Proje 1

1- Simple_example.py uygulaması

```
C:\Users\betul\OneDrive\Masaüstü\biyoinf\proje1>python simple_example.py
Id: sp|P25730|FMS1_ECOLI
Name: sp|P25730|FMS1_ECOLI
Description: sp|P25730|FMS1_ECOLI CS1 fimbrial subunit A precursor (CS1 pilin)
Annotations: {}
Sequence Data:
MKLKKTIGAMALATLFATMGASAVEKTISVTASVDPTVDLLQSDGSALPNSVALTYSPAVNNFEAHTINTVVHTNDSDKGVVVKLSADP
VLSNVLNPTLQIPVSVNFAGKPLSTTGITIDSNDLNFASSGVNKVSSTQKLSIHADATRVTGGALTAGQYQGLVSIILTKSTTTTTTK
GT
Sequence Alphabet: SingleLetterAlphabet()
Id: sp|P15488|FMS3_ECOLI
Name: sp|P15488|FMS3_ECOLI
Description: sp|P15488|FMS3_ECOLI CS3 fimbrial subunit A precursor (CS3 pilin)
Annotations: {}
Sequence Data:
MLKIKYLLIGLSLSAMSSYSLAAAGPTLTKELALNVLSPAALDATWAPQDNLTLSNTGVSNTLVGVLTLSNTSIDTVSIASTNVSDTSK
NGTVTFAHETNNSASFATTISTDNANITLDKNAGNTIVKTTNGSQLPTNLPLKFITTEGNEHLVSGNYRANITITSTIKGGGTKKGTTD
KK
Sequence Alphabet: SingleLetterAlphabet()

```

Fasta uzantılı dosyalar biyoinformatik ve biyokimyada sıkça kullanılan dosya uzantı formatıdır. .fasta uzantılı dosyalar nükleotid ya da amino asitlerin tek harfli kodlar kullanılarak temsil edildiği dizilerin gösterildiği bir dosya formatıdır. Dizilerin başında “>” işareti kullanılarak yazılmış yerler o dizilime verilmiş olan ismi temsil eder. Bu dizilimlerinin her birinin kendine özgü kimliği, adı, açıklaması ve veri dizilimi vardır.

Simple_example.py dosyasında yazdığımız kod parçasında example.fasta dosyasında yazılı olan dizilimlerin her birinin özelliklerini yazdırdık. Yukarıdaki terminal ekranında simple_example.py dosyasına “file = open(“example.fasta”)” komutu ile example.fasta dosyasını python içerisine aldık ve sonrasında kullandığımız “parse” tanımı ile tanımladığımız records değişkenine atayarak kolayca işleme alıp fasta uzantılı dosyanın içerisindeki dizilimlerinin özelliklerini çözümledik.

2- Seq uygulaması;

```
C:\Users\betul>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from Bio.Seq import Seq
>>> seq = Seq("AGCT")
>>> seq
Seq('AGCT')
>>> print(seq)
AGCT

```

Bir ‘sequence’ protein, DNA ya da RNA’yı temsil eden harflerin birleşiminden oluşan bir dizidir. Bio.Seq modülünden Seq sınıfının import edilmesi ile tanımlanır. Bu uygulamada ‘seq’ adlı string parametreyi dizi nesnesine döndürmek için Seq() kullandık.

3- Alphabet uygulaması;

Alphabet modülü Seq nesnesi, dizilerin tipini, harflerini ya da olası diğer işlemleri belirtmek istediğinde kullanılır. Bio.Alphabetten tanımlanmıştır. Alphabet sınıfları bize tanımlamış olduğumuz dizinin bir protein dizisi mi, dna dizisi mi yoksa rna dizisi mi olduğunu belirtir. Farklı dizilimlere göre Alphabet sınıfı belirtilmediği sürece çıktı ekranında Alphabet() olarak gözükür. Aşağıda yaptığımız ilk uygulamada myseq nesnesinin hangi dizilim olduğunu Seq sınıfı içerisinde tanımlamadığımız için sonuçta genel Alphabet() sınıfı gözüktü.

```
>>> from Bio.Seq import Seq
>>> myseq = Seq("AGCT")
>>> myseq
Seq('AGCT')
>>> myseq.alphabet
Alphabet()
```

```
SingleLetterAlphabet
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import single_letter_alphabet
>>> test_seq = Seq('AGTACACTGGT', single_letter_alphabet)
>>> test_seq
Seq('AGTACACTGGT', SingleLetterAlphabet())
```

SingleLetterAlphabet dizideki harflerin boyutunun bir olmasıdır.

```
ProteinAlphabet
>>> from Bio.Seq import Seq
>>> from Bio.Alphabet import generic_protein
>>> test_seq = Seq('AGTACACTGGT', generic_protein)
>>> test_seq
Seq('AGTACACTGGT', ProteinAlphabet())
```

Üstteki Alphabet uygulamasında test_seq nesnesini Seq() sınıfı içerisinde tanımlarken protein sequence olduğunu belirttiğimiz için (bu belirtmede Bio.Alphabet ‘ten generic_proteini içe aktarmamız) sonradan çağırdığımızda terminal ekranında bize çıktı olarak ProteinAlphabet() sınıfını gösterdi.

Nucleotide Alphabet uygulamasında Bio.Alphabet ’ten generic_nucleotide içe aktararak tanımladığımız sequence nesnesinin nükleotid sequence olduğunu belirtmiş olduk. Ekranda çıktı olarak NucleotideAlphabet() sınıfı gözükür.

4- Bio.Data uygulaması;

```
>>> from Bio.Data import IUPACData
>>> IUPACData.protein_letters
'ACDEFGHIKLMNPQRSTVWY'
```

Bio.Data çeşitli faydalı veri bitlerinin koleksiyonudur. Biyoinformatikle ilgili yapılandırma verilerini sunar. IUPAC ise bir adlandırma sistemidir. Kimyasal bileşenleri adlandırılmasında ve çoğunlukla kimya biliminde kullanılır. Bu

uygulamada 2.satırda IUPACData.protein_letters komutunu yazarak ekrana bir protein dizilimi yazdırıldı. Bu komutun işlevi IUPACProtein alfabesinin olası harflerine sahiptir.

```
1.  Basic Operations
2.  >>> seq_string = Seq("AGCTAGCT")
3.  >>> seq_string[0]
4.  'A'
5.  >>> seq_string[0:2]
6.  Seq('AG')
7.  >>> seq_string[ : ]
8.  Seq('AGCTAGCT')
9.  >>> len(seq_string)
10. 8
11. >>> seq_string.count('A')
12. 2
13.
```

5- Yukarıdaki uygulamada python kodu yazarken kullanılan işlevler ile tanımlanan değişken üzerinden uygulama yaptım. 2.satırda string değişkeni Seq() sınıfı içerisine atayarak bir sequence nesnesi haline getirdik. Bir sonraki satırda bu dizilimin 0. İndisinin hangi harfe karşılık geldiğini sorguladık ve ekrana dizinin ilk harfi olan 'A' yazıldı. Sonraki satırda dizilimin ilk indisi ile 2. İndisi arasındaki harfleri sorguladık ve ekrana 'AG' şeklinde yazıldı. Sonrasında pythonda sayı tanımlaması yapılmadan köşeli parantezler içerisine iki nokta(:) işaretinin konulmasıyla tanımlanan tüm dizi ekrana yazdırıldığı için buradaki uygulamada da aynı şekilde tanımlama yapılarak tüm sequence nesnesinin ekrana yazdırılması sağlandı. 9.satırda tanımlanan nesnenin 'len' fonksiyonu kullanılarak uzunluğu istenildi ve ekrana nesnenin uzunluğu yazıldı. Son olarak dizilimin içerisinde bulunan A harfi sayısı count ifadesi kullanılarak soruldu ve ekrana 2 sayısı yazdırıldı.

6- İki dizilimin eklendiği uygulama;

```
1.  >>> from Bio.Alphabet import generic_dna, generic_protein
2.  >>> seq1 = Seq("AGCT", generic_dna)
3.  >>> seq2 = Seq("TCGA", generic_dna)
4.  >>> seq1 + seq2
5.  Seq('AGCTTCGA', DNAAlphabet())
6.  >>> dna_seq = Seq('AGTACACTGGT', generic_dna)
7.  >>> protein_seq = Seq('AGUACACUGGU', generic_protein)
8.  >>> dna_seq + protein_seq
9.  Traceback (most recent call last):
10.   File "<stdin>", line 1, in <module>
11.   File "C:\Users\betul\AppData\Local\Programs\Python\Python39\lib\site-
    packages\Bio\Seq.py", line 309, in __add__
12.     raise TypeError(
13. TypeError: Incompatible alphabets DNAAlphabet() and ProteinAlphabet()
14.
```

Yukarıdaki uygulamada ilk olarak Bio.Alphabet modülünden generic_dna ve generic_protein içeri aktarılmıştır. Böylelikle DNAAlphabet() ve ProteinAlphabet() sınıflarının kullanılacağı belirtilmiştir. 2. ve 3. satırda iki farklı DNA dizilimi tanımlanmış ve bir dna dizilimi olmaları için Seq() sınıfı içerisine aktarılmıştır. 4.satırda bu iki farklı DNA diziliminin birbirlerine eklenmeleri istenmiş ve 5.satırda bu iki farklı DNA dizilimi (ikisi de DNA oldukları için birbirlerine eklenebilirler.) birbirlerine eklenerek ekrana yazdırılmış ve hangi Alphabet sınıfı içerisinde olduğu da belirtilmiştir.

İkinci uygulamada (6. ve 14.satırlar arası) iki farklı dizilim tanımlanmış, bunlardan birisi DNA dizilimi diğeri ise protein dizilimidir. 8.komut satırında bu iki farklı dizilimin birbirlerine eklenmesi komutu çağrılmış ve sonucun ekrana yazdırılması istenmiştir ancak burada bu iki farklı dizilim birbirlerine eklenemez. Program bu komut

çalıştırılmak istenildiğinde hata vermiştir çünkü bu iki dizilim birbirleriyle uyumlu değildir. Bir dizilim DNA dizilimi iken diğer dizilim protein dizilimidir. Verilen hata da bu iki dizilimin farklı Alphabet() sınıflarından olduğu, uyumsuz oldukları belirtilmiştir.

7- For ile uygulama;

```
1.  >>> from Bio.Alphabet import generic_dna
2.  >>> list = [Seq("AGCT", generic_dna), Seq("TCGA", generic_dna), Seq("AAA", generic_dna)]
3.  >>> for s in list:
4.  ...     print(s)
5.  ...
6.  AGCT
7.  TCGA
8.  AAA
9.  >>> final_seq= Seq(" ", generic_dna)
10. >>> for s in list:
11. ...     final_seq = final_seq + s
12. ...
13. >>> final_seq
14. Seq(' AGCTTCGAAAA', DNAAlphabet())
15.
```

Bu uygulamada ilk olarak Bio.Alphabet içinden generic_dna içe aktarıldı. Sonrasında list adında içerisinde 3 farklı dna dizilimi bulunan bir dizi tanımlandı. 3.satırda for döngüsü ile liste içindeki dizilimleri sırasıyla yazdırmak için değişken tanımlaması yapıldı ve print komutu ile dna dizilimleri sırasıyla ekrana yazdırıldı. 9.satırda final_seq isimli içerisinde sadece boşluk karakteri bulunan string ifade Seq() sınıfı içerisine dna dizilimi olarak tanımlandı ve tekrardan bir for döngüsü ile tüm dizilimlerin sırasıyla birbirlerinin sonuna eklenerek yazılması sağlandı. 14.satırda da hangi alfabe sınıfına ait olduğu ekranda gösterilmiş oldu.

8- RNA uygulaması;

```
1.  >>> from Bio.Alphabet import generic_rna
2.  >>> rna = Seq("agct", generic_rna)
3.  >>> rna.upper()
4.  Seq('AGCT', RNAAlphabet())
5.  >>> rna = Seq("agct", generic_rna)
6.  >>> 'a' in rna
7.  True
8.  >>> 'A' in rna
9.  False
10. >>> rna1 = Seq("AGCT", generic_dna)
11. >>> rna is rna1
12. False
13.
```

Bu uygulamada RNA dizilimi ve RNA dizilimi ile DNA dizilimi arasındaki fark gösterilmiştir. DNA çift zincirden oluşurken, RNA tek zincirden oluşur ve RNA, DNA'nın şifrelerine göre sentezlenir. Uygulamanın başlangıcında RNA içeri aktarılmıştır, sonrasında bir RNA dizilimi tanımlanmış olup bu RNA dizilimi upper ifadesi kullanılarak DNA formuna getirilmiştir. 5. Ve 9. Satırlar arasında RNA içerisinde kontrol yapılmıştır. RNA'nın içerisinde olan harflerin durumuna göre true ya da false ifadesi ekrana yazdırılmıştır. 10.satırda rna1 adında farklı bir dizilim tanımlanmıştır fakat bu dizilimin ismi rna1 olsada DNA dizilimi olarak tanımlanmıştır. 11.satırda ilk tanımlanan rna diziliminin sonradan tanımlanan rna1 dizilimi ile aynı olup olmadığı sorgulanmış ve ekrana FALSE ifadesi yazdırılarak bu iki dizilimin aynı harfler kullanılsa da(boyut farkına bakmadan) farklı boyutlarda olması ile farklı Alphabet() sınıflarına ait oldukları görülmüştür.

9- Harf bulma uygulaması;

```
>>> protein_seq = Seq('AGUACACUGGU', generic_protein)
>>> protein_seq.find('G')
1
>>> protein_seq.find('GG')
8
```

10- Belirli harften ayırma uygulaması ;

```
>>> protein_seq = Seq('AGUACACUGGU', generic_protein)
>>> protein_seq.split('A')
[Seq('', ProteinAlphabet()), Seq('GU', ProteinAlphabet()), Seq('C', ProteinAlphabet()),
Seq('CUGGU', ProteinAlphabet())]
```

Bu uygulamada tanımlanan protein dizi nesnesinin içerisindeki dizilimde her bir 'A' harfinden ayrışma yapılmıştır. İlk 'A' harfinde ilk ayrışmadan öncesinde bir harf dizilimi olmadığı için boşluk karakteri sonrasında ise diğer gelen karakterler ekrana yazdırılmıştır. Tanımladığımız sequence dizilimi bir protein dizilimi olduğundan ProteinAlphabet() sınıfı ekran çıktısında gösterilmiştir.

11- Boşluk karakteri kaldırma uygulaması ;

```
>>> strip_seq = Seq("    AGCT    ")
>>> strip_seq
Seq('    AGCT    ')
>>> strip_seq.strip()
Seq('AGCT')
```

Pythonda strip methodu, tanımlanmış olan string değerin başındaki ve sonundaki boşluk karakterinin ortadan kalkmasını sağlayan bir methodtur. Bu uygulamada tanımlanmış olan strip_seq isimli string değer Seq() sınıfı içerisine aktarılarak bir dizilim haline getirilmiştir. 2.satırda tanımlanmış olan dizilimin ilk hali yani boşluk karakterlerinin bulunduğu hali ekrana yazdırılmıştır, sonrasında strip() methodu kullanılarak dizilimin başındaki ve sonundaki boşluk karakterlerinin ortadan kaldırılarak ekrana yazdırılması sağlanmıştır.