.reverse_complement().transcribe())

messenger_rna = coding_dna.transcribe()
print(messenger_rna)
```

CTATCGGGCACCCT

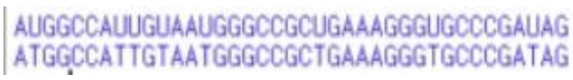
Transcribe: T>U/ back_transcribe: U->T



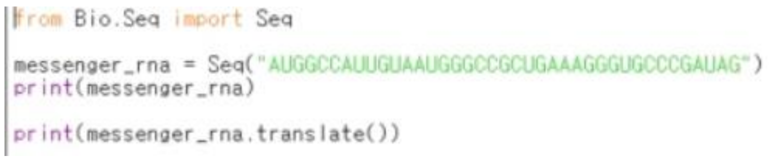
```
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
CTATCGGGCACCCTTTCAGCGGCCCATTACAATGGCCAT
AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG
AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG
>>> 
```



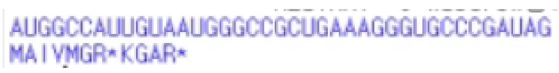
```python
from Bio.Seq import Seq

messenger_rna = Seq("AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG")
print(messenger_rna)

print(messenger_rna.back_transcribe())
```



```
AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
```



```
from Bio.Seq import Seq

messenger_rna = Seq("AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG")
print(messenger_rna)

print(messenger_rna.translate())
```



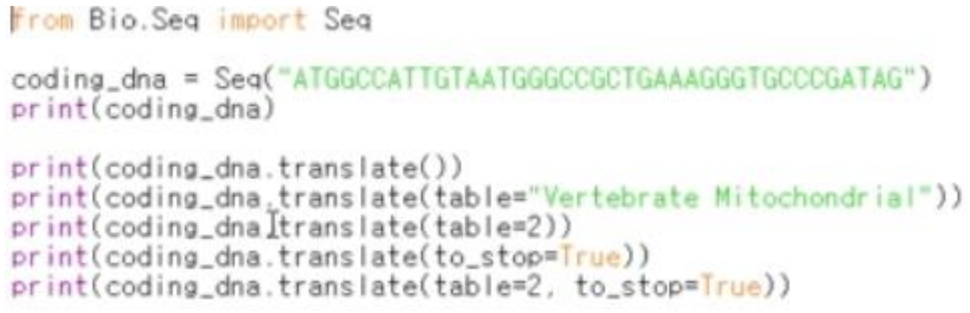
```
AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG
MAIVMGR*KGAR*
```

얘는 rna. 뒤의 dna와 단백질 서열 동일.

Translate(): 단백질로 변환.

AUG-> M, GCC->A, AUU-> I,...

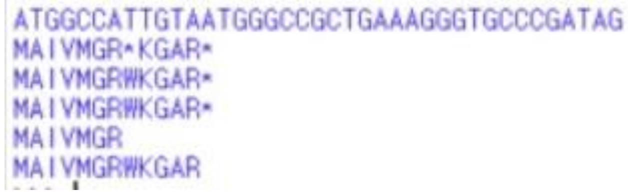
*: 종결 코돈, 시작 코돈..



```
from Bio.Seq import Seq

coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
print(coding_dna)

print(coding_dna.translate())
print(coding_dna.translate(table="Vertebrate Mitochondrial"))
print(coding_dna.translate(table=2))
print(coding_dna.translate(to_stop=True))
print(coding_dna.translate(table=2, to_stop=True))
```

특정 table사용

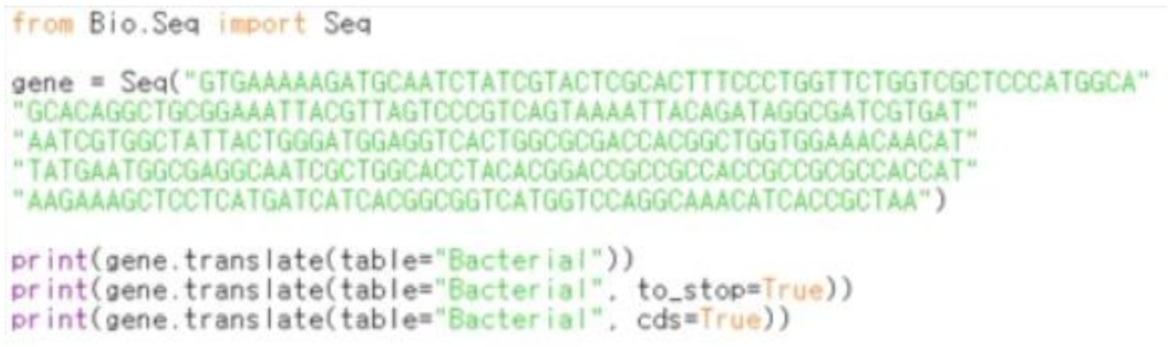
T0_stop=True: 종결 코돈에서 끝내라.



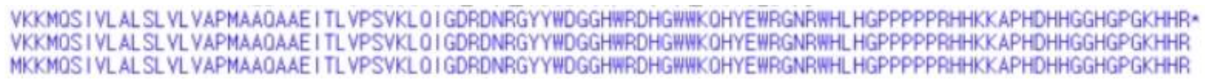
```
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
MAIVMGR*KGAR*
MAIVMGRWKGAR*
MAIVMGRWKGAR*
MAIVMGR
MAIVMGRWKGAR
```

Table 2와 vertebratemitodhondrial 은 같은가봄...

Table 2에서는 *이 무조건 종결코돈은 아닌가봄. ₩로 해석되고 넘어감.



```
from Bio.Seq import Seq

gene = Seq("GTGAAAAAGATGCAATCTATCGTACTCGCACTTTCCCTGGTTCTGGTCGCTCCCATGGCA"
"GCACAGGCTGCGGAAATTACGTTAGTCCCGTCAGTAAAATTACAGATAGGCGATCGTGAT"
"AATCGTGGCTATTACTGGGATGGAGGTCACTGGCGCGACCACGGCTGGTGGAAACAACAT"
"TATGAATGGCGAGGCAATCGCTGGCACCTACACGGACCGCCGCCACCGCCGCGCCACCAT"
"AAGAAAGCTCCTCATGATCATCACGGCGGTCATGGTCCAGGCAAACATCACCGCTAA")

print(gene.translate(table="Bacterial"))
print(gene.translate(table="Bacterial", to_stop=True))
print(gene.translate(table="Bacterial", cds=True))
```



```
VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR*
VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR
MKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR
```

cds: 다음주에...