Biopython ders 2

* **Nəyi Bildirir?**

ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG DNA ardıcıllığı **zülal kodlayan bir genin kodlama hissəsidir**.

Bu DNA ardıcıllığı **transkripsiya və translasiya** prosesi nəticəsində **aminoturşu zəncirinə** çevrilir.

Belə bir nəticə alınır:

ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG (DNA) ↓ Transkripsiya (mRNA sintezi)

AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG (mRNA) ↓ Translasiya (Amin turşu ardıcıllığı) MAIVMGR*KGAR*

Buradakı *** işarəsi STOP kodonu**nu (durdurucu kodonu) bildirir.

STOP Kodonları Nədir?

Protein sintezi **kodonlar (üçlü nukleotid qrupları)** vasitəsilə idarə olunur.

Genetik kodda **3 növ STOP kodonu** var:

•**UAA** (**Ochre**)

•**UAG** (**Amber**)

•**UGA** (**Opal**)

Bu kodonlardan biri ribosom tərəfindən oxunduqda, **protein sintezi dayanır**.

STOP Kodonu Necə Yaranır?

Sizin DNA ardıcıllığınıza baxaq:

ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG (DNA)

AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG (mRNA)

Burada **UAG kodonu** STOP kodonudur.

UAG ("Amber Stop Kodonu") adlanır və **protein sintezini dayandırır**.

Xülasə

✓ *** STOP kodonu deməkdir**.

✓ **UAA, UAG, UGA zülal sintezini dayandıran kodonlardır**.

✓ Bu kodonlardan biri gəldikdə, **ribosom zülalı tamamlayır və onu sərbəst buraxır**.

Yəni, *** işarəsi protein sintezinin orada bitdiyini göstər**

Genetik Kod: Standart Kod və Onurğalıların Mitoxondrial Kodu Arasındakı Fərqlər

Genetik kod, **DNT və ya RNT-dəki kodonların (üçlü nukleotidlər)** hansı amin turşularına uyğun gəldiyini müəyyən edən qaydalar toplusudur.

Fərqli orqanizmlər və orqanoidlər (məs., mitoxondrilər) müxtəlif genetik kodlardan istifadə edə bilər.

Biopython-da genetik kod cədvəlləri **transl_table** ilə göstərilir.

Ən çox istifadə olunan iki kod:

- **Standart Genetik Kod (The Standard Code, transl_table=1)**
- **Onurğalıların Mitoxondrial Kodu (The Vertebrate Mitochondrial Code, transl_table=2)**

Əsas Fərqlər

Xüsusiyyət	Standart Kod (Table 1)	Mitoxondrial Kod (Table 2)
Start Kodonu	AUG (M)	AUG, AUA, AUU (M)
Stop Kodonları	UAA, UAG, UGA	UAA, UAG (amma UGA = W)
UGA kodonu	STOP	Tryptophan (W)
AUA kodonu	Isoleucine (I)	Methionine (M)

https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi

1. Standart Genetik Kod (transl_table=1)

Bütün hüceyrə nüvəsində (nüvə genomu) istifadə olunan əsas kod budur.

Bakteriyalar, arxeylər və nüvə genomuna malik olan **heyvan, bitki və göbələklər** tərəfindən istifadə olunur.

📌 **Xüsusiyyətləri:**

✓ **Start kodonu:** AUG (Methionine, M)

✓ **Stop kodonları:** UAA, UAG, UGA

✓ Bütün digər kodonlar **standart amin turşularını kodlayır**.

2. Onurğalıların Mitoxondrial Kodu (transl_table=2)

Mitoxondrilərdə fərqli təkamül yolu olduğuna görə **mitoxondrial DNT (mtDNA) nüvə DNT-dən fərqli genetik koddan istifadə edir.**

📌 **Xüsusiyyətləri:**

✓ **Start kodonları:** AUA, AUG, AUU (hamısı Methionine, M kodlayır)

✓ **Stop kodonları fərqlidir:**

•**UGA stop kodonu deyil**, **Tryptophan (W)** kodlayır!

•Stop kodonlar **yalnız UAA və UAG-dır**.

Biopython ilə İstifadə

Siz Biopython ilə **müxtəlif translasiya cədvəllərindən istifadə edə bilərsiniz**.

Standart Kodla Translasiya (Table 1)

from Bio.Seq import Seq dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG") protein = dna.translate(table=1) print(protein)

📌 **Çıxış:**

markdown

MAIVMGR*KGAR*

Burada * STOP kodonunu göstərir.

Nəticə

✓ **Standart kod** hüceyrə nüvəsində istifadə olunur.

✓ **Mitoxondrial kod** mitoxondrilərdə istifadə edilir və bəzi fərqlər var:

•**UGA STOP deyil, W (Tryptophan) kodlayır.**

•**AUA Methionine (M) kodlayır, nüvə DNT-də isə Isoleucine (I).**

•**Stop kodonlar sadəcə UAA və UAG olur.**

Mitoxondrial Kodla Translasiya (Table 2)

protein_mito = dna.translate(table=2) print(protein_mito)

📌 Fərqlilik:

Əgər UGA kodonu varsa, **STOP əvəzinə W (Tryptophan) olacaq.**

Mitoxondrial Kodla Translasiya Nəticəsi: MAIVMGRWKGAR* Nə Deməkdir?

Mitoxondrial genetik koddan (Vertebrate Mitochondrial Code) istifadə etməklə translyasiya aparmışıq:

print(coding_dna.translate(table="Vertebrate Mitochondrial"))

Output**:**

MAIVMGRWKGAR*

```
#back-transcription method
messenger_rna=Seq("AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG")
print(messenger_rna)
print(messenger_rna.back_transcribe())
print(messenger_rna.translate())
#ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG burdan ede bilirsen...
```

```
AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
MAIVMGR*KGAR*
```

1. Standart Kod ilə Fərq Nədir?

Əvvəlki standart genetik kodda output belə idi:

MAIVMGR*KGAR*

Lakin **mitoxondrial kodla** nəticə:

MAIVMGRWKGAR*

Burada **əsas fərq W (Tryptophan) amin turşusudur.**

2. UGA Kodonu STOP Əvəzinə Tryptophan (W) Kodlayır

Standart kodda UGA **STOP kodonu** olduğu üçün **zülal erkən dayanır.**

Lakin **onurğalıların mitoxondrial kodunda** UGA **STOP deyil**, əvəzinə **Tryptophan (W) kodlayır.**

DNT ardıcıllığı:

ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG (DNT)

AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG (mRNT)

Standart kodla translasiya:

MAIVMGR*KGAR* (UGA STOP-dur)

Mitoxondrial kodla translasiya:

MAIVMGRWKGAR* (UGA → W)

Burada **UGA artıq STOP kodonu deyil**, **o, Tryptophan (W) kodlayır.**

```
#veya
coding_dna=Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
print(coding_dna)
print(coding_dna.translate())
```

```
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
MAIVMGR*KGAR*
```

```
print(coding_dna.translate(table="Vertebrate Mitochondrial"))
#veya
coding_dna.translate(table=2)
```

```
MAIVMGRWKGAR*
Seq('MAIVMGRWKGAR*')
```

3. Nəticə

✓ **Onurğalı mitoxondrial kodu** UGA kodonunu **STOP kimi yox, Tryptophan (W) kimi oxuyur.**

✓ **Bu səbəbdən standart kodla alınan nəticədəki STOP əvəzinə W (Tryptophan) görünür.**

✓ Digər kodonlar eyni qalır.

Ona görə də çıxış MAIVMGRWKGAR* olur.

Stop kodonların silinməsi və əlavə olunması

```
print(coding_dna.translate())
#Seq("MAIVMGR*KGAR*")
print(coding_dna.translate(to_stop=True))
```

```
MAIVMGR*KGAR*
MAIVMGR
```

```
print(coding_dna.translate(table=2))
```

```
MAIVMGRWKGAR*
```

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

```
MAIVMGRWKGAR
```

Bu kod **BioPython** kitabxanasından istifadə edərək **DNT ardıcıllığını (coding_dna) amin turşularına çevirir** (translyasiya edir).

print(coding_dna.translate())

➞ **Standart genetik kodla** DNT ardıcıllığı amin turşularına çevrilir.

Output:

MAIVMGR*KGAR*

Ulduz (*) STOP kodonlarını göstərir.

STOP kodonu, zülal sintezinin dayandığı yerdir.

print(coding_dna.translate(to_stop=True))

➞ to_stop=True istifadə edildikdə, **STOP kodonundan sonra gələn hissə atılır**.

Output :

MAIVMGR

Bu, **zülalın tam dayandığı yerə qədər olan hissəsini göstərir.**

Alternativ Genetik Kod (Table=2)

print(coding_dna.translate(table=2))

➞ **Genetik kod table=2 seçilir (Vertebrate Mitochondrial Code).**

Output :

MAIVMGRWKGAR*

Burada **UGA STOP əvəzinə Tryptophan (W) kodlayır.**

print(coding_dna.translate(table=2, to_stop=True))

➞ Burada **mitoxondrial kodla STOP-a qədər çevirir**.

Output :

MAIVMGRWKGAR

STOP kodonundan sonrakı hissə atılır.

```
print(coding_dna.translate())
#Seq("MAIVMGR*KGAR*")
print(coding_dna.translate(to_stop=True))
print(coding_dna.translate(table=2))
print(coding_dna.translate(table=2, to_stop=True))
```

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

🔍 **STOP Kodonu (*) Niyə Görünür?**

DNT **üçlü kodonlarla** oxunur və bəzi kodonlar **zülal sintezinin bitdiyini bildirir (STOP kodonu).**

Əsas STOP kodonları:

•TAA, TAG, TGA (Standart kodda)

•TGA → W (Tryptophan) (Mitoxondrial kodda)

```
coding_dna.translate(table=2, stop_symbol="@")
```

```
Seq('MAIVMGRWKGAR@')
```

📌 **Əgər to_stop=True istifadə olunmazsa, STOP kodonları * kimi çıxışda görünür.**

📌 Əgər to_stop=True əlavə etsək, **STOP kodonundan sonrakı hissə atılır.**

Tam kodlaşdırıcı ardıcıllıq (CDS – Complate Coding Sequence) və standart olmayan start kodonları

CDS (Coding Sequence) nədir?

CDS tam bir zülal kodlaşdıran genetik ardıcıllıqdır:

✓ **Uzunluğu 3-ə bölünür** (hər biri bir amin turşusunu kodlayan üçlü kodonlardan ibarətdir).

✓ **Start kodonu (ATG) ilə başlayır** (zülal sintezinin başlama nöqtəsi).

✓ **Stop kodonu (TAA, TAG, TGA) ilə bitir** (zülal sintezinin dayanma nöqtəsi).

✓ **Daxilində STOP kodonu olmur** (əgər olsa, zülal yarımçıq qalar).

Bu cür ardıcıllıq varsa, translate() metodu ilə rahat şəkildə zülala çevrilir.

Standart Olmayan Start Kodonu Nədir?

Bakteriyalarda bəzi genlər ATG (Metionin) əvəzinə başqa kodonlarla başlaya bilər.

Məsələn, **E. coli K12 bakteriyasında "yaaX" geni ATG əvəzinə GTG (Valin) ilə başlayır.**

Standart Olmayan Start Kodonu ilə Problem

Əgər belə bir ardıcıllıqda translate() metodunu istifadə etsək, **əvvəlki kodonların başqa amin turşuları ilə başlayır.**

Standart olaraq ATG Metionin (M) verir, amma bəzi bakteriyalar GTG və ya TTG kodonlarını da start kimi istifadə edir.

Bakteriya üçün spesifik genetik koddan istifadə etdikdə:

print(coding_dna.translate(table=11)) # Bakteriya və Arxeya üçün genetik kod

Bu zaman bəzi qeyri-standart start kodonları da ATG kimi oxunacaq.

Nəticə

📌 **Tam bir kodlaşdırıcı ardıcıllıq (CDS)** varsa, **translate() metodu düzgün işləyəcək.**

📌 **Bakteriyalarda və bəzi orqanizmlərdə standart olmayan start kodonları istifadə olunur.**

📌 **Bakteriya genlərini düzgün oxumaq üçün table=11 kimi uyğun genetik kodlar seçilməlidir.**

[70]:
```
from Bio.Seq import Seq
gene=Seq("GTGAAAAAGATGCAATCTATCGTACTCGCACTTTCCCTGGTTCTGGTCGCTCCCATGGCA"
    "GCACAGGCTGCGGAAATTACGTTAGTCCCGTCAGTAAAATTACAGATAGGCGATCGTGAT"
    "AATCGTGGCTATTACTGGGATGGAGGTCACTGGCGCGACCACGGCTGGTGGAAACAACAT"
    "TATGAATGGCGAGGCAATCGCTGGCACCTACACGGACCGCCGCCACCGCCGCGCCACCAT"
    "AAGAAAGCTCCTCATGATCATCACGGCGGTCATGGTCCAGGCAAACATCACCGCTAA")
print(gene)
```

```
GTGAAAAAGATGCAATCTATCGTACTCGCACTTTCCCTGGTTCTGGTCGCTCCCATGGCAGCACAGGCTGCGGAAATTACGTTAGTCCCGTCAGTAAAATTACAGATAGGCGATCGTGATAATCGTGGCTATTACTGGGATGGAGGTCACTG
GCGCGACCACGGCTGGTGGAAACAACATTATGAATGGCGAGGCAATCGCTGGCACCTACACGGACCGCCGCCACCGCCGCGCCACCATAAGAAAGCTCCTCATGATCATCACGGCGGTCATGGTCCAGGCAAACATCACCGCTAA
```

[72]:
```
print(gene.translate(table="Bacterial"))
```

```
VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR*
```

[74]:
```
gene.translate(table="Bacterial", to_stop=True)
```

[74]:
```
Seq('VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

[76]:
```
gene.translate(table="Bacterial", cds=True)
```

[76]:
```
Seq('MKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

[84]:
```
print(gene)
print(gene.translate(table="Bacterial"))
print(gene.translate(table="Bacterial", to_stop=True))
print(gene.translate(table="Bacterial", cds=True))
```

```
GTGAAAAAGATGCAATCTATCGTACTCGCACTTTCCCTGGTTCTGGTCGCTCCCATGGCAGCACAGGCTGCGGAAATTACGTTAGTCCCGTCAGTAAAATTACAGATAGGCGATCGTGATAATCGTGGCTATTACTGGGATGGAGGTCACTG
GCGCGACCACGGCTGGTGGAAACAACATTATGAATGGCGAGGCAATCGCTGGCACCTACACGGACCGCCGCCACCGCCGCGCCACCATAAGAAAGCTCCTCATGATCATCACGGCGGTCATGGTCCAGGCAAACATCACCGCTAA
VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR*
VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR
MKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR
```

[100]:
```
print (gene.translate(table="Bacterial"))
#Seq('VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HR*',ProteinAlpabet())
```

```
VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR*
```

[104]:
```
gene.translate(table="Bacterial", to_stop=True)
Seq('VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
#Bakterial genetik kodda GTG etibarlı başlanğıc kodonudur və
#normal olaraq Valini kodlasa da, başlanğıc kodonu kimi istifadə edilərsə,
#metionin kimi tərcümə edilməlidir.
```

[104]:
```
Seq('VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

[106]:
```
gene.translate(table="Bacterial", cds=True)
Seq('MKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

[106]:
```
Seq('MKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

[112]:
```
#Translation tables
#Standart translation cədvəli və  Vertebrate Mitoxondrial DNT üçün translation cədvəli.
from Bio.Data import CodonTable
standard_table = CodonTable.unambiguous_dna_by_name["Standard"]
mito_table = CodonTable.unambiguous_dna_by_name["Vertebrate Mitochondrial"]
print(standard_table)
print(mito_table)
```

```
Table 1 Standard, SGC0

  |  T      |  C      |  A      |  G      |
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Stop| TGA Stop| A
T | TTG L(s)| TCG S   | TAG Stop| TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
C | CTC L   | CCC P   | CAC H   | CGC R   | C
C | CTA L   | CCA P   | CAA Q   | CGA R   | A
C | CTG L(s)| CCG P   | CAG Q   | CGG R   | G
--+---------+---------+---------+---------+--
A | ATT I   | ACT T   | AAT N   | AGT S   | T
A | ATC I   | ACC T   | AAC N   | AGC S   | C
A | ATA I   | ACA T   | AAA K   | AGA R   | A
A | ATG M(s)| ACG T   | AAG K   | AGG R   | G
--+---------+---------+---------+---------+--
G | GTT V   | GCT A   | GAT D   | GGT G   | T
G | GTC V   | GCC A   | GAC D   | GGC G   | C
G | GTA V   | GCA A   | GAA E   | GGA G   | A
G | GTG V   | GCG A   | GAG E   | GGG G   | G
--+---------+---------+---------+---------+--

Table 2 Vertebrate Mitochondrial, SGC1

  |  T      |  C      |  A      |  G      |
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Stop| TGA W   | A
T | TTG L   | TCG S   | TAG Stop| TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
C | CTC L   | CCC P   | CAC H   | CGC R   | C
C | CTA L   | CCA P   | CAA Q   | CGA R   | A
C | CTG L   | CCG P   | CAG Q   | CGG R   | G
--+---------+---------+---------+---------+--
A | ATT I(s)| ACT T   | AAT N   | AGT S   | T
A | ATC I(s)| ACC T   | AAC N   | AGC S   | C
A | ATA M(s)| ACA T   | AAA K   | AGA Stop| A
A | ATG M(s)| ACG T   | AAG K   | AGG Stop| G
--+---------+---------+---------+---------+--
G | GTT V   | GCT A   | GAT D   | GGT G   | T
G | GTC V   | GCC A   | GAC D   | GGC G   | C
G | GTA V   | GCA A   | GAA E   | GGA G   | A
G | GTG V(s)| GCG A   | GAG E   | GGG G   | G
--+---------+---------+---------+---------+--
```

Bu cədvəl **genetik kod cədvəlidir** və **mRNA kodonlarının hansı amin turşularına çevrildiyini göstərir**.

Genetik kod cədvəlində hər bir amin turşusu **bir hərflə təmsil olunur**:

Simvol	Amin turşusu	Tam adı
D	Aspartik turşusu	Aspartate (Asp)
E	Qlutamik turşusu	Glutamate (Glu)
K	Lizin	Lysine (Lys)
Q	Qlutamin	Glutamine (Gln)
H	Histidin	Histidine (His)
R	Arginin	Arginine (Arg)
W	Triptofan	Tryptophan (Trp)
L	Leysin	Leucine (Leu)
F	Fenilalanin	Phenylalanine (Phe)
S	Serin	Serine (Ser)
A	Alanin	Alanine (Ala)
M	Metionin	Methionine (Met)
V	Valin	Valine (Val)

❖ **unambiguous_dna_by_name nə edir?**

Biopython kitabxanasında **CodonTable.unambiguous_dna_by_name** metodu **qeyri-müəyyən kodonları olmayan (yəni tam dəqiq) genetik kod cədvəllərini çağırmaq üçün istifadə olunur.**

Bu o deməkdir ki, **yalnız standart kodonlar istifadə olunur və "ambiguous" (yəni qeyri-müəyyən) bazlar yoxdur**. Məsələn:

•**A, T, G, C** bazları tam dəqiqdir (**unambiguous**)

•**Ambiguous bazlar** isə **N, Y, R və s.** kimi müəyyən olunmayan nukleotidləri göstərir (məsələn, **N = A/T/G/C ola bilər**).

```
from Bio.Data import CodonTable

standard_table = CodonTable.unambiguous_dna_by_name["Standard"]
mito_table = CodonTable.unambiguous_dna_by_name["Vertebrate Mitochondrial"]
```

✓ Bu kod iki fərqli genetik kod cədvəlini çağırır:

Standard (Standart genetik kod) – bütün canlılarda, əsasən nüvə DNT-si üçün istifadə olunur.
Vertebrate Mitochondrial (Onurğalılara aid mitoxondrial kod) – mitoxondrilərdə istifadə olunur və bəzi fərqli kodon mənaları var.

- **L(s) → Start kodonu ola bilən leysin**
- **M(s) → Start kodonu ola bilən metionin**

•Normalda **AUG kodonu Metionin (M) üçün istifadə olunur və başlanğıc kodonu (start codon) kimi çıxış edir.**
•Ancaq bəzi bakteriyalarda **L(s) və ya M(s)** başlanğıc kodonu ola bilər.

•**TAA, TAG, TGA → Stop kodonlarıdır (tərcümə dayanır).**
•**ATG → Metionin (M) və həm də Start kodonudur.**

Bir **kodon**, **3 nukleotiddən ibarət bir qrupdur** və bir amin turşusuna uyğun gəlir. Məsələn:
•**AUG → Metionin (M)**
•**UUU → Fenilalanin (F)**
•**UGA → Stop (tərcüməni dayandırır)**

"Ambiguous" genetik kodlar nedir?
Qeyri-müəyyən (ambiguous) kodonları olan genetik kodlar da istifadə edilə bilər. Bunun üçün ambiguous_dna_by_name metodu var.

Məsələn:

python
Copy
Edit
from Bio.Data import CodonTable

ambiguous_table = CodonTable.ambiguous_dna_by_name["Standard"]
print(ambiguous_table)
Bu versiya qeyri-müəyyən bazlar da qəbul edir (məsələn, N, Y, R kimi kodlar).

✅ "Unambiguous" o deməkdir ki, yalnız A, T, G, C bazlarından istifadə edilir.
✅ unambiguous_dna_by_name dəqiq və tam müəyyən edilmiş genetik kod cədvəllərini çağırmaq üçün istifadə olunur.

```
[ ]: #stop ve start codonlari...
mito_table.stop_codons
['TAA', 'TAG', 'AGA', 'AGG']
mito_table.start_codons
['ATT', 'ATC', 'ATA', 'ATG', 'GTG']
mito_table.forward_table["ACG"]
'T'
```

```
[114]: print(mito_table.stop_codons)
```

```
['TAA', 'TAG', 'AGA', 'AGG']
```

```
[116]: print(mito_table.start_codons)
```

```
['ATT', 'ATC', 'ATA', 'ATG', 'GTG']
```

```
[118]: print(mito_table.forward_table)
```

```
{'TTT': 'F', 'TTC': 'F', 'TTA': 'L', 'TTG': 'L', 'TCT': 'S', 'TCC': 'S', 'TCA': 'S', 'TCG': 'S', 'TAT': 'Y', 'TAC': 'Y', 'TGT': 'C', 'TGC': 'C', 'TGA':
'W', 'TGG': 'W', 'CTT': 'L', 'CTC': 'L', 'CTA': 'L', 'CTG': 'L', 'CCT': 'P', 'CCC': 'P', 'CCA': 'P', 'CCG': 'P', 'CAT': 'H', 'CAC': 'H', 'CAA': 'Q', 'CA
G': 'Q', 'CGT': 'R', 'CGC': 'R', 'CGA': 'R', 'CGG': 'R', 'ATT': 'I', 'ATC': 'I', 'ATA': 'M', 'ATG': 'M', 'ACT': 'T', 'ACC': 'T', 'ACA': 'T', 'ACG': 'T',
'AAT': 'N', 'AAC': 'N', 'AAA': 'K', 'AAG': 'K', 'AGT': 'S', 'AGC': 'S', 'GTT': 'V', 'GTC': 'V', 'GTA': 'V', 'GTG': 'V', 'GCT': 'A', 'GCC': 'A', 'GCA':
'A', 'GCG': 'A', 'GAT': 'D', 'GAC': 'D', 'GAA': 'E', 'GAG': 'E', 'GGT': 'G', 'GGC': 'G', 'GGA': 'G', 'GGG': 'G'}
```

```
[ ]:
```

Seq obyektlərinin müqayisə edilməsi

[128]:
```
#Seq obyektlərinin müqayisə edilməsi
from Bio.Seq import Seq
seq1 = Seq("ACGT")
"ACGT" == seq1
print("ACGT"==seq1)
print(seq1=="ACGT")
```

```
True
True
```

Ardıcıllığı naməlum olan ardıcıllıqlar

Bəzi bioinformatik fayllarda (məsələn, **GenBank və EMBL**) bəzən **tam genomik DNT ardıcıllığı verilmir**, əksinə, **yalnız konfiqurasiya (struktur) məlumatı mövcuddur**. Bu o deməkdir ki, **ardıcıllığın uzunluğu bilinir, lakin konkret olaraq hansı hərflərdən (nukleotidlərdən) ibarət olduğu göstərilmir**.

Bu nə üçün istifadə olunur?

•**Bütün genomu tam göstərmək lazım deyil** – sadəcə uzunluğu bilmək kifayət edir.

•**Məlumat böyük olduqda yaddaşa qənaət etmək üçün istifadə edilə bilər**.

•**Sekans hələ məlum deyil və ya araşdırılma prosesindədir. Beləliklə, Seq(None, length=1000) bir ardıcıllıq yaratmış olur, amma içi boşdur və yalnız 1000 nukleotiddən ibarət olduğu məlumdur.**

```
from Bio.Seq import Seq
seq_unknown = Seq(None, length=1000)  # 1000 uzunluğunda, lakin içi boş olan bir Seq obyekti
print("seq_unknown")
print(len("seq_unknown"))  # 1000 çap edəcək
```

```
seq_unknown
11
```

MAF (Multiple Alignment Format)

- **MAF (Multiple Alignment Format)** bioinformatikada **birdən çox genom ardıcıllığını müqayisə etmək üçün istifadə edilən bir formatdır**.
- Müxtəlif orqanizmlərin **DNT sekanslarının hizalanmasını** göstərir.
- Genetik təkamül, mutasiyalar və ortolog genlərin tapılması üçün istifadə edilir.

Strukturun İzahı

Hər sətirdə fərqli bir orqanizmə aid genomik məlumat var.

Burada **insan (hg38)**, **şimpanze (panTro4)**, **makaque (rheMac3)**, **siçan (mm10)**, **siçovul (rn5)**, **it (canFam3)** və **opossum (monDom5)** genomları verilmişdir.

Hər sətrin formatı belədir:

```
#MAF (Multiple Alignment Format)
s hg38.chr7     117512683 36 + 159345973 TTGAAAACCTGAATGTGAGAGTCAGTCAAGGATAGT
s panTro4.chr7  119000876 36 + 161824586 TTGAAAACCTGAATGTGAGAGTCACTCAAGGATAGT
s rheMac3.chr3  156330991 36 + 198365852 CTGAAATCCTGAATGTGAGAGTCAATCAAGGATGGT
s mm10.chr6      18207101 36 + 149736546 CTGAAAACCTAAGTAGGAGAATCAACTAAGGATAAT
s rn5.chr4       42326848 36 + 248343840 CTGAAAACCTAAGTAGGAGAGACAGTTAAAGATAAT
s canFam3.chr14  56325207 36 +  60966679 TTGAAAAACTGATTATTAGAGTCAATTAAGGATAGT
s monDom5.chr8  173163865 36 + 312544902 TTAAGAAACTGGAAATGAGGGTTGAATGACAAACTT
```

Hər sətrin formatı belədir:

s <genom adı> <xromosom> <başlanğıc pozisiyası> <uzunluq> <zolaq (strand)> <ümumi uzunluq> <DNT sekansı>

Nümunəyə baxaq:

s hg38.chr7 117512683 36 + 159345973 TTGAAAACCTGAATGTGAGAGTCAGTCAAGGATAGT

•s → sekans məlumatı (bu satırın ardıcıllıq olduğunu göstərir)

•hg38.chr7 → İnsan genomu, xromosom **7**

•117512683 → Bu hissənin **xromosom üzərindəki başlanğıc mövqeyi**

•36 → **Sekansın uzunluğu** (36 nukleotid)

•+ → **Strand yönü** (ön strand "+" və ya tərs strand "-")

•159345973 → Bu xromosomun ümumi uzunluğu

•TTGAAAACCTGAATGTGAGAGTCAGTCAAGGATAGT → **DNT sekansı**

Bu **müxtəlif növlərdə eyni DNT bölgəsinin müqayisəsini** göstərir.

•**İnsan** və **şimpanze** genomları çox oxşardır.

•**Siçan və siçovul** genomlarında bəzi fərqlər var.

•**Opossum** isə daha çox fərqlənir.

Bütün bu məlumat nə üçün lazımdır?

•**Genetik təkamülü anlamaq** (hansı növlər daha yaxındır?)

•**Təkamül prosesində hansı mutasiyalar baş verib?**

•**Xəstəliklərin genetik səbəblərini araşdırmaq**

•**Hansı genlərin qorunduğunu və funksional olduğunu anlamaq**

• Bu cür çoxlu hizalama (multiple alignment) tədqiqatları genetik analizlərdə, xüsusilə də təkamül, xəstəliklərin genetik səbəbləri və gen funksiyalarını anlamaq üçün istifadə olunur.

MutableSeq

MutableSeq, kolleksiyanın sırasını dəyişdirməyə, elementləri əlavə edib silməyə imkan verir.

MutableSeq obyektləri, adətən Python proqramlaşdırmasında verilənlər strukturları ilə işləyərkən istifadə olunan və elementlərinin dəyişdirilməsinə imkan verən funksiya növüdür. Bu cür funksiyalar, verilənlər üzərində edilən dəyişikliklərin birbaşa dodun özündə dəyişiklik etməsinə imkan tanıyır.

Python-da MutableSeq, ümumiyyətlə collections.abc modulunda tərtib olunan bir abstrakt sinifdir. Bu sinif, Python-da kolleksiyaları və siyahıları təmsil etmək üçün istifadə olunan bir əsas sinifdir. MutableSeq, bir siyahını təmsil edir və elementləri üzərində birbaşa dəyişiklik etməyə imkan verir (əlavə etmə, silmə, yeniləmə və s.).

[160]:
```
#MutableSeq objects
from Bio.Seq import Seq
my_seq = Seq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA")
from Bio.Seq import MutableSeq
mutable_seq = MutableSeq(my_seq)
MutableSeq('GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA')
```

[160]:
```
MutableSeq('GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA')
```

[162]:
```
mutable_seq[5]="C"
```

[164]:
```
print(mutable_seq)
```

```
GCCATCGTAATGGGCCGCTGAAAGGGTGCCCGA
```

[166]:
```
#remove funkisyasi T nukleotidini cixaracaq.
mutable_seq.remove("T")
print(mutable_seq)
```

```
GCCACGTAATGGGCCGCTGAAAGGGTGCCCGA
```

[170]:
```
#reverse funksiyasi
mutable_seq.reverse()
print(mutable_seq)
```

```
AGCCCGTGGGAAAGTCGCCGGGTAATGCACCG
```

[180]:
```
#Stringlerle islemek...
from Bio.Seq import reverse_complement, transcribe, back_transcribe, translate
my_string = "GCTGTTATGGGTCGTTGGAAGGGTGGTCGTGCTGCTGGTTAG"
print(reverse_complement(my_string))
print(transcribe(my_string))
print(back_transcribe(my_string))
print(translate(my_string))
```

```
CTAACCAGCAGCACGACCACCCTTCCAACGACCCATAACAGC
GCUGUUAUGGGUCGUUGGAAGGGUGGUCGUGCUGCUGGUUAG
GCTGTTATGGGTCGTTGGAAGGGTGGTCGTGCTGCTGGTTAG
AVMGRWKGGRAAG*
```

[182]:
```
#Alt ardıcıllıqların tapılması
from Bio.Seq import Seq, MutableSeq
seq = Seq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA")
print(seq.index("ATGGGCCGC"))
print(seq.index(b"ATGGGCCGC"))
print(seq.index(bytearray(b"ATGGGCCGC")))
print(seq.index(Seq("ATGGGCCGC")))
print(seq.index(MutableSeq("ATGGGCCGC")))
```

```
9
9
9
9
9
```

1.Seq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA"): Bu, Biopython kitabxanasının Seq sinifindən istifadə edərək bir genetik ardıcıllığı (GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA) təmsil edir.
2.seq.index("ATGGGCCGC"): Bu, Seq obyektində "ATGGGCCGC" ardıcıllığının başlanğıc indeksini tapır. Çıxış 9 olacaq, çünki "ATGGGCCGC" ardıcıllığı seq obyektində 9-cu mövqedən başlayır.
3.seq.index(b"ATGGGCCGC"): Burada "ATGGGCCGC" ardıcıllığı bytes formatında verilmişdir (yəni b"ATGGGCCGC"). Biopython bytes formatını da qəbul edir və bu da 9 indeksini qaytarır, çünki bu ardıcıllıq da seq obyektində 9-cu mövqedən başlayır.
4.seq.index(bytearray(b"ATGGGCCGC")): Bu, "ATGGGCCGC" ardıcıllığının bytearray formatında verilməsidir. bytearray, bytes-in dəyişdirilə bilən versiyasıdır və bu da 9 indeksini qaytarır.
5.seq.index(Seq("ATGGGCCGC")): Burada "ATGGGCCGC" ardıcıllığı Seq obyektinə çevrilərək seq obyektində axtarılır. Bu da yenə 9 indeksini verir, çünki Seq obyektləri də uyğunlaşdırıla bilir.
6.seq.index(MutableSeq("ATGGGCCGC")): Burada "ATGGGCCGC" ardıcıllığı MutableSeq obyektinə çevrilərək seq obyektində axtarılır. Bu da 9 indeksini qaytarır, çünki MutableSeq də Seq obyektinə bənzər şəkildə işləyir və uyğunlaşdırıla bilir.

Nəticə:
Bu əməliyyatlar göstərir ki, Biopython, müxtəlif verilən formatları (string, bytes, bytearray, Seq, MutableSeq) qarşılaşdırmaq və onların indekslərini tapmaq üçün çox çevikdir. index() metodu verilən ardıcıllığı təmsil edən obyektin növündən asılı olmayaraq, ardıcıllığın yerləşdiyi indeksin dəyərini qaytarır. Bu halda, bütün bu əməliyyatlar "ATGGGCCGC" ardıcıllığının seq obyektindəki indeksini 9 olaraq qaytarır.

[198]:
```
for index, sub in seq.search(["CC", "GGG", "CC"]):
    print(index, sub)
seq.find("ACTG")
seq.find("CC")
seq.rfind("CC")
#GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA
```

```
1 CC
11 GGG
14 CC
23 GGG
28 CC
29 CC
```

[198]:
```
29
```

Bu kod parçası, **Biopython** kitabxanasının search() metodunun necə işlədiyini göstərir. Bu metod, verilən ardıcıllıqların (sub-sekansların) müəyyən bir genetik ardıcıllıqda (Seq obyektində) hansı mövqelərdə yerləşdiyini tapmaq üçün istifadə olunur.

Kodun anlamını açıqlayaq:

1.seq.search(["CC", "GGG", "CC"]):

- search() metodu, ardıcıllığın (seq) içindəki hər bir sub-sekansı (bu halda ["CC", "GGG", "CC"] siyahısındakı ardıcıllıqları) axtarır.
- Bu metod, verilən ardıcıllıqların içində olduğu mövqeləri və hər bir sub-sekansı tapır.

2.for index, sub in seq.search(["CC", "GGG", "CC"])::

- Burada dövr (for), search() metodunun döndərdiyi nəticəni hər bir sub-sekans və onun tapıldığı indeksi götürərək işləyir.
- index, sub-sekansın başladığı indeks mövqeyini göstərir.
- sub, tapılan sub-sekansdır.

3.print(index, sub):

- Hər tapılan sub-sekansın indeksini və sub-sekansın özünü ekrana çap edir.

Nəticə:

Bu əməliyyat seq ardıcıllığında ["CC", "GGG", "CC"] ardıcıllıqlarını axtarır və tapıldığı hər mövqeyi və sub-sekansı çap edir. Çıxışda:

- CC sub-sekansı 1-ci, 14-cü və 28-ci indekslərdə tapılıb.
- GGG sub-sekansı 11-ci və 23-cü indekslərdə tapılıb.