Biopython ders 1

pip install biopython

```
pip install biopython
```

```
Requirement already satisfied: biopython in c:\users\acer\anaconda3\lib\site-packages (1.85)Note: you may need to restart the kernel to use updated packa
ges.

Requirement already satisfied: numpy in c:\users\acer\anaconda3\lib\site-packages (from biopython) (1.26.4)
```

Hansı versiya olmasına baxırıq

```
import Bio
print (Bio.__version__)
```

```
1.85
```

```python
from Bio.Seq import Seq
my_Seq=Seq("AGTACACTGGT")
print(my_Seq)
```

```
AGTACACTGGT
```

Bu kodda **Bio.Seq** və **Seq Biopython** kitabxanasının bir hissəsidir.

1.Bio.Seq – Biopython kitabxanasının **Seq** modulunu çağırmaq üçün istifadə olunur. Bu modul, nukleotid və ya amin turşusu ardıcıllıqlarını emal etməyə imkan verir.

2.Seq – Biopython-un içindəki **Seq** sinfidir. Bu sinif, DNT, RNT və ya protein ardıcıllıqlarını təmsil edir və onlar üzərində müxtəlif əməliyyatlar aparmağa imkan verir (məsələn, tamamlayıcı ardıcıllıq tapmaq, tərsinə çevirmək və s.).

3.my_Seq = Seq("AGTACACTGGT") – Burada **my_Seq** adlı dəyişən yaradılır və ona **"AGTACACTGGT"** nukleotid ardıcıllığı təyin edilir. Bu ardıcıllıq **Seq** obyektinə çevrilir, yəni o, artıq Biopython-un funksiyaları ilə işləyə bilən xüsusi bir obyekt olur.

4.print(my_Seq) – Bu isə **my_Seq** dəyişəninin tərkibini ekrana çap edir, yəni **AGTACACTGGT** nəticəsini alırsan.

Yəni, **Seq** adi bir mətn kimi görünən DNT/RNT/protein ardıcıllığını Biopython funksiyaları ilə işləmək üçün xüsusi bir obyektə çevirir. Bu obyektlə DNT-nin tamamlayıcı zəncirini tapmaq, transkripsiya etmək, tərsinə çevirmək kimi müxtəlif əməliyyatlar aparmaq mümkündür.

```
my_Seq.complement()
```

```
Seq('TCATGTGACCA')
```

Kodda **my_Seq.complement()** funksiyası **Biopython** kitabxanasının **Seq** obyektinin bir metodudur və DNT ardıcıllığının **tamamlayıcı (complementary) zəncirini** qaytarır.

Necə işləyir?

DNT-də hər nukleotid öz qarşısındakı tamamlayıcı baza ilə cütləşir:

- **A (Adenin) → T (Timin)**
- **T (Timin) → A (Adenin)**
- **C (Sitozin) → G (Guanin)**
- **G (Guanin) → C (Sitozin)**

Kodun iş prinsipi belədir:

Əgər əvvəlcə **my_Seq** belə təyin olunubsa:

from Bio.Seq import Seq my_Seq = Seq("AGTACACTGGT") print(my_Seq.complement())

Burada **my_Seq.complement()** funksiyası çağırıldıqda, hər bir nukleotid uyğun tamamlayıcısı ilə əvəz olunur:

Nəticə:

Ekrana çap olunan tamamlayıcı ardıcıllıq:

Seq('TCATGTGACCA')

Nəticə olaraq, my_Seq.complement() metodu DNT-nin ikinci zəncirini (complementary strand) qaytarır. Bu metod bioinformatikada, xüsusilə genetik analizlərdə və PCR dizaynında tez-tez istifadə olunur.

Əsas zəncir (my_Seq)	A	G	T	A	C	A	C	T	G	G	T
Tamamlayıcı zəncir	T	C	A	T	G	T	G	A	C	C	A

```
my_Seq.reverse_complement()
```

```
Seq('ACCAGTGTACT')
```

my_Seq.reverse_complement() Metodu Nə Edir?

my_Seq.reverse_complement() **Biopython**-un **Seq** obyektinin bir metodudur. Bu metod DNT ardıcıllığının **tamamlayıcı (complementary) və tərsinə çevrilmiş (reverse) zəncirini** qaytarır.

Fərq nədir?

- **complement()** → DNT ardıcıllığının **tamamlayıcı zəncirini** qaytarır.
- **reverse_complement()** → Əvvəlcə **tamamlayıcı zənciri** tapır, sonra onu **tərsinə çevirir**.

Kodun İşləmə Prosesi

1.Əsas ardıcıllıq: "AGTACACTGGT"

2.complement() tətbiq olunur → "TCATGTGACCA"

3.Tərsinə çevrilir (reverse) → "ACCAGTGTACT"

4.Son nəticə çap olunur:

Seq('ACCAGTGTACT')

[241]:
```python
from Bio import SeqIO
file_path = r"C:\Users\Acer\Downloads\ls_orchid.fasta"  # Fayl adını düzgün yaz
for seq_record in SeqIO.parse(file_path, "fasta"):
    print("ID:", seq_record.id)
    print("Sequence Length:", len(seq_record))
    print("Sequence:", repr(seq_record.seq))  # Nukleotid ardıcıllığını çap edir
```

```
ID: gi|2765658|emb|Z78533.1|CIZ78533
Sequence Length: 740
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC')
ID: gi|2765657|emb|Z78532.1|CCZ78532
Sequence Length: 753
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAG...GGC')
ID: gi|2765656|emb|Z78531.1|CFZ78531
Sequence Length: 748
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...TAA')
ID: gi|2765655|emb|Z78530.1|CMZ78530
Sequence Length: 744
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAAACAACAT...CAT')
ID: gi|2765654|emb|Z78529.1|CLZ78529
Sequence Length: 733
Sequence: Seq('ACGGCGAGCTGCCGAAGGACATTGTTGAGACAGCAGAATATACGATTGAGTGAA...AAA')
ID: gi|2765652|emb|Z78527.1|CYZ78527
Sequence Length: 718
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...CCC')
```

Bu **Python** kodu **Biopython** kitabxanasının **SeqIO** modulundan istifadə edərək **FASTA formatlı** bir bioinformatik sekvensiya faylını oxuyur və aşağıdakı məlumatları ekrana çıxarır:

1.Sekvensiyanın (DNT və ya zülal ardıcıllığının) ID nömrəsi

2.Sekvensiyanın uzunluğu (nukleotid və ya amin turşusu sayı)

3.Sekvensiyanın özünü

1. **SeqIO** → Biopython kitabxanasının **sekvensiya fayllarını** oxumaq və yazmaq üçün istifadə olunan moduldur.

2. file_path = r"C:\Users\Acer\Downloads\ls_orchid.fasta"

- **file_path** → Sekvensiya məlumatlarını ehtiva edən .fasta formatlı faylın tam yolunu göstərir.
- **r" " (raw string)** → \ simvollarının xüsusi simvol kimi qəbul edilməməsi üçün istifadə olunur.

3. Sekvensiya Faylını Oxuyur

for seq_record in SeqIO.parse(file_path, "fasta"):

- **SeqIO.parse(file_path, "fasta")** → Verilən faylı **FASTA formatında** oxuyur.
- **for seq_record in ...** → Fayldakı **hər bir sekvensiyanı (record)** tək-tək götürüb işləyir.

4. Sekvensiya Məlumatlarını Çap Etmək

print("ID:", seq_record.id)
- **seq_record.id** → Sekvensiyanın **FASTA başlıq hissəsindəki (header) identifikatorunu (ID)** çıxarır.

print**("Sequence Length:", len(seq_record))**
len(seq_record) → Sekvensiyanın uzunluğunu (nukleotid və ya amin turşusu sayı) qaytarır.

print("Sequence:", repr(seq_record.seq))
seq_record.seq → Sekvensiya ardıcıllığını ekrana çap edir.
repr() → Sekvensiyanı nöqtələr ("...") əvəzinə tam şəkildə göstərmək üçün istifadə olunur.

Kodun İşləmə Nəticəsi (Örnək Çıxış)
Tutaq ki, ls_orchid.fasta faylında bir sekvensiya belədir:

>gi|2765658|emb|AJ001588.1| Orchidaceae rbcL gene ATGTCACGCTTACCGTGGG
Kodun işləməsi nəticəsində ekrana çıxacaq:

ID: gi|2765658|emb|AJ001588.1| Sequence Length: 19 Sequence: 'ATGTCACGCTTACCGTGGG'

Kodun Bioinformatik İstifadəsi
✅ **FASTA formatlı genomik və ya protein sekvensiyalarını oxumaq**
✅ **Genetik analizlər üçün sekvensiya məlumatlarını əldə etmək**
✅ **DNT və ya protein sekvensiyalarının uzunluğunu yoxlamaq**
✅ **Sekvensiya emalı və analiz proseslərinin avtomatlaşdırılması**
Qeyd:
•**FASTA** formatı bioinformatikada **DNT, RNT və ya protein sekvensiyalarını** saxlamaq üçün geniş istifadə olunur.
•SeqIO.parse() funksiyası yalnız **oxumaq üçün** istifadə olunur, əgər sekvensiya dəyişdiriləcəksə, **SeqIO.write()** funksiyası ilə yenidən yazılmalıdır.

[235]:
```python
from Bio import SeqIO

file_path = r"C:\Users\Acer\Downloads\ls_orchid.gbk"  # Düzgün fayl yolu

for seq_record in SeqIO.parse(file_path, "genbank"):  # "gbk" əvəzinə "genbank" yazdıq
    print("ID:", seq_record.id)
    print("Sequence Length:", len(seq_record))
    print("Sequence:", repr(seq_record.seq))
```

```
ID: Z78533.1
Sequence Length: 740
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC')
ID: Z78532.1
Sequence Length: 753
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAG...GGC')
ID: Z78531.1
Sequence Length: 748
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...TAA')
ID: Z78530.1
Sequence Length: 744
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAAACAACAT...CAT')
ID: Z78529.1
Sequence Length: 733
Sequence: Seq('ACGGCGAGCTGCCGAAGGACATTGTTGAGACAGCAGAATATACGATTGAGTGAA...AAA')
ID: Z78527.1
Sequence Length: 718
Sequence: Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...CCC')
```

Kodun İzahı

Bu **Python** kodu **Biopython** kitabxanasının **SeqIO** modulundan istifadə edərək **GenBank formatlı (.gbk)** sekvensiya faylını oxuyur və aşağıdakı məlumatları ekrana çıxarır:

1.Sekvensiyanın (DNT və ya zülal ardıcıllığının) ID nömrəsi
2.Sekvensiyanın uzunluğu (nukleotid və ya amin turşusu sayı)
3.Sekvensiyanın özünü

Kodun işləməsi nəticəsində ekrana çıxacaq:

ID: AJ001588 Sequence Length: 19 Sequence: 'ATGTCACGCTTACCGTGGG'

Kodun Bioinformatik İstifadəsi

✅ **GenBank formatlı genomik və ya protein sekvensiyalarını oxumaq**
✅ **Genetik analizlər üçün sekvensiya məlumatlarını əldə etmək**
✅ **Sekvensiya emalı və annotasiya proseslərini avtomatlaşdırmaq**
✅ **GenBank faylından əlavə metadata (gen adı, xüsusiyyətlər, annotasiyalar və s.) almaq**

GenBank və FASTA Fərqləri

Xüsusiyyət	**GenBank (.gbk)**	**FASTA (.fasta)**
Məlumat tipi	Genetik sekvensiya + annotasiya	Yalnız sekvensiya
Format	Çoxsətirli, strukturlaşdırılmış	Sadə və yığcam
Dəstəklənən məlumatlar	Sekvensiya, gen annotasiyası, xüsusiyyətlər, mənbə	Sekvensiya və başlıq (ID)
Biopython-da oxumaq üçün	"genbank"	"fasta"

Bu kod **FASTA formatı əvəzinə GenBank faylları üçün istifadə olunur** və əlavə annotasiya məlumatlarını emal etməyə imkan yaradır.

[193]:
```
from Bio.Seq import Seq
my_seq = Seq("GATCG")
for index, letter in enumerate(my_seq):
    print(index, letter)
print(len(my_seq))
```

```
0 G
1 A
2 T
3 C
4 G
5
```

Nukleotidlər Üzərində Dövr Qurmaq (Loop)

for index, letter in enumerate(my_seq): print(index, letter)

- **enumerate(my_seq)** → Sekvensiyanın hər bir nukleotidini indekslə birlikdə götürür.
- **for index, letter in ...** → Dövr **hər bir nukleotid üçün indeks (index) və simvol (letter) qaytarır**.
- **print(index, letter)** → Hər bir nukleotidi və onun indeksini ekrana çap edir.
- **İlk beş sətirdə** sekvensiyadakı **hər bir nukleotid və onun indeks** nömrəsi çap olunur.
- **Sonuncu sətirdə** isə sekvensiyanın ümumi uzunluğu (5) ekrana çıxır.

Kodun Bioinformatik İstifadəsi

✅ **DNT/RNT/protein sekvensiyalarında fərdi nukleotidləri və amin turşularını emal etmək**

✅ **Sekvensiyanın uzunluğunu və tərkibini analiz etmək**

✅ **Biyoinformatika alqoritmlərində iterasiya (dövr) əməliyyatları aparmaq**

✅ **Sekvensiya ilə işləyərkən Python string-ləri ilə eyni metodlardan istifadə etmək**

Bu kod **DNT sekvensiyasını analiz etmək üçün sadə bir nümunədir** və daha irəliləmiş bioinformatik əməliyyatlara əsas ola bilər.

```python
#stride
from Bio.Seq import Seq
my_seq=Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")
print (len(my_seq))
print (my_seq.count("G"))
my_seq[4:12]
print(my_seq[4:12])
my_seq[3::2]
print(my_seq[3::2])
my_seq[::-1]
print(my_seq[::-1])
str(my_seq)
```

```
70
20
ACAAGGTT
ACAGTCGAGGACGGAGACTGTAAACGAAAGTGGG
GTGAGCTAGTATATAAGACAACAGAGTTGTTACTAGGAAGGCGTCCAAGTGGATGCCTTTGGAACAATGC
'CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG'
```

len(my_seq)

len(my_seq) → Sekvensiyadakı nukleotidlərin sayını qaytarır.

✅ Nəticə: 60 (çünki sekvensiya 60 nukleotiddən ibarətdir)

my_seq.count("G")

my_seq.count("G") → Sekvensiyada "G" nukleotidinin neçə dəfə təkrarlandığını hesablayır.

✅ Məsələn, əgər sekvensiyada 15 dəfə "G" varsa, nəticə 15 olacaq.

Sekvensiyadan Müəyyən Bir Hissəni Seçmək (Slicing)

my_seq[4:12] print(my_seq[4:12])

•**my_seq[4:12] → 4-cü indeksdən başlayaraq 12-ci indeksə qədər** (12 daxil deyil) olan hissəni seçir.

•**Python indeksləşməsi 0-dan başladığı üçün** 4-cü indeks A hərfi olacaq.

✅ **Məsələn, əgər sekvensiya "CGTAACAAGGTTTCC..." kimidirsə, 4:12 hissəsi "CAAGGTTT" olacaq.**

Müəyyən Addımlarla (Stride) Sekvensiyadan Hissələr Seçmək

my_seq[3::2] print(my_seq[3::2])

•**my_seq[3::2] →**

- •**3-cü indeksdən başlayır** (A hərfi).
- •**Hər dəfə 2 indeks atlayaraq** yeni bir sekvensiya yaradır.

•**Məsələn:**

- •Əsas sekvensiya →
  "CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATC
  GAGTG"
- •3::2 → "AAGTTCGAGGAACCTGGAATGTTAGAAATAAGT"

Sekvensiyanı Tərsinə Çevirmək

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

•**[::-1]** → Sekvensiyanı **tamamilə tərsinə çevirir**.

•**Məsələn:**

- •Əsas sekvensiya →

  "CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG"

- •Tərs sekvensiya →

  "GTGAGCTAGTATATAAGACAACTGAAGTTGATCATGGAGGCGTCCAGGAATGGTCCTTGAACTAAGTACG"

Seq Obyektini Stringə Çevirmək

str(my_seq)

•**str(my_seq)** → Seq obyektini **adi bir Python stringinə** çevirir.

•**Bu əməliyyat string metodları ilə işləməyi asanlaşdırır** (məs., .replace(), .find() və s.).

Kodun İstifadə Sahələri

✓ **Genetik analizlərdə müəyyən sekvensiyaları ayırmaq**

✓ **DNT sekvensiyalarının müxtəlif hissələrini araşdırmaq**

✓ **Sekvensiyaları tərs-tamamlayıcı analizlər üçün çevirmək**

✓ **Genom tədqiqatlarında fərdi nukleotidlərin sayını müəyyənləşdirmək**

[245]:
```
from Bio.Seq import Seq
from Bio.SeqUtils import GC   # SeqUtils düzgün yazıldı
my_seq = Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")
print(GC(my_seq))  # GC faizini çap edir
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[245], line 2
      1 from Bio.Seq import Seq
----> 2 from Bio.SeqUtils import GC   # SeqUtils düzgün yazıldı
      3 my_seq = Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")
      4 print(GC(my_seq))

ImportError: cannot import name 'GC' from 'Bio.SeqUtils' (C:\Users\Acer\anaconda3\Lib\site-packages\Bio\SeqUtils\__init__.py)
```

[249]:
```
from Bio.Seq import Seq
from Bio.SeqUtils import gc_fraction  # Yeni funksiya adı
my_seq = Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")
gc_content = gc_fraction(my_seq) * 100  # GC faizi üçün 100-ə vururuq
print(f"GC Content: {gc_content:.2f}%")
```

```
GC Content: 44.29%
```

Bu kod **Biopython** kitabxanasının **Seq** və **SeqUtils** modullarından istifadə edərək **DNT sekvensiyasının GC faizini hesablayır**.

GC Məzmunu nədir?

DNT sekvensiyasında **Guanin (G) və Sitozin (C)** nukleotidlərinin ümumi faizini ifadə edir.

Formula:

$$GC\% = \left(\frac{G + C}{ümuminukleotidsayı} \right) \times 100$$

GC məzmunu **DNT-nin stabilliyi və təkamülü** haqqında məlumat verir.

- **Seq** → Biopython-un **DNT, RNT və ya protein sekvensiyaları ilə işləmək üçün** obyekt sinfidir.
- **gc_fraction** → DNT sekvensiyasındakı **GC faizini hesablayan funksiya**.

Qeyd:

Əvvəllər GC funksiyası istifadə olunurdu, amma **Biopython 1.78 versiyasından** etibarən **gc_fraction** tövsiyə edilir.

GC Məzmununu Hesablamaq

gc_content = gc_fraction(my_seq) * 100 # GC faizi üçün 100-ə vururuq

•**gc_fraction(my_seq)** → Sekvensiyada **G və C nukleotidlərinin ümumi faizini** qaytarır.

•*** 100** → Onluq kəsrdən **faiz dərəcəsinə** çevirmək üçün 100-ə vururuq.

Məsələn:

Əgər GC fraksiyası 0.45 olarsa,

GC faizi: 0.45 * 100 = 45%

•**F-string (f"")** → Çıxışı formatlı şəkildə yazmaq üçün istifadə olunur.

•**{gc_content:.2f}** → GC faizini **virgüldən sonra iki rəqəm göstərərək** çap edir.

•**Ekrana çıxış (məsələn, GC faizi 45.67% olarsa):**

GC Content: 45.67%

Kodun İstifadə Sahələri

✓ **Genetik analizlərdə** GC məzmununun hesablanması

✓ **DNT-nin stabil olub-olmadığını öyrənmək** (Yüksək GC faizi → Daha stabil DNT)

✓ **Mikroorqanizmlərin təkamülü və adaptasiyası barədə analizlər**

✓ **PCR və digər genetik mühəndislik tətbiqlərində** primer dizaynı

```
•[9]: from Bio.Seq import Seq
      list_of_seqs=[Seq("ACGT"), Seq("AACC"), Seq("TTCC")]
      concatenated=Seq("")
      for s in list_of_seqs:
          concatenated+=s
          pass
          print(list_of_seqs)
          print("Concatenated Sequence:", concatenated)
```

```
[Seq('ACGT'), Seq('AACC'), Seq('TTCC')]
Concatenated Sequence: ACGT
[Seq('ACGT'), Seq('AACC'), Seq('TTCC')]
Concatenated Sequence: ACGTAACC
[Seq('ACGT'), Seq('AACC'), Seq('TTCC')]
Concatenated Sequence: ACGTAACCTTCC
```

DNT Sekvensiyalarının Siyahısını Yaratmaq

list_of_seqs = [Seq("ACGT"), Seq("AACC"), Seq("TTCC")]

•**list_of_seqs** → DNT sekvensiyalarından ibarət siyahıdır.

•**Siyahının elementləri:**

“ACGT"

“AACC"

“TTCC"

Qeyd:

Seq("ACGT") → Bu **str tipi deyil**, **Biopython-un Seq obyektidir**.

Boş Sekvensiya Yaratmaq

concatenated = Seq("")
•**concatenated** → Birləşdiriləcək sekvensiyalar üçün **boş Seq obyekti**.

Əgər biz **birləşdirilmiş sekvensiyanı çap etmək istəyiriksə**, kodun sonuna bunu əlavə etməliyik:

print("Concatenated Sequence:", concatenated)

Kodun Tətbiq Sahələri
✅ **Birdən çox DNT sekvensiyasını birləşdirmək**
✅ **Genetik analizlər üçün primerlər yaratmaq**
✅ **Gen birləşdirmə və mutasiya tədqiqatları**

```
[7]: from Bio.Seq import Seq
     contigs=[Seq("ATG"), Seq("TTCC"), Seq("TTGCA")]
     spacer=Seq("N"*10)
     spacer.join(contigs)
```

```
[7]: Seq('ATGNNNNNNNNNNTTCCNNNNNNNNNNTTGCA')
```

Kodda aşağıdakı addımlar baş verir:

1.contigs adlı siyahıda üç fərqli Seq (ardıcıllıq) obyekti yaradılır:

- Seq("ATG"): Bu bir start kodonu (ATG) ardıcıllığıdır.
- Seq("TTCC"): Bu isə digər bir qısamüddətli ardıcıllıqdır.
- Seq("TTGCA"): Bu da başqa bir ardıcıllıqdır.

2.spacer adlı dəyişkəndə 10 "N" ardıcıllığından ibarət olan bir Seq obyekti yaradılır (Seq("N"*10)), burada "N" hərfi bilinməyən bazaları təmsil edir.

3.spacer.join(contigs) əmri ilə, spacer ardıcıllığı contigs siyahısındakı bütün ardıcıllıqlar arasında birləşdirilir. Bu join əməliyyatı, ardıcıllıqların birləşdirilməsini təmin edir və hər bir ardıcıllıq arasında 10 "N" olan bir fasilə əlavə edir.

Nəticə olaraq, contigs ardıcıllıqları arasına 10 "N" hərfini daxil edən yeni bir seqvens yaranır. Bu üsul genetik məlumatları və ya bioinformatikada istifadə olunan digər məlumatları birləşdirmək üçün faydalıdır.

[113]:
```
from Bio.Seq import Seq
dna_seq=Seq("acgtACGT")
print(dna_seq)
print(dna_seq.upper())
print(dna_seq.lower())
```

```
acgtACGT
ACGTACGT
acgtacgt
```

Bu kod BioPython kitabxanasının Bio.Seq modulundan istifadə edərək bir DNA ardıcıllığının müxtəlif hallarda necə təmsil olunduğunu göstərir. Aşağıda hər bir komanda haqqında izah verilmişdir:

1.dna_seq = Seq("acgtACGT"):

- Bu sətir bir Seq obyekti yaradır və dna_seq adlı dəyişkəndə "acgtACGT" ardıcıllığını saxlayır. Burada Seq obyektinin daxilində kiçik və böyük hərflərlə göstərilmiş bir DNA ardıcıllığı var.

2.print(dna_seq):

- Bu komanda dna_seq dəyişkənini ekrana çap edir. Nəticədə "acgtACGT" ardıcıllığı olduğu kimi göstəriləcək.

3.print(dna_seq.upper()):

- Bu komanda dna_seq ardıcıllığının bütün hərflərini böyük hərflərə çevirir. upper() metodu, hər hansı bir ardıcıllıq üzərindəki kiçik hərfləri böyük hərflərlə əvəz edir. Nəticə olaraq, ACGTACGT çap ediləcək.

4.print(dna_seq.lower()):

- Bu komanda isə dna_seq ardıcıllığının bütün hərflərini kiçik hərflərə çevirir. lower() metodu, hər hansı bir ardıcıllıq üzərindəki böyük hərfləri kiçik hərflərlə əvəz edir. Nəticə olaraq, acgtacgt çap ediləcək.

Bu kodun ümumi məqsədi, bir DNA ardıcıllığının müxtəlif yazım formalarını (kiçik və böyük hərflər) göstərməkdir. upper() və lower() metodları, ardıcıllıqların yazı stilini dəyişdirmək üçün istifadə edilir, xüsusilə bioinformatik analizlərdə belə dəyişikliklər faydalı ola bilər.

[129]:
```
from Bio.Seq import Seq
my_seq=Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")
print(my_seq)
print(my_seq.complement())
print(my_seq.reverse_complement())
```

```
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG
GCATTGTTCCAAAGGCATCCACTTGGACGCCTTCCTAGTAACAACTCTGTTGTCTTATATACTAGCTCAC
CACTCGATCATATATTCTGTTGTCTCAACAATGATCCTTCCGCAGGTTCACCTACGGAAACCTTGTTACG
```

Bu kod BioPython kitabxanasının Bio.Seq modulundan istifadə edərək bir DNA ardıcıllığının tamamlayıcı (complement) və tərsinə tamamlayıcı (reverse complement) formalarını əldə etmək üçün istifadə edilir. Kodda istifadə olunan metodlar və onların məqsədləri aşağıda izah edilmişdir:

1.my_seq = Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG"):

- Bu sətir bir Seq obyekti yaradır və my_seq adlı dəyişkəndə "CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG" ardıcıllığını saxlayır. Bu ardıcıllıq bir DNA nümunəsidir.

2.print(my_seq):

- Bu komanda my_seq ardıcıllığını ekrana çap edir. Nəticədə, DNA ardıcıllığı olduğu kimi ("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG") göstəriləcək.

3.print(my_seq.complement()):

- complement() metodu, ardıcıllığın tamamlayıcı bazlarını qaytarır. DNA-nın tamamlayıcı bazları belədir:
  - A (Adenin) → T (Timin)
  - T (Timin) → A (Adenin)
  - C (Sitozin) → G (Guanin)
  - G (Guanin) → C (Sitozin)

Bu komanda, my_seq ardıcıllığının tamamlayıcı formunu (yəni hər bir bazın tamamlayıcı bazla əvəzlənməsi) ekrana çap edir.

4.print(my_seq.reverse_complement()):

- reverse_complement() metodu isə iki əməliyyatı birləşdirir:
  - İlk olaraq, ardıcıllığın tamamlayıcı bazlarını tapır.
  - Daha sonra bu tamamlayıcı ardıcıllığı tərsinə çevirir.

Nəticədə, ardıcıllığın həm tamamlayıcı, həm də tərsinə çevrilmiş forması əldə edilir və ekrana çap olunur.

Nümunə:

- **my_seq**: CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG
- **complement()**: GCATTTGTTCCAAGGCATCCACTTGGACGCCCTTCCTAGTAACAACACTCTTGTCCTATATACTAGCTCA
- **reverse_complement()**: GCTCAGTATATCCACTTGTTCAGGACAGTCAAGTGAATCCTTCAGCTGACCTTGGACGGAAGTTAACATGCG

Bu metodlar xüsusilə bioinformatikada, DNA ardıcıllıqlarını təhlil edərkən və müqayisə edərkən çox istifadə olunur.

Təşəkkürlər