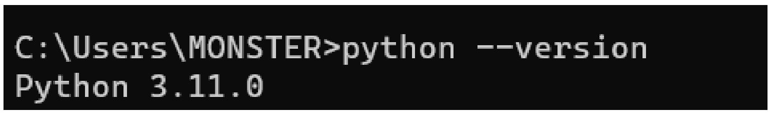
Biyoinformatiğe Giriş Proje 1:

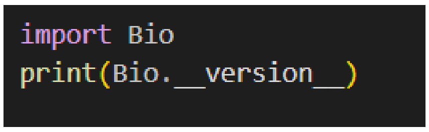
Derya Öztürk

STEP 1: Python Version:

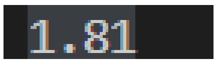


```
C:\Users\MONSTER>python --version
Python 3.11.0
```

STEP 2:



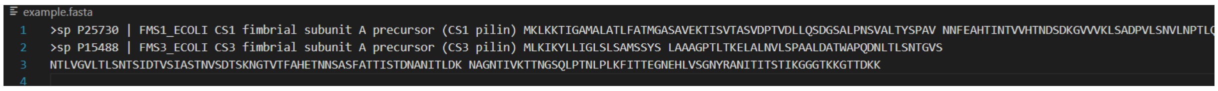
```python
import Bio
print(Bio.__version__)
```



```
1.81
```

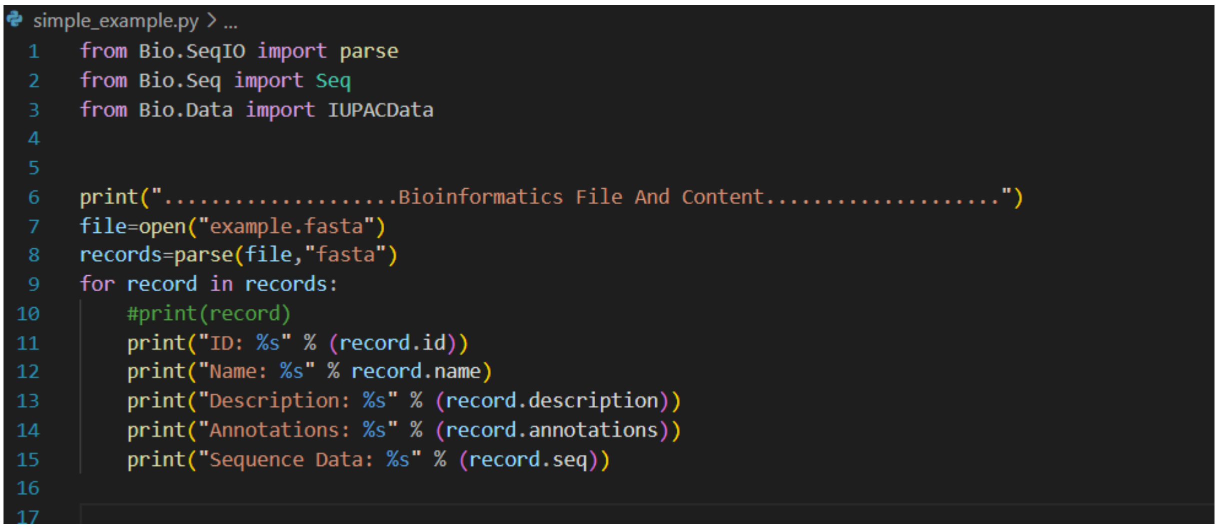
Creating Simple Application

STEP 1:



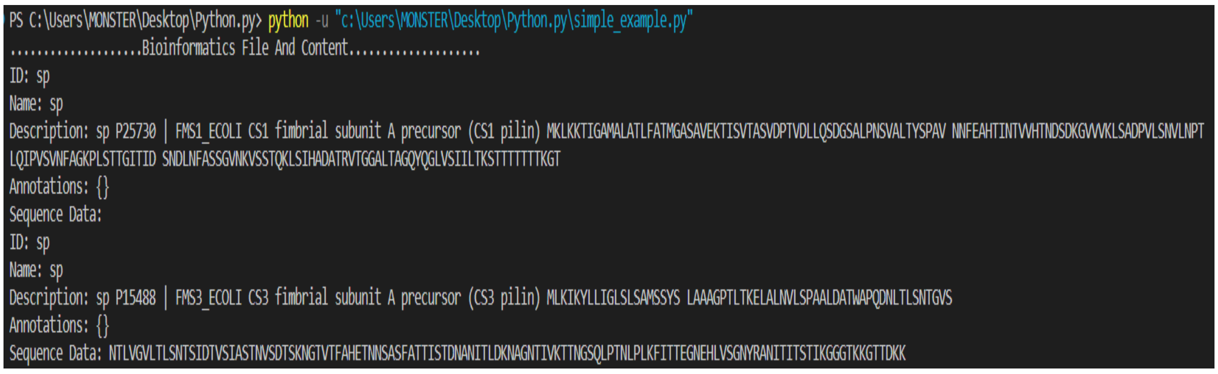
```
>sp P25730 | FMS1_ECOLI CS1 fimbrial subunit A precursor (CS1 pilin) MKLKKTIGAMALATLFATMGASAVEKTISVTASVDPTVDLLQSDGSALPNSVALTYSPAV NNFEAHTINTVVHTNDSDKGVVVKLSADPVLSNVLNPTLQ
>sp P15488 | FMS3_ECOLI CS3 fimbrial subunit A precursor (CS3 pilin) MLKIKYLLIGLSLSAMSSYS LAAAGPTLTKELALNVLSPAALDATWAPQDNLTLSNTGVS
NTLVGVLTLSNTSIDTVSIASTNVSDTSKNGTVTFAHETNNSASFATTISTDNANITLDK NAGNTIVKTTNGSQLPTNLPLKFITTEGNEHLVSGNYRANITITSTIKGGGTKKGTTDKK
```

STEP 2,3,4:



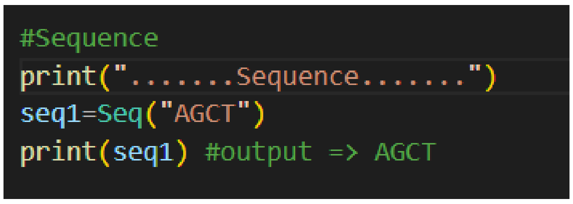
```python
from Bio.SeqIO import parse
from Bio.Seq import Seq
from Bio.Data import IUPACData


print("...................Bioinformatics File And Content...................")
file=open("example.fasta")
records=parse(file,"fasta")
for record in records:
    #print(record)
    print("ID: %s" % (record.id))
    print("Name: %s" % record.name)
    print("Description: %s" % (record.description))
    print("Annotations: %s" % (record.annotations))
    print("Sequence Data: %s" % (record.seq))

```

Output:



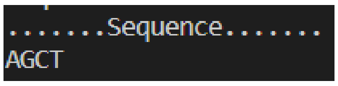
```
PS C:\Users\MONSTER\Desktop\Python.py> python -u "c:\Users\MONSTER\Desktop\Python.py\simple_example.py"
...................Bioinformatics File And Content...................
ID: sp
Name: sp
Description: sp P25730 | FMS1_ECOLI CS1 fimbrial subunit A precursor (CS1 pilin) MKLKKTIGAMALATLFATMGASAVEKTISVTASVDPTVDLLQSDGSALPNSVALTYSPAV NNFEAHTINTVVHTNDSDKGVVVKLSADPVLSNVLNPT
LQIPVSVNFAGKPLSTTGITID SNDLNFASSGVNKVSSTQKLSIHADATRVTGGALTAGQYQGLVSIILTKSTTTTTTTKGT
Annotations: {}
Sequence Data:
ID: sp
Name: sp
Description: sp P15488 | FMS3_ECOLI CS3 fimbrial subunit A precursor (CS3 pilin) MLKIKYLLIGLSLSAMSSYS LAAAGPTLTKELALNVLSPAALDATWAPQDNLTLSNTGVS
Annotations: {}
Sequence Data: NTLVGVLTLSNTSIDTVSIASTNVSDTSKNGTVTFAHETNNSASFATTISTDNANITLDKNAGNTIVKTTNGSQLPTNLPLKFITTEGNEHLVSGNYRANITITSTIKGGGTKKGTTDKK
```

Sequence:



```
#Sequence
print(".......Sequence.......")
seq1=Seq("AGCT")
print(seq1) #output => AGCT
```

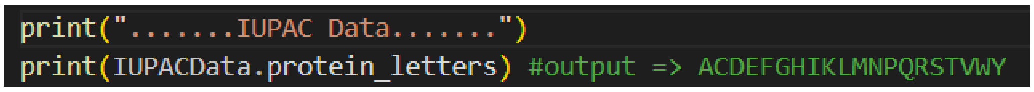
Output:



```
.......Sequence.......
AGCT
```

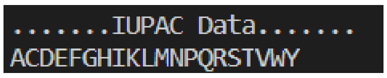
Alphabet Module:

Biopython version 1.77'den sonra Alphabet modülü kaldırılmıştır.



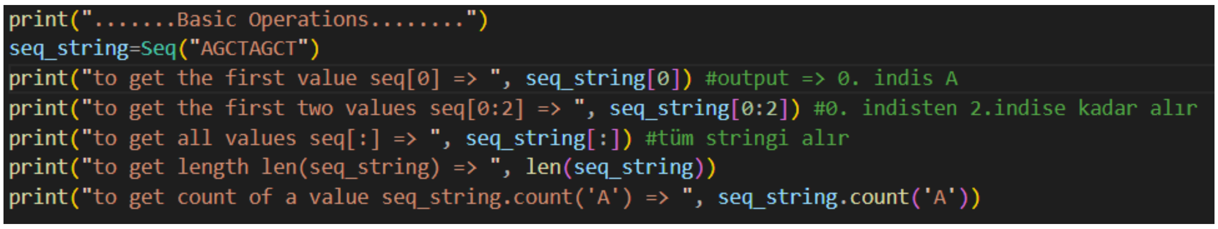
```
print(".......IUPAC Data.......")
print(IUPACData.protein_letters) #output => ACDEFGHIKLMNPQRSTVWY
```

Output:



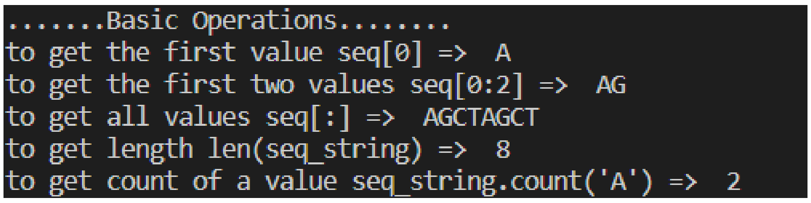
```
.......IUPAC Data.......
ACDEFGHIKLMNPQRSTVWY
```

Basic Operations:



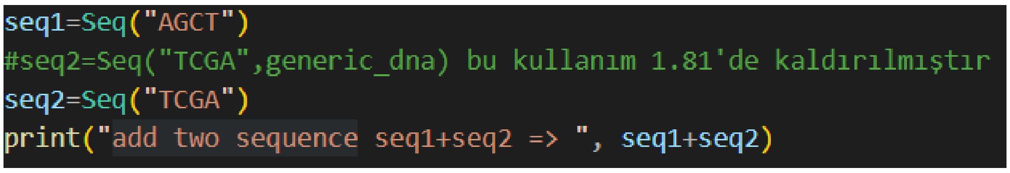
```
print(".......Basic Operations........")
seq_string=Seq("AGCTAGCT")
print("to get the first value seq[0] => ", seq_string[0]) #output => 0. indis A
print("to get the first two values seq[0:2] => ", seq_string[0:2]) #0. indisten 2.indise kadar alır
print("to get all values seq[:] => ", seq_string[:]) #tüm stringi alır
print("to get length len(seq_string) => ", len(seq_string))
print("to get count of a value seq_string.count('A') => ", seq_string.count('A'))
```

Output:



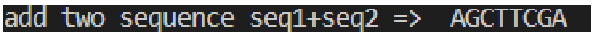
```
.......Basic Operations........
to get the first value seq[0] =>  A
to get the first two values seq[0:2] =>  AG
to get all values seq[:] =>  AGCTAGCT
to get length len(seq_string) =>  8
to get count of a value seq_string.count('A') =>  2
```

Add two sequence:



```
seq1=Seq("AGCT")
#seq2=Seq("TCGA",generic_dna) bu kullanım 1.81'de kaldırılmıştır
seq2=Seq("TCGA")
print("add two sequence seq1+seq2 => ", seq1+seq2)
```

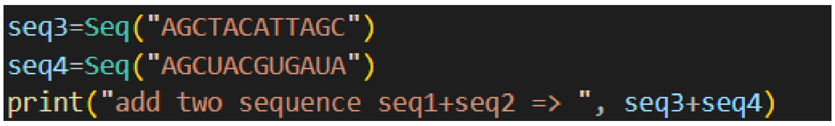
Output:



```
add two sequence seq1+seq2 =>  AGCTTCGA
```

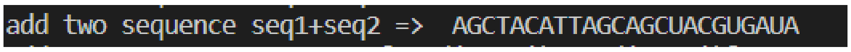
Cascading a protein sequence with a dna sequence:

Biopython 1.81, protein ve dna toplanmasını kontrol etmiyor bu yüzden bunu kullanıcının kendisi kontrol etmeli.



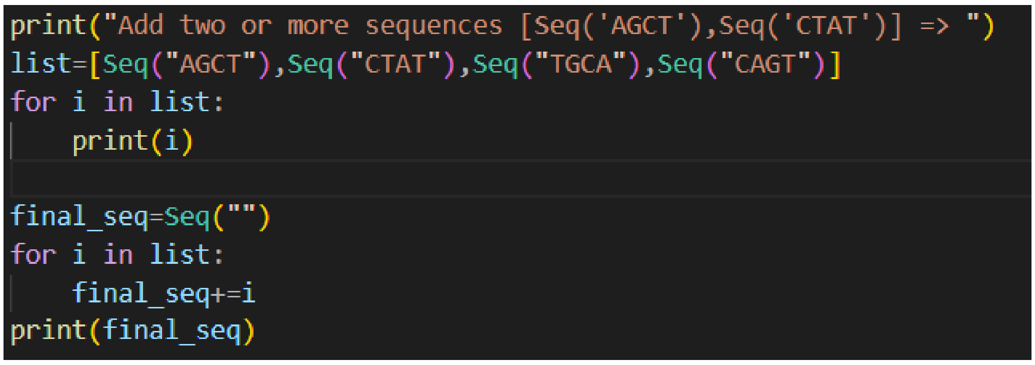
```
seq3=Seq("AGCTACATTAGC")
seq4=Seq("AGCUACGUGAUA")
print("add two sequence seq1+seq2 => ", seq3+seq4)
```

Output:



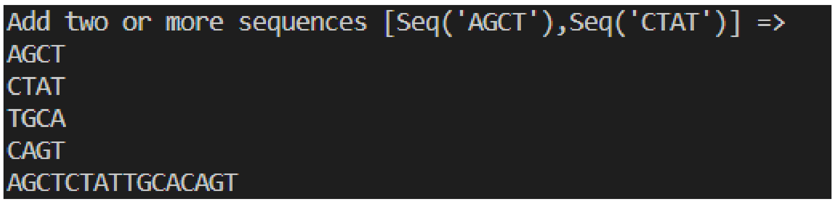
```
add two sequence seq1+seq2 =>  AGCTACATTAGCAGCUACGUGAUA
```

Add two or more sequences:



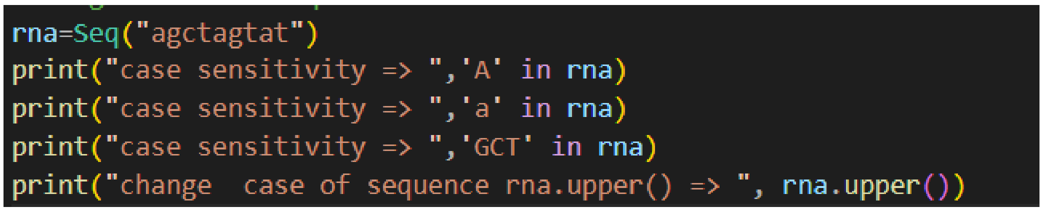
```
print("Add two or more sequences [Seq('AGCT'),Seq('CTAT')] => ")
list=[Seq("AGCT"),Seq("CTAT"),Seq("TGCA"),Seq("CAGT")]
for i in list:
    print(i)

final_seq=Seq("")
for i in list:
    final_seq+=i
print(final_seq)
```

Output:

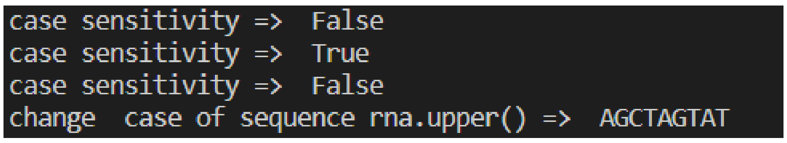


```
Add two or more sequences [Seq('AGCT'),Seq('CTAT')] =>
AGCT
CTAT
TGCA
CAGT
AGCTCTATTGCACAGT
```

To change the case of sequence:

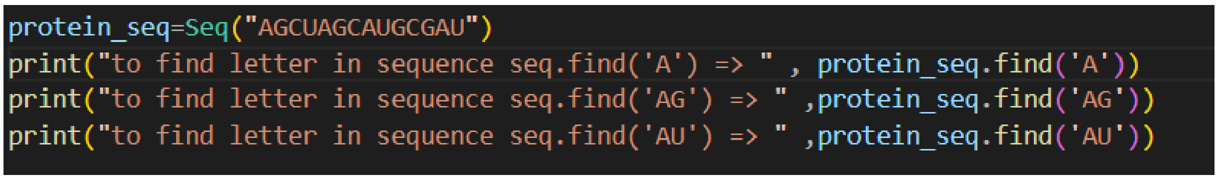


```
rna=Seq("agctagtat")
print("case sensitivity => ",'A' in rna)
print("case sensitivity => ",'a' in rna)
print("case sensitivity => ",'GCT' in rna)
print("change  case of sequence rna.upper() => ", rna.upper())
```



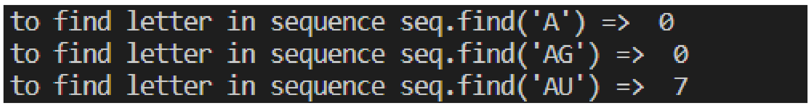
```
case sensitivity =>  False
case sensitivity =>  True
case sensitivity =>  False
change  case of sequence rna.upper() =>  AGCTAGTAT
```

To find letter in sequence:



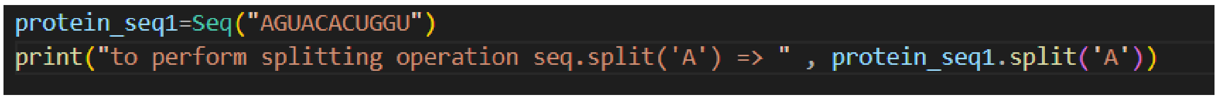
```
protein_seq=Seq("AGCUAGCAUGCGAU")
print("to find letter in sequence seq.find('A') => " , protein_seq.find('A'))
print("to find letter in sequence seq.find('AG') => " ,protein_seq.find('AG'))
print("to find letter in sequence seq.find('AU') => " ,protein_seq.find('AU'))
```

Output:



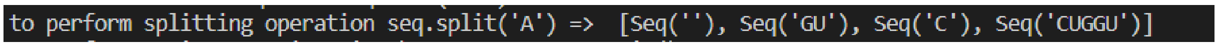
```
to find letter in sequence seq.find('A') =>  0
to find letter in sequence seq.find('AG') =>  0
to find letter in sequence seq.find('AU') =>  7
```

Split operation:



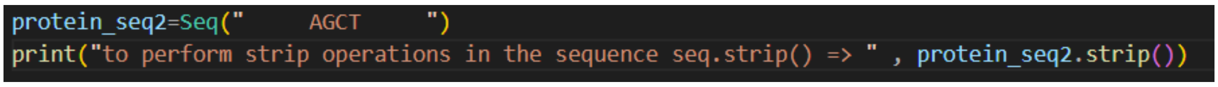
```
protein_seq1=Seq("AGUACACUGGU")
print("to perform splitting operation seq.split('A') => " , protein_seq1.split('A'))
```

Output:



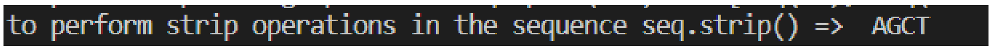
```
to perform splitting operation seq.split('A') =>  [Seq(''), Seq('GU'), Seq('C'), Seq('CUGGU')]
```

Strip operations in sequence:



```
protein_seq2=Seq("     AGCT      ")
print("to perform strip operations in the sequence seq.strip() => " , protein_seq2.strip())
```

Output:



```
to perform strip operations in the sequence seq.strip() =>  AGCT
```