Tapsiriq 1. Transkripsiya etmək – Verilmiş DNT ardıcıllığını transcribe() metodundan istifadə edərək RNT ardıcıllığına çevirin. Məsələn: Seq("AGTACACTGGT").transcribe()

[3]:
```
from Bio.Seq import Seq
my_seq=Seq("AGTACACTGGT")
rna_seq = my_seq.transcribe()
my_seq.transcribe()
print(rna_seq)
```

```
AGUACACUGGU
```

- **Seq("AGTACACTGGT")** – Bu, DNT ardıcıllığını yaradır.
- **transcribe()** – Bu metod **T** bazasını **U** ilə əvəz edərək RNT ardıcıllığı yaradır.
- **print(rna_seq)** – Transkripsiya olunmuş RNT ardıcıllığını ekrana çap edir.

Tapsiriq 2. Translyasiya etmək – Verilmiş RNT ardıcıllığını translate() metodundan istifadə edərək müvafiq zülal (protein) sekvensiyasına çevirin. Məsələn: Seq("AUGGCCAUUGUAAUGGGUAG").translate()

[13]:
```
from Bio.Seq import Seq

# RNT ardıcıllığını yaranir
my_seq = Seq("AUGGCCAUUGUAAUGGGUAG")

# Translyasiya edirik (RNT → Protein)
protein_seq = my_seq.translate()

# Nəticəni çap edirik
print(protein_seq)
```

```
MAIVMG
```

Açıqlama:

1.Seq("AUGGCCAUUGUAAUGGGUAG") – RNT ardıcıllığıdır.

2.translate() – Bu metod 3-lü kodonları amin turşularına çevirir.

3.print(protein_seq) – Zülal sekvensiyasını çap edir.

- **M** → Metionin (AUG start kodonu)
- **A** → Alanin (GCC)
- **P** → Prolin (AUU)
- **I** → İzolösin (GUA)
- **Stop kodonu** → (* işarəsi ilə göstərilir)

Tapsiriq 3. DNT və RNT fərqi – replace() metodundan istifadə edərək DNT ardıcıllığındakı T hərflərini U ilə əvəz edin və nəticəni çap edin.

[21]:
```
from Bio.Seq import Seq

# DNT ardıcıllığını yaradirik
my_seq = Seq("ATGGCCATTGTAATGGGTAG")

# "T" hərflərini "U" ilə əvəz edərək RNT ardıcıllığını yaradirik
rna_seq = str(my_seq).replace("T", "U")

print(rna_seq)
```

```
AUGGCCAUUGUAAUGGGUAG
```

Biopython-un Seq obyektində replace() metodu yoxdur, çünki **Seq obyekti Python str kimi birbaşa dəyişdirilə bilməz**.

Bunun əvəzinə, DNT ardıcıllığındakı **"T"** hərflərini **"U"** ilə əvəz etmək üçün **Python str.replace() metodundan istifadə edə bilərsiniz**. Amma bunu str formatına çevirərək etməlisiniz.

Açıqlama:
1.Seq("ATGGCCATTGTAATGGGTAG") – DNT ardıcıllığını yaradır.
2.str(my_seq).replace("T", "U") – str() metodu ilə **DNT ardıcıllığını string formatına** çevirir və sonra **"T" hərflərini "U" ilə əvəz edir**.
3.print(rna_seq) – Əldə edilən RNT ardıcıllığını çap edir.

Tapsiriq 4. DNT sekvensiyasını amin turşularına çevirmək – Verilmiş DNT sekvensiyasını əvvəlcə RNT-yə, sonra isə protein sekvensiyasına çevirin.

[25]:
```
from Bio.Seq import Seq

# DNT ardıcıllığını yaradiriq
my_seq = Seq("ATGGCCATTGTAATGGGTAG")

# DNT-ni RNT-yə çeviririk (T → U)
rna_seq = my_seq.transcribe()

# RNT-ni protein sekvensiyasına çeviririk
protein_seq = rna_seq.translate()

print("RNT sekvensiyası:", rna_seq)
print("Protein sekvensiyası:", protein_seq)
```

```
RNT sekvensiyası: AUGGCCAUUGUAAUGGGUAG
Protein sekvensiyası: MAIVMG
```

- **"AUG"** → Metionin (**M**, start kodonu)
- **"GCC"** → Alanin (**A**)
- **"AUU"** → İzolösin (**I**)
- **"UGU"** → Cistein (**C**)
- **"AAU"** → Asparagin (**N**)
- **"GGG"** → Glisin (**G**)
- **"UAG"** → STOP kodonu (**"*"** ilə göstərilir)

- **Seq("ATGGCCATTGTAATGGGTAG")** – DNT ardıcıllığını Seq obyektində təyin edir.
- **transcribe()** – DNT-ni RNT-yə çevirir (T bazalarını U ilə əvəz edir).
- **translate()** – RNT-ni amin turşularına çevirir.
- **print()** – Həm RNT, həm də protein sekvensiyalarını çap edir.

Tapsiriq 5. FASTA Faylından Məlumat Çıxarmaq – SeqIO.parse() metodundan istifadə edərək .fasta faylında olan bütün sekvensiyaların ID və uzunluqlarını çap edin.

[33]:
```
from Bio import SeqIO
file_path = r"C:\Users\Acer\Downloads\ls_orchid.fasta"
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))
```

```
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')
```

[35]:
```
from Bio import SeqIO

# FASTA faylının yolunu daxil et
file_path = r"C:\Users\Acer\Downloads\ls_orchid.fasta"

# FASTA faylını oxu və ID, uzunluq məlumatlarını çap et
for seq_record in SeqIO.parse(file_path, "fasta"):
    print("ID:", seq_record.id)
    print("Sequence Length:", len(seq_record.seq))  # Burada `.seq` olmalıdır
    print("Sequence:", repr(seq_record.seq))  # Sekvensiyanı tam göstərir
    print("-" * 50)  # Hər bir sekvensiyanı ayırmaq üçün xətt
```

```
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
```

Tapsiriq 6. GenBank Faylından Annotasiyalar – .gbk faylını oxuyaraq, genlərin adlarını və onların start-stop mövqelərini çıxarın.

[37]:
```python
from Bio import SeqIO

# GenBank faylının yolunu daxil edirik
file_path = r"C:\Users\Acer\Downloads\ls_orchid.gbk"

# Faylı oxuyuruq və annotasiya məlumatlarını çıxarırıq
for seq_record in SeqIO.parse(file_path, "genbank"):
    print("ID:", seq_record.id)
    print("Sequence Length:", len(seq_record.seq))

    # Hər bir annotasiya (feature) üzərindən keçirik
    for feature in seq_record.features:
        if feature.type == "gene":  # Yalnız "gene" anotasiya tipini götür
            gene_name = feature.qualifiers.get("gene", ["Unknown"])[0]  # Gen adını al
            start = feature.location.start
            end = feature.location.end
            print(f"Gen: {gene_name}, Start: {start}, End: {end}")

    print("-" * 50)  # Məlumatları ayırmaq üçün xətt
```

```
ID: Z78533.1
Sequence Length: 740
Gen: 5.8S rRNA, Start: 380, End: 550
--------------------------------------------------
ID: Z78532.1
Sequence Length: 753
Gen: 5.8S rRNA, Start: 380, End: 550
--------------------------------------------------
ID: Z78531.1
Sequence Length: 748
Gen: 5.8S rRNA, Start: 380, End: 550
--------------------------------------------------
```

Kodun işləmə prinsipi:

1.SeqIO.parse(file_path, "genbank") – GenBank faylını oxuyur.

2.seq_record.features – Sekvensiyaya aid bütün annotasiyaları saxlayan siyahıdır.

3.feature.type == "gene" – Yalnız **"gene"** anotasiya tipini seçir (digərləri, məsələn, CDS və ya tRNA nəzərə alınmır).

4.feature.qualifiers.get("gene", ["Unknown"])[0] – Genin adını alır (əgər gen adı yoxdursa, "Unknown" kimi göstərir).

5.feature.location.start və feature.location.end – Genin **başlanğıc və bitmə koordinatlarını** çıxarır.

6.Çap edilən məlumatlar hər sekvensiya üçün ayrıca göstərilir.

Tapsiriq 7. FASTA Faylında Ən Uzun Sekvensiyanı Tapın – .fasta faylında olan sekvensiyalardan ən uzununu seçin və çap edin.

[39]:
```python
from Bio import SeqIO

# FASTA faylının yolunu daxil edin
file_path = r"C:\Users\Acer\Downloads\ls_orchid.fasta"

# Ən uzun sekvensiyanı tapmaq üçün dəyişənlər
longest_seq = None
max_length = 0

# Faylı oxuyuruq və ən uzun sekvensiyanı tapırıq
for seq_record in SeqIO.parse(file_path, "fasta"):
    seq_length = len(seq_record.seq)
    if seq_length > max_length:
        max_length = seq_length
        longest_seq = seq_record

# Ən uzun sekvensiyanı çap edirik
if longest_seq:
    print("Ən uzun sekvensiyanın ID-si:", longest_seq.id)
    print("Uzunluğu:", max_length)
    print("Sekvensiya:", longest_seq.seq)
```

```
Ən uzun sekvensiyanın ID-si: gi|2765620|emb|Z78495.1|PEZ78495
Uzunluğu: 789
Sekvensiya: CGTAACAAGGTTTCCGTAGGTGAACCTCCGGAAGGATCATTGTTGAGATCACATAATAATTGATCGAGATAATCCAGAGGATCGGTTTACTTTGGTCACCCATGGGCGCTTGCTATTGCGGTGACCTAGAGTTGCCATGGA
GAGCCTCCTTGGGAGCTTTCTTGCCGGCGATCTAACCCTTGTCCGGCGCGGTTTTGCGCCAAGTCATATGACACATAATTGGTGAAGGGCATAGCCCTTCGTGTATTCAAGGAGGGGGGGCGGCATGTGGCCTTGACACTGCACTCGCTCTCC
CCCTCTCCAAAGTATTTTTCTGAACAACTCTCAGCAACGGATATCTCAGCTCTTGCATCGGATGGAGGAACGCAGCGAAATCCGATAAGTGGTGTGAATTGCAGAATCCCGTGAACCATCGAGTCTTTTGAACGCAAGTTGCGCCCGAGGCCA
TGAGGCCAAGGGCACGCCTGCCTGGGCATTGCGAGTCATCTCTCTCCCTCAATGAGGCTGTCCATGCATACTGTTCAGCCGGTGCGGATGTGAGTTTGGCCCCTTGTTCTTTGGTGCTGGGGGTCTAAGAGCAGCAGGGGCTTTTGATGGTCC
TAAATTCGGCAAGAGGTGGACGAATCAGGCTACAACAACACTGTTGTTGTGCGAATGCTCCAGGTTGTCGTATTAGATGGGCCGGCATAATCCAGAGACCCTTGTGAACCCCATTGGAGGCCCATCAACCCATGATCAGTTGATGGCCATTCG
GTTGCGACCCCAGGTCAGTGAGCAACCCGCTGAGTG
```

Kodun işləmə prinsipi:

1.SeqIO.parse(file_path, "fasta") – FASTA faylını oxuyur.

2.max_length və longest_seq dəyişənləri – Ən uzun sekvensiyanı və uzunluğunu saxlamaq üçün istifadə edilir.

3.Döngü ilə bütün sekvensiyalara baxırıq:

- Hər sekvensiyanın uzunluğunu (len(seq_record.seq)) yoxlayırıq.
- Əgər cari sekvensiya **indiyə qədər tapılan ən uzundursa**, onu yadda saxlayırıq.

4.Ən uzun sekvensiyanı çap edirik.

Tapsiriq 8. GC məzmunu hesablamaq – gc_fraction() metodundan istifadə edərək DNT sekvensiyasının GC tərkibini faizlə hesablayın.

[41]:
```
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%
```

Tapsiriq 9. DNT sekvensiyasının motivini tapmaq – find() metodundan istifadə edərək verilmiş DNT sekvensiyasında müəyyən bir motivin (məsələn, "ATG") olub-olmadığını yoxlayın.

[45]:
```
from Bio.Seq import Seq

# DNT sekvensiyası
my_seq = Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")

# Axtarılacaq motiv
motif = "ATG"

# find() metodu yalnız str obyektlərində işlədiyi üçün Seq-i str-ə çeviririk
position = str(my_seq).find(motif)

# Nəticəni çap edirik
if position != -1:
    print(f"Motiv '{motif}' sekvensiyada {position}-ci indeksdə tapıldı.")
else:
    print(f"Motiv '{motif}' sekvensiyada tapılmadı.")
```

```
Motiv 'ATG' sekvensiyada 59-ci indeksdə tapıldı.
```

Kodun işləmə prinsipi:

1.str(my_seq).find(motif)

- Seq obyektini str formatına çeviririk (str(my_seq)).
- find(motif) funksiyası motivin indeksini qaytarır.
- Əgər motiv yoxdursa, -1 qaytarır.

2.Şərt yoxlanılır (if position != -1)

- Əgər indeks -1 deyilsə, motivin **ilk tapıldığı yeri** çap edirik.
- Əks halda "tapılmadı" mesajını göstəririk.

Tapsiriq 10. DNT sekvensiyalarını müqayisə etmək – İki fərqli DNT sekvensiyasını müqayisə edərək oxşar və fərqli hissələri çap edin.

İki fərqli DNT sekvensiyasını müqayisə edərkən **oxşar** və **fərqli** hissələri tapmaq üçün align() və ya sadəcə == operatorunu istifadə edə bilərsiniz. Bunun üçün **Seq obyektlərindən** və ya pairwise2 modulundan istifadə etmək olar.

[51]:
```
from Bio.Seq import Seq

# İki fərqli DNT sekvensiyası
seq1 = Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")
seq2 = Seq("ACGAAGTAGGTTTCCGTAAGTGAACCTGCGGAAGGATATCGGTTGAGACAACAGAATATATGATCGAGTG")

# Müqayisə edirik
matching_positions = []
non_matching_positions = []

# Hər iki sekvensiyanın uzunluğunu alırıq (bərabər olmalıdır)
min_len = min(len(seq1), len(seq2))

# İki sekvensiyanı element üzrə müqayisə edirik
for i in range(min_len):
    if seq1[i] == seq2[i]:
        matching_positions.append(i)
    else:
        non_matching_positions.append(i)

# Oxşar və fərqli hissələri çap edirik
print(f"Oxşar hissələr: {matching_positions}")
print(f"Fərqli hissələr: {non_matching_positions}")
```

```
Oxşar hissələr: [3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 41, 42, 43, 44, 4
5, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]
Fərqli hissələr: [0, 1, 2, 5, 6, 18, 37, 38, 39, 40]
```

[53]:
```python
from Bio import pairwise2
from Bio.Seq import Seq

# İki fərqli DNT sekvensiyası
seq1 = Seq("CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAGAATATATGATCGAGTG")
seq2 = Seq("ACGAAGTAGGTTTCCGTAAGTGAACCTGCGGAAGGATATCGGTTGAGACAACAGAATATATGATCGAGTG")

# Align etmək
alignments = pairwise2.align.globalxx(seq1, seq2)

# Nəticəni çap edirik
for alignment in alignments:
    print("Aligned sequences:")
    print("Seq1: ", alignment[0])
    print("Seq2: ", alignment[1])
    print("Score: ", alignment[2])
    print()
```

```
Aligned sequences:
Seq1:  -CGTAACA--AGGTTTCCGTAG-GTGAACCTGCGGAAGGATCATT-G-TTGAGACAACAGAATATATGATCGAGTG
Seq2:  ACG--A-AGTAGGTTTCCGTA-AGTGAACCTGCGGAAGGAT-A-TCGGTTGAGACAACAGAATATATGATCGAGTG
Score:  64.0

Aligned sequences:
Seq1:  -CGTAACA--AGGTTTCCGTAG-GTGAACCTGCGGAAGGATCATT-G-TTGAGACAACAGAATATATGATCGAGTG
Seq2:  ACG-A--AGTAGGTTTCCGTA-AGTGAACCTGCGGAAGGAT-A-TCGGTTGAGACAACAGAATATATGATCGAGTG
Score:  64.0
```

Kodun işləmə prinsipi:

1.Sadə müqayisə:

- DNT sekvensiyalarını mövqelər üzrə müqayisə edirik.
- Oxşar və fərqli olan indeksləri **siyahıya** əlavə edirik.
- Nəticədə oxşar və fərqli hissələri çıxarırıq.

2.Pairwise align:

- pairwise2.align.globalxx(seq1, seq2) funksiyası, **daha dəqiq** bir şəkildə iki sekvensiyanı **align** edir və ən yaxşı uyğunluğu tapır.
- Çıxışda **aligned** (uzlaşdırılmış) sekvensiyalar və **nəticə dəyəri** göstərilir.

Tapsiriq 11. Verilmiş DNT sekvensiyasında potensial kodlaşdırıcı bölgələri (ATG start kodonu ilə başlayıb TAA, TAG və ya TGA stop kodonları ilə bitən bölgələri) tapın.

[1]:
```python
from Bio.Seq import Seq

# DNA sekansı
my_Seq = Seq("ATGGCATCGGTGACCAGTATGCTACGATTAAATGCGTAAGTTGCCATTATGGATCGATTCAAGTGA")

# Kodonlar
start_codon = "ATG"
stop_codons = ["TAA", "TAG", "TGA"]

def find_coding_regions(my_Seq):
    coding_regions = []

    # Start kodonlarını bul
    start_indices = [i for i in range(len(my_Seq) - 2) if my_Seq[i:i+3] == start_codon]

    for start_index in start_indices:
        possible_stops = []

        # Stop kodonlarını bul (okuma çerçevesine uygun olanları al)
        for stop_codon in stop_codons:
            stop_index = my_Seq.find(stop_codon, start_index + 3)
            while stop_index != -1:
                if (stop_index - start_index) % 3 == 0:  # 3'ün katı mı?
                    possible_stops.append(stop_index)
                    break  # En erken stop kodonunu al
                stop_index = my_Seq.find(stop_codon, stop_index + 3)  # Bir sonraki stop kodonuna bak

        # Eğer uygun stop kodonu bulunduysa, kodlaşdırıcı bölgeyi ekle
        if possible_stops:
            stop_index = min(possible_stops)  # En erken uygun stop kodonunu seç
            coding_regions.append(my_Seq[start_index:stop_index + 3])

    return coding_regions

# Kodlaşdırıcı bölgeleri bul
coding_regions = find_coding_regions(my_Seq)

# Sonuçları yazdır
print("Tapılan potensial kodlaşdırıcı bölgələr:")
for region in coding_regions:
    print(region)
```

```
Tapılan potensial kodlaşdırıcı bölgələr:
ATGGCATCGGTGACCAGTATGCTACGATTAAATGCGTAA
ATGCTACGATTAAATGCGTAA
ATGGATCGATTCAAGTGA
```

✅ **Sadece 3'ün katı olan bölgeleri alıyor** (Gerçek gen yapısını daha doğru simüle eder).
✅ **Her ATG için en yakın uygun TAA, TAG veya TGA'yı buluyor** (Daha doğru analiz yapar).
✅ **Okuma çerçevesini ihlal etmeyen stop kodonlarını değerlendiriyor**.
✅ **En erken doğru stop kodonunu alıyor**, gereksiz tarama yapmıyor.

Tapsiriq 12. DNT sekvensiyasında tandem təkrarlanan motivləri (yəni ardıcıl olaraq bir neçə dəfə təkrarlanan qısa nukleotid ardıcıllıqları) tapın və onların təkrarlanma sayını müəyyən edin.

- **Tandem təkrarlanan motivlər:** DNT ardıcıllığında ardıcıl olaraq bir neçə dəfə təkrarlanan motivlər.
- **Nümunə motiv:** Məsələn, "AT" və ya daha uzun ardıcıllıqlar.

- **count() metodu:** Bu metod, müəyyən bir motivin (burada "AT") sekvensiyada neçə dəfə təkrarladığını tapır.
- **Sadə DNT sekvensiyası:** Məsələn, ATATATCGCGCGCAGCTGATATATAGCGCGCGAT üzərində "AT" motivinin neçə dəfə təkrarlanmasını tapırıq.

[63]:
```
from Bio.Seq import Seq

# DNT sekvensiyası
my_seq = Seq("ATATATCGCGCGCAGCTGATATATAGCGCGCGAT")

# Tandem motivini təyin edirik
motif = "AT"

# Sekvensiyada motivin təkrarlanma sayını tapırıq
count = my_seq.count(motif)

# Nəticəni çap edirik
print(f"Motif '{motif}' appears {count} times in the sequence.")
```

```
Motif 'AT' appears 7 times in the sequence.
```

Tandem Təkrarlanan Motivi Tapmaq:
Bu sadə yanaşma yalnız müəyyən bir motivin neçə dəfə təkrarlanmasını tapır. Əgər daha mürəkkəb tandem təkrarlanan ardıcıllıqları (məsələn, "ATATAT") tapmaq istəyirsinizsə, onda daha uzun motivlər üçün count() metodunu istifadə edə bilərsiniz. Ancaq bu yanaşma üçün motivin təkrarlanma sayını və mövqelərini tapmaq üçün idealdır.