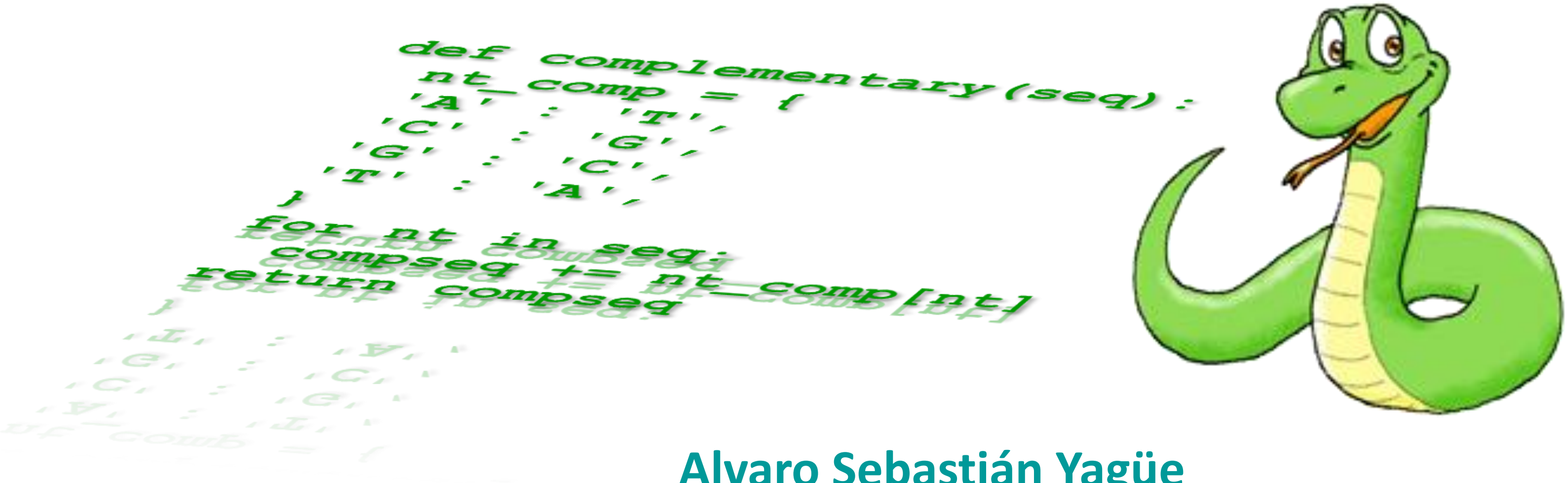
Python for scientists

Lesson 10

Biopython



```
def complementary(seq):
    nt_comp = {
        'A' : 'T',
        'C' : 'G',
        'G' : 'C',
        'T' : 'A',
    }
    for nt in seq:
        compseq += nt_comp[nt]
    return compseq
```

Alvaro Sebastián Yagüe

www.sixthresearcher.com

@SixthResearcher

Biopython

Biopython

Biopython is a Python package that include tools for working with biological data: sequences, structures, databases, population genetics, phylogenetics, sequence motifs and machine learning.

Using Biopython we can save time and effort in writing code for biological data analysis.

Biopython is included in Anaconda distribution as an optional package that we should install before using.

The most important modules included in Biopython are:

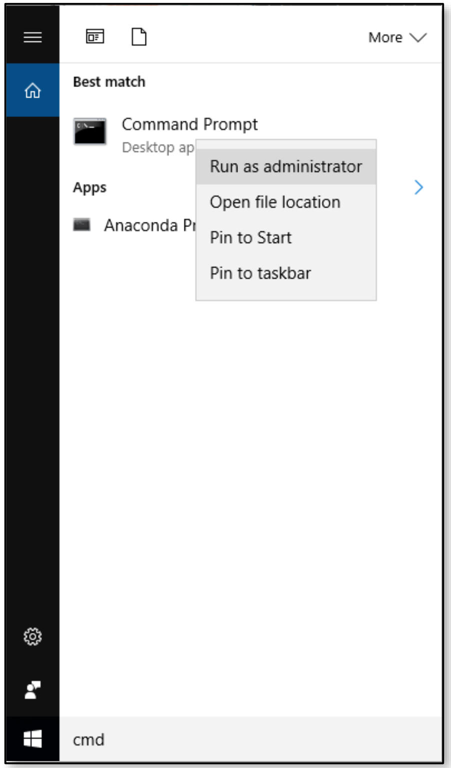
- **Seq and SeqIO**: to read and write sequences in FASTA, FASTQ, GenBank and other formats.
- **Entrez**: to download biological data from NCBI databases.
- **Phylo**: to work with and visualise phylogenetic trees.
- **PDB**: to read and process molecular structures from PDB and mmCIF files.
- **PopGen**: for statistical analysis of population genetics

http://www.biopython.org

Biopython - install

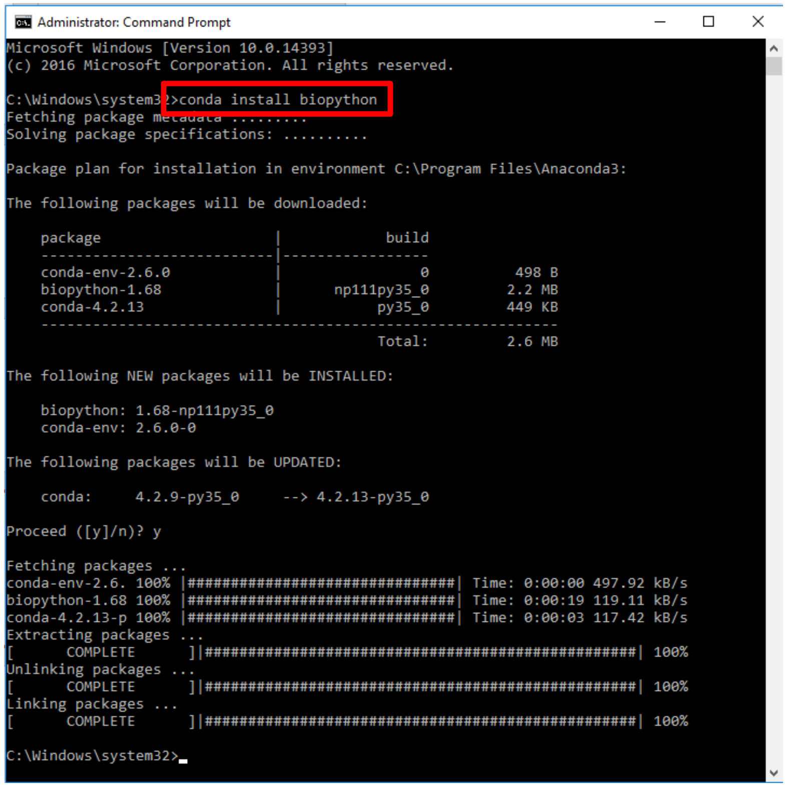
Biopython is included in Anaconda distribution but we should activate the package.

First, **execute the Windows Command Prompt as Administrator**:



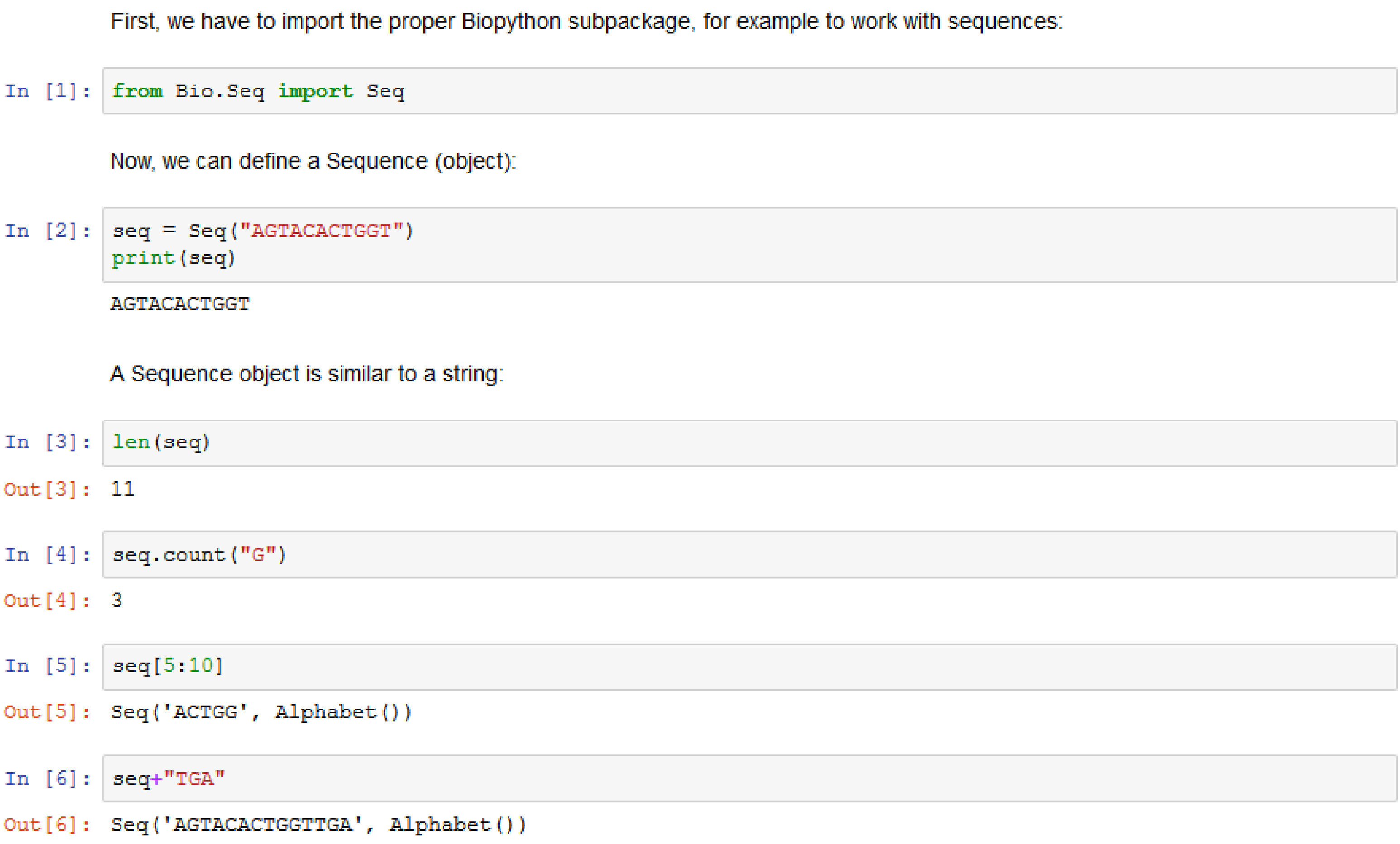
Second**, run the following command:**

```
> conda install biopython
```



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>conda install biopython
Fetching package metadata .........
Solving package specifications: ..........

Package plan for installation in environment C:\Program Files\Anaconda3:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-env-2.6.0            |                0         498 B
    biopython-1.68             |      np111py35_0         2.2 MB
    conda-4.2.13               |           py35_0         449 KB
    ------------------------------------------------------------
                                           Total:         2.6 MB

The following NEW packages will be INSTALLED:

    biopython: 1.68-np111py35_0
    conda-env: 2.6.0-0

The following packages will be UPDATED:

    conda:     4.2.9-py35_0     --> 4.2.13-py35_0

Proceed ([y]/n)? y

Fetching packages ...
conda-env-2.6. 100% |###############################| Time: 0:00:00 497.92 kB/s
biopython-1.68 100% |###############################| Time: 0:00:19 119.11 kB/s
conda-4.2.13-p 100% |###############################| Time: 0:00:03 117.42 kB/s
Extracting packages ...
[      COMPLETE      ]|##################################################| 100%
Unlinking packages ...
[      COMPLETE      ]|##################################################| 100%
Linking packages ...
[      COMPLETE      ]|##################################################| 100%

C:\Windows\system32>_
```

Biopython - sequences



First, we have to import the proper Biopython subpackage, for example to work with sequences:

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

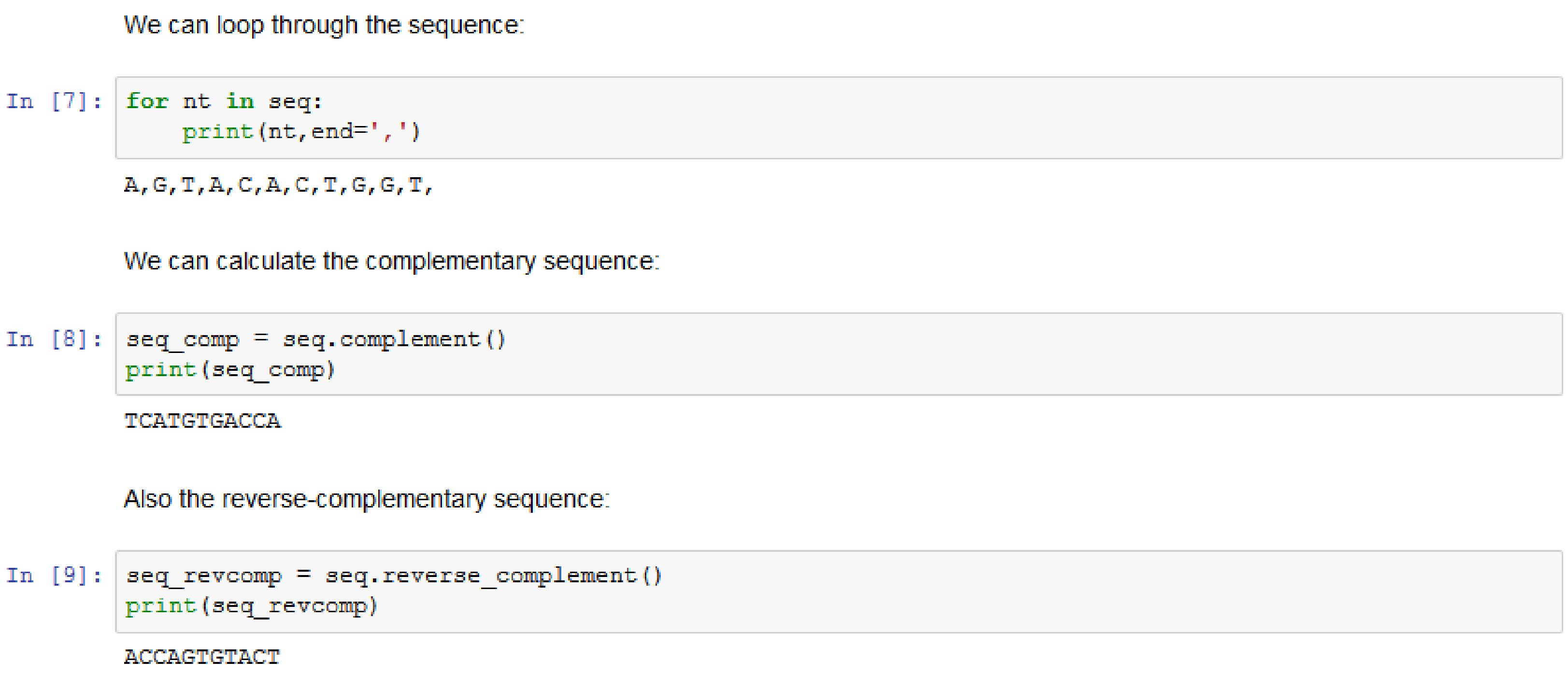
Now, we can define a Sequence (object):

```
In [2]: seq = Seq("AGTACACTGGT")
        print(seq)

AGTACACTGGT
```

A Sequence object is similar to a string:

```
In [3]: len(seq)

Out[3]: 11

In [4]: seq.count("G")

Out[4]: 3

In [5]: seq[5:10]

Out[5]: Seq('ACTGG', Alphabet())

In [6]: seq+"TGA"

Out[6]: Seq('AGTACACTGGTTGA', Alphabet())
```

Biopython - sequences



We can loop through the sequence:

```
In [7]: for nt in seq:
            print(nt,end=',')
```

```
A,G,T,A,C,A,C,T,G,G,T,
```

We can calculate the complementary sequence:

```
In [8]: seq_comp = seq.complement()
        print(seq_comp)
```

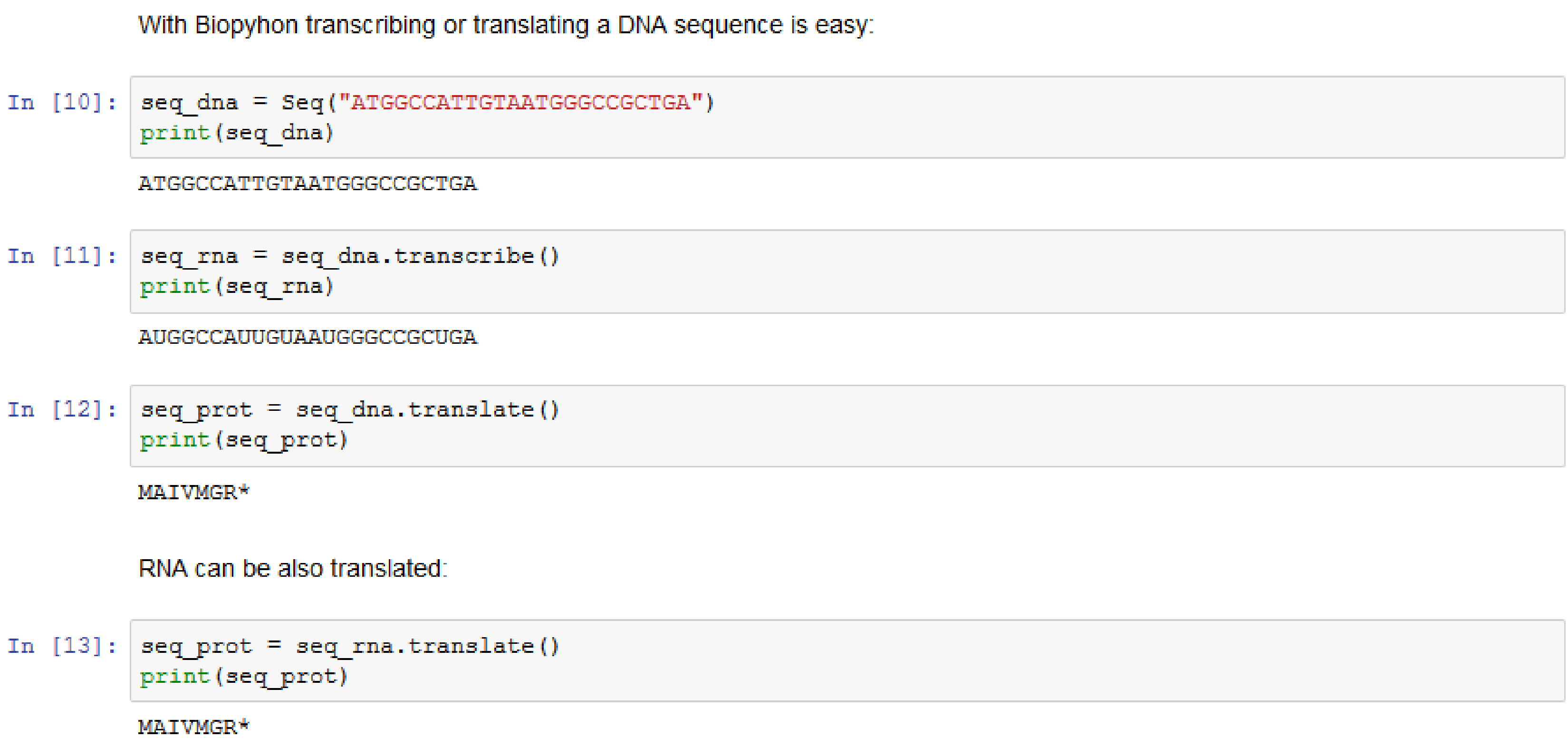
```
TCATGTGACCA
```

Also the reverse-complementary sequence:

```
In [9]: seq_revcomp = seq.reverse_complement()
        print(seq_revcomp)
```

```
ACCAGTGTACT
```

Biopython - sequences



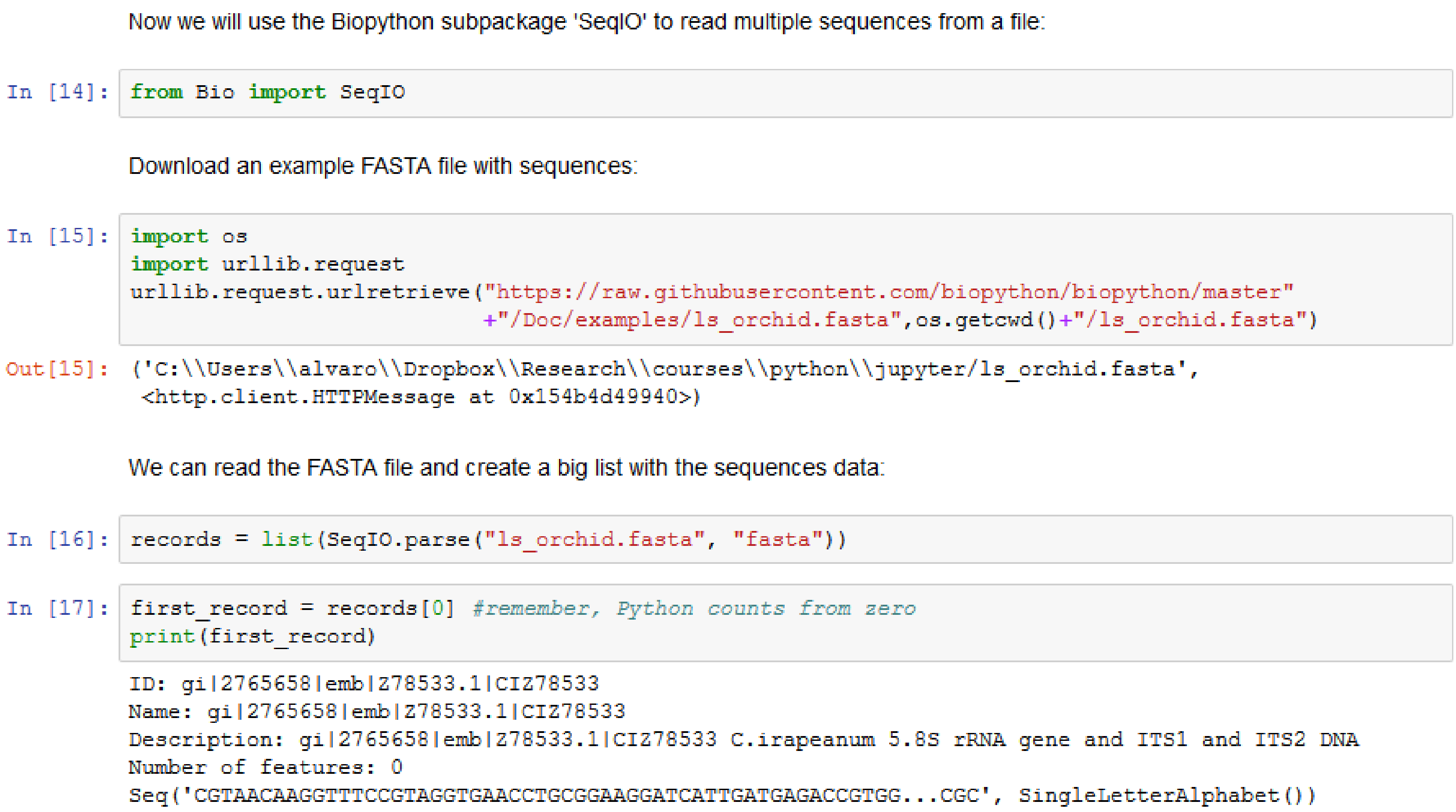
With Biopyhon transcribing or translating a DNA sequence is easy:

```
In [10]: seq_dna = Seq("ATGGCCATTGTAATGGGCCGCTGA")
         print(seq_dna)

ATGGCCATTGTAATGGGCCGCTGA

In [11]: seq_rna = seq_dna.transcribe()
         print(seq_rna)

AUGGCCAUUGUAAUGGGCCGCUGA

In [12]: seq_prot = seq_dna.translate()
         print(seq_prot)

MAIVMGR*
```

RNA can be also translated:

```
In [13]: seq_prot = seq_rna.translate()
         print(seq_prot)

MAIVMGR*
```

Biopython - sequences



Now we will use the Biopython subpackage 'SeqIO' to read multiple sequences from a file:

```
In [14]: from Bio import SeqIO
```

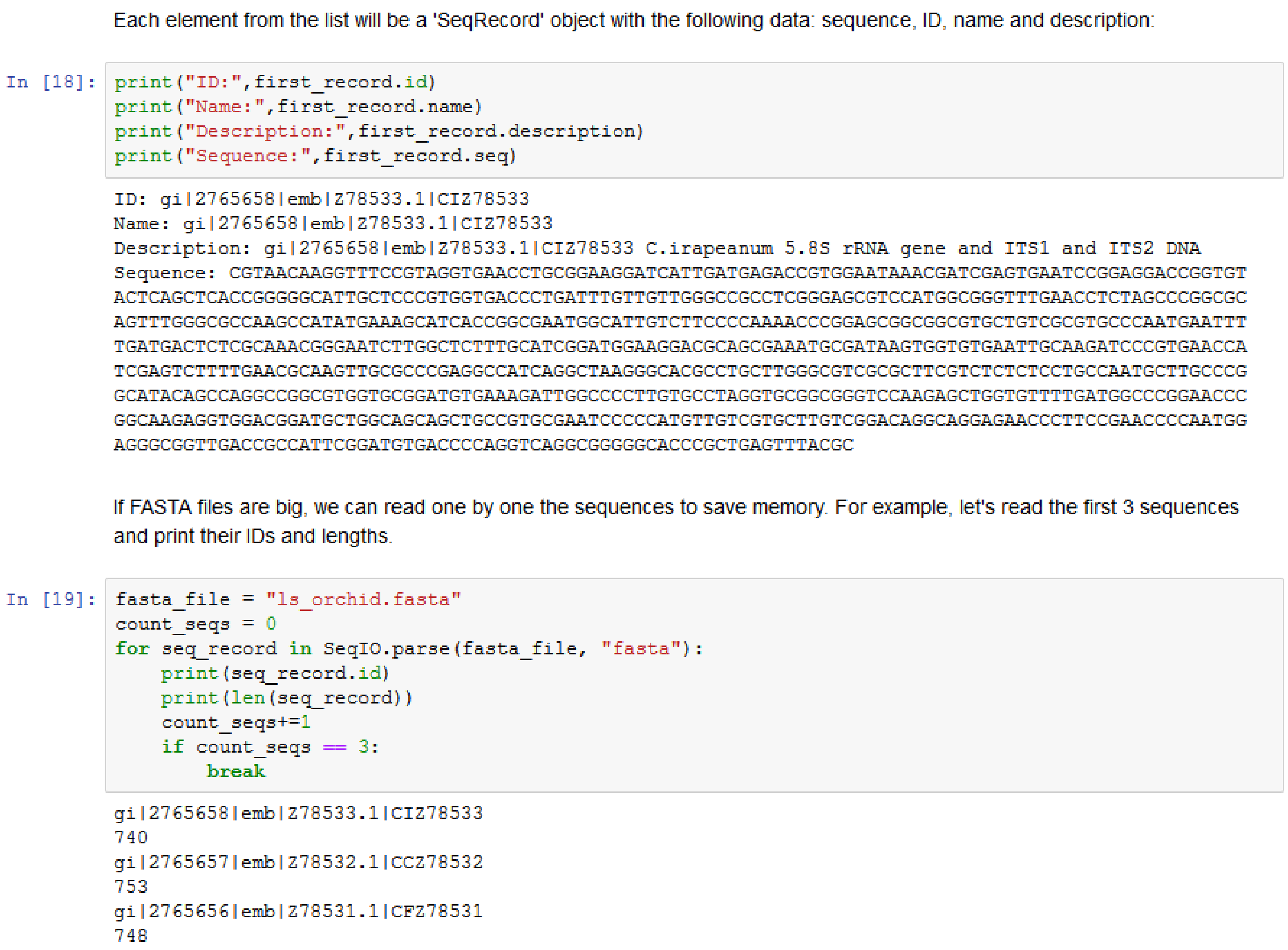
Download an example FASTA file with sequences:

```
In [15]: import os
         import urllib.request
         urllib.request.urlretrieve("https://raw.githubusercontent.com/biopython/biopython/master"
                                    +"/Doc/examples/ls_orchid.fasta",os.getcwd()+"/ls_orchid.fasta")

Out[15]: ('C:\\Users\\alvaro\\Dropbox\\Research\\courses\\python\\jupyter/ls_orchid.fasta',
          <http.client.HTTPMessage at 0x154b4d49940>)
```

We can read the FASTA file and create a big list with the sequences data:

```
In [16]: records = list(SeqIO.parse("ls_orchid.fasta", "fasta"))

In [17]: first_record = records[0] #remember, Python counts from zero
         print(first_record)

         ID: gi|2765658|emb|Z78533.1|CIZ78533
         Name: gi|2765658|emb|Z78533.1|CIZ78533
         Description: gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DNA
         Number of features: 0
         Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC', SingleLetterAlphabet())
```

Biopython - sequences



Each element from the list will be a 'SeqRecord' object with the following data: sequence, ID, name and description:

In [18]:
```
print("ID:",first_record.id)
print("Name:",first_record.name)
print("Description:",first_record.description)
print("Sequence:",first_record.seq)
```

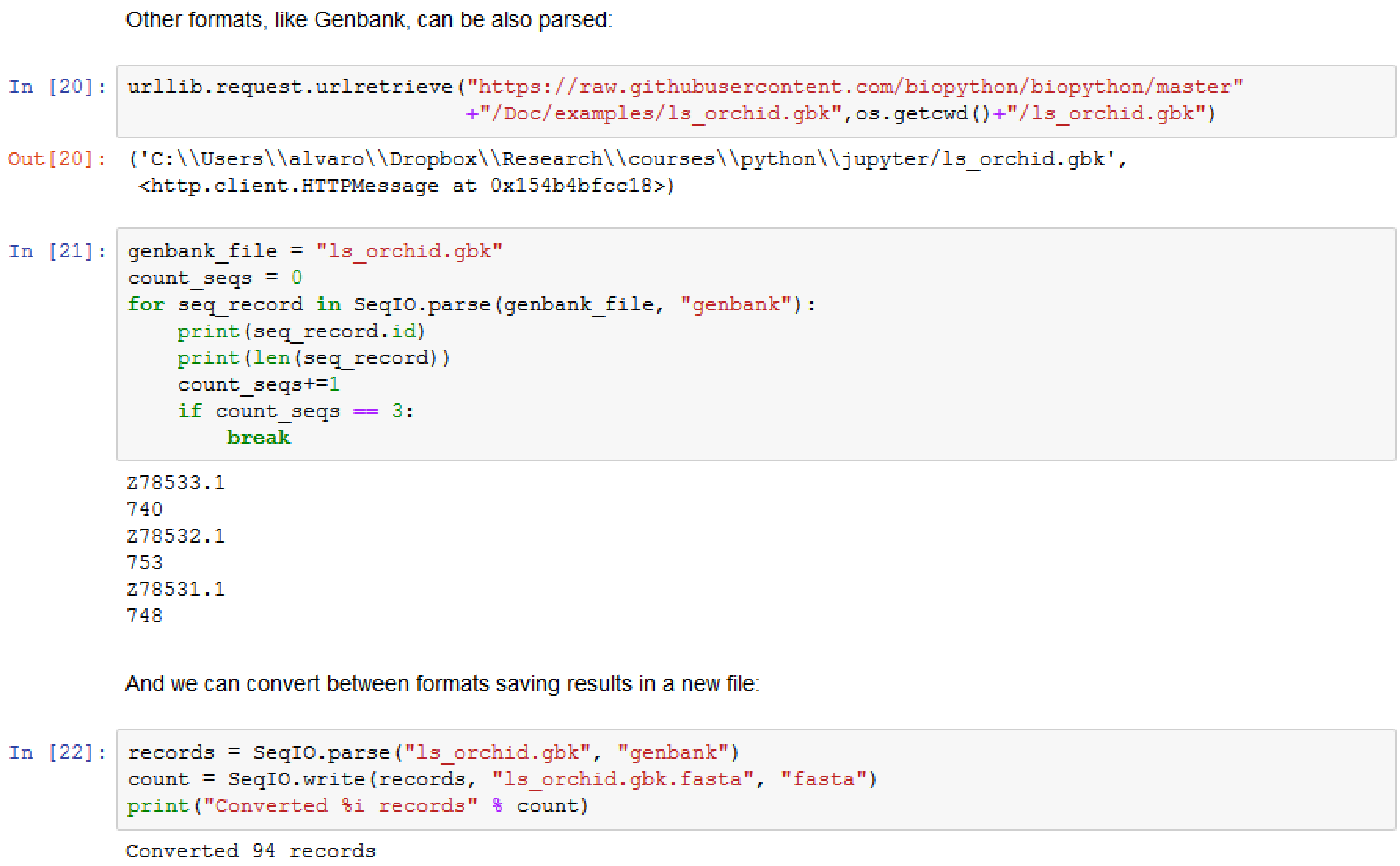
```
ID: gi|2765658|emb|Z78533.1|CIZ78533
Name: gi|2765658|emb|Z78533.1|CIZ78533
Description: gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DNA
Sequence: CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAGTGAATCCGGAGGACCGGTGT
ACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTTGGGCCGCCTCGGGAGCGTCCATGGCGGGTTTGAACCTCTAGCCCGGCGC
AGTTTGGGCGCCAAGCCATATGAAAGCATCACCGGCGAATGGCATTGTCTTCCCCAAAACCCGGAGCGGCGGCGTGCTGTCGCGTGCCCAATGAATTT
TGATGACTCTCGCAAACGGGAATCTTGGCTCTTTGCATCGGATGGAAGGACGCAGCGAAATGCGATAAGTGGTGTGAATTGCAAGATCCCGTGAACCA
TCGAGTCTTTTGAACGCAAGTTGCGCCCGAGGCCATCAGGCTAAGGGCACGCCTGCTTGGGCGTCGCGCTTCGTCTCTCTCCTGCCAATGCTTGCCCG
GCATACAGCCAGGCCGGCGTGGTGCGGATGTGAAAGATTGGCCCCTTGTGCCTAGGTGCGGCGGGTCCAAGAGCTGGTGTTTTGATGGCCCGGAACCC
GGCAAGAGGTGGACGGATGCTGGCAGCAGCTGCCGTGCGAATCCCCCATGTTGTCGTGCTTGTCGGACAGGCAGGAGAACCCTTCCGAACCCCAATGG
AGGGCGGTTGACCGCCATTCGGATGTGACCCCAGGTCAGGCGGGGGCACCCGCTGAGTTTACGC
```

If FASTA files are big, we can read one by one the sequences to save memory. For example, let's read the first 3 sequences and print their IDs and lengths.

In [19]:
```
fasta_file = "ls_orchid.fasta"
count_seqs = 0
for seq_record in SeqIO.parse(fasta_file, "fasta"):
    print(seq_record.id)
    print(len(seq_record))
    count_seqs+=1
    if count_seqs == 3:
        break
```

```
gi|2765658|emb|Z78533.1|CIZ78533
740
gi|2765657|emb|Z78532.1|CCZ78532
753
gi|2765656|emb|Z78531.1|CFZ78531
748
```

Biopython - sequences



Other formats, like Genbank, can be also parsed:

```
In [20]: urllib.request.urlretrieve("https://raw.githubusercontent.com/biopython/biopython/master"
                                    +"/Doc/examples/ls_orchid.gbk",os.getcwd()+"/ls_orchid.gbk")
```

```
Out[20]: ('C:\\Users\\alvaro\\Dropbox\\Research\\courses\\python\\jupyter/ls_orchid.gbk',
          <http.client.HTTPMessage at 0x154b4bfcc18>)
```

```
In [21]: genbank_file = "ls_orchid.gbk"
         count_seqs = 0
         for seq_record in SeqIO.parse(genbank_file, "genbank"):
             print(seq_record.id)
             print(len(seq_record))
             count_seqs+=1
             if count_seqs == 3:
                 break
```

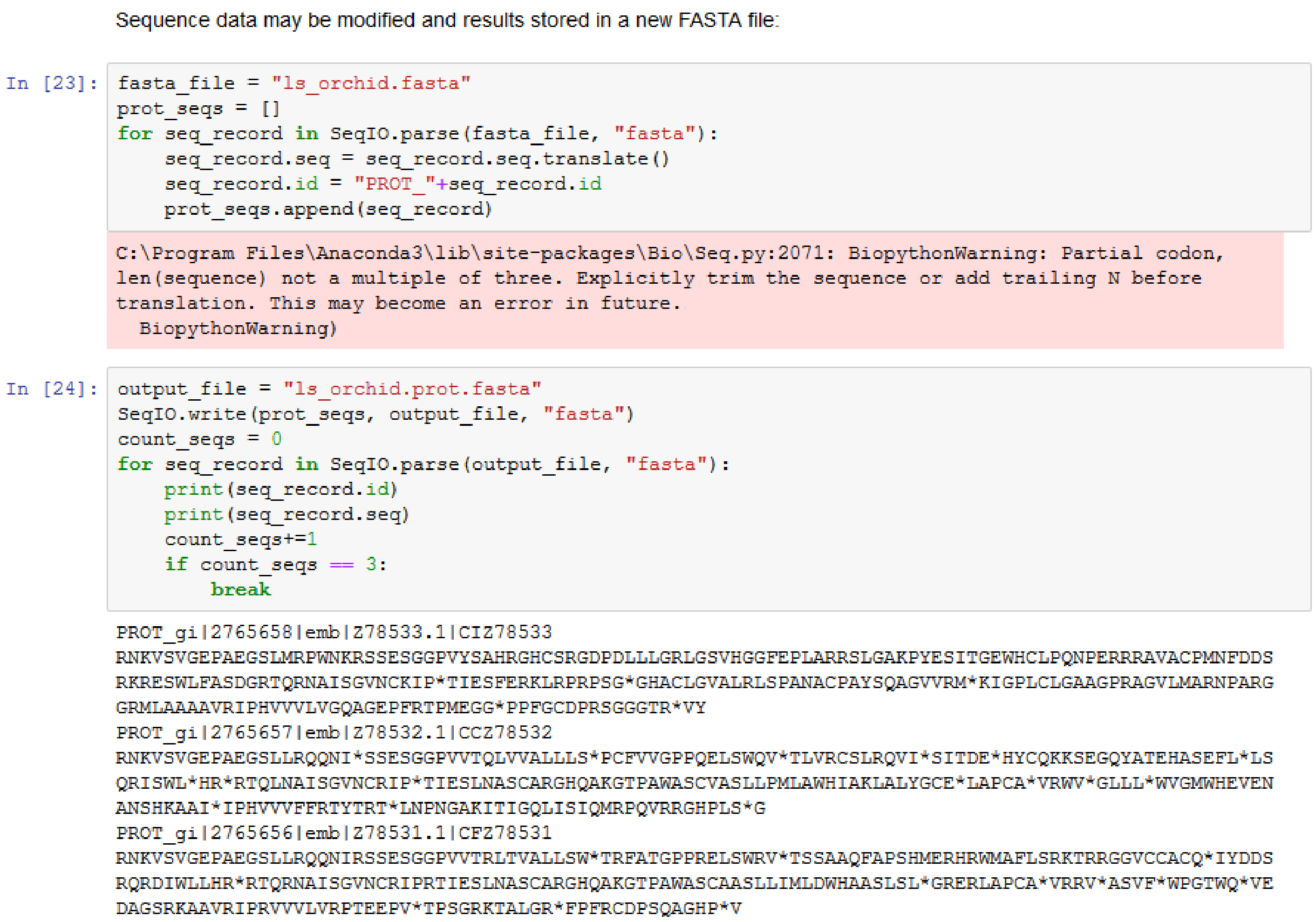
```
Z78533.1
740
Z78532.1
753
Z78531.1
748
```

And we can convert between formats saving results in a new file:

```
In [22]: records = SeqIO.parse("ls_orchid.gbk", "genbank")
         count = SeqIO.write(records, "ls_orchid.gbk.fasta", "fasta")
         print("Converted %i records" % count)
```

```
Converted 94 records
```

Biopython - sequences



Sequence data may be modified and results stored in a new FASTA file:

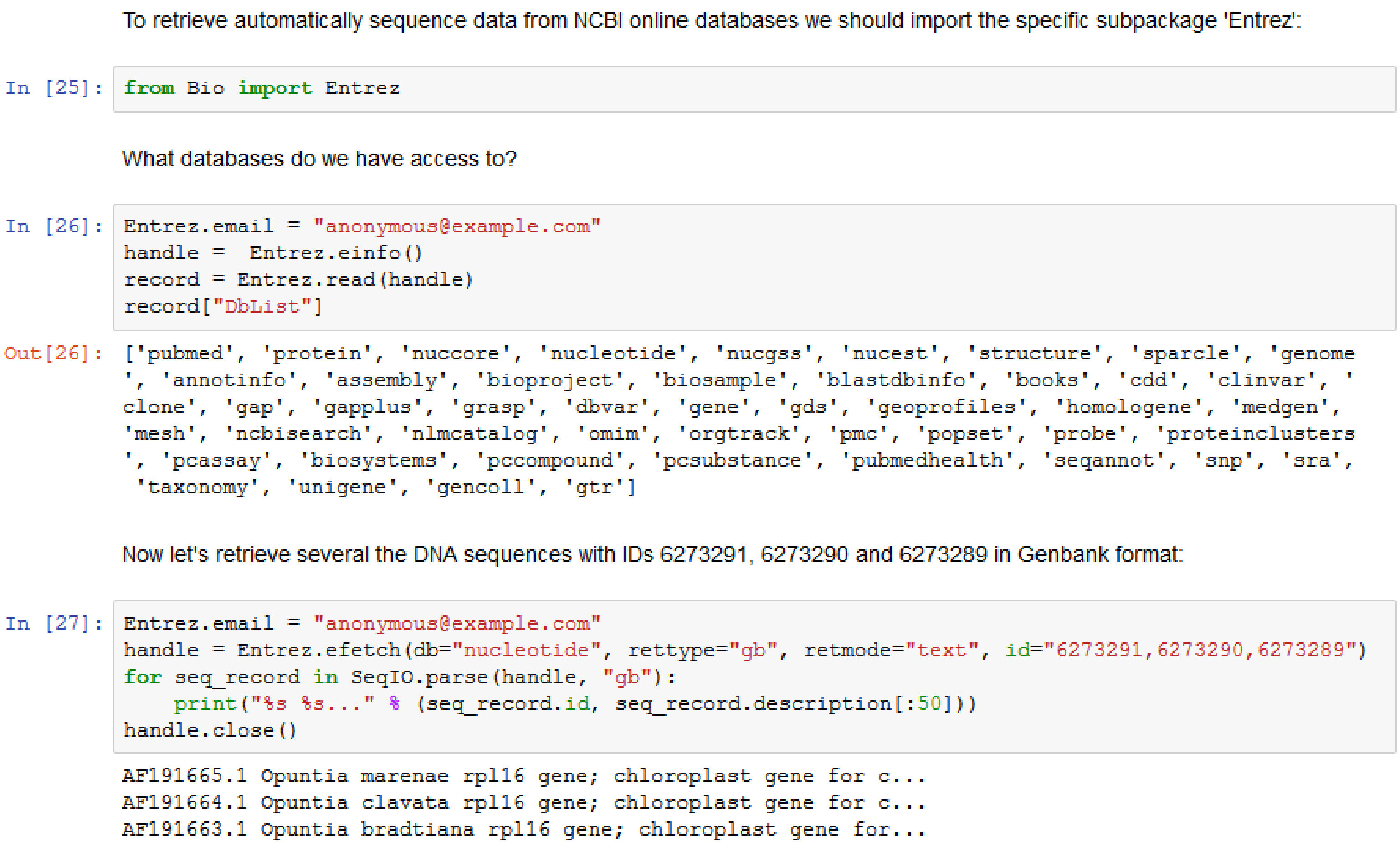
In [23]:
```
fasta_file = "ls_orchid.fasta"
prot_seqs = []
for seq_record in SeqIO.parse(fasta_file, "fasta"):
    seq_record.seq = seq_record.seq.translate()
    seq_record.id = "PROT_"+seq_record.id
    prot_seqs.append(seq_record)
```

```
C:\Program Files\Anaconda3\lib\site-packages\Bio\Seq.py:2071: BiopythonWarning: Partial codon,
len(sequence) not a multiple of three. Explicitly trim the sequence or add trailing N before
translation. This may become an error in future.
  BiopythonWarning)
```

In [24]:
```
output_file = "ls_orchid.prot.fasta"
SeqIO.write(prot_seqs, output_file, "fasta")
count_seqs = 0
for seq_record in SeqIO.parse(output_file, "fasta"):
    print(seq_record.id)
    print(seq_record.seq)
    count_seqs+=1
    if count_seqs == 3:
        break
```

```
PROT_gi|2765658|emb|Z78533.1|CIZ78533
RNKVSVGEPAEGSLMRPWNKRSSESGGPVYSAHRGHCSRGDPDLLLGRLGSVHGGFEPLARRSLGAKPYESITGEWHCLPQNPERRRAVACPMNFDDS
RKRESWLFASDGRTQRNAISGVNCKIP*TIESFERKLRPRPSG*GHACLGVALRLSPANACPAYSQAGVVRM*KIGPLCLGAAGPRAGVLMARNPARG
GRMLAAAAVRIPHVVVLVGQAGEPFRTPMEGG*PPFGCDPRSGGGTR*VY
PROT_gi|2765657|emb|Z78532.1|CCZ78532
RNKVSVGEPAEGSLLRQQNI*SSESGGPVVTQLVVALLLS*PCFVVGPPQELSWQV*TLVRCSLRQVI*SITDE*HYCQKKSEGQYATEHASEFL*LS
QRISWL*HR*RTQLNAISGVNCRIP*TIESLNASCARGHQAKGTPAWASCVASLLPMLAWHIAKLALYGCE*LAPCA*VRWV*GLLL*WVGMWHEVEN
ANSHKAAI*IPHVVVFFRTYTRT*LNPNGAKITIGQLISIQMRPQVRRGHPLS*G
PROT_gi|2765656|emb|Z78531.1|CFZ78531
RNKVSVGEPAEGSLLRQQNIRSSESGGPVVTRLTVALLSW*TRFATGPPRELSWRV*TSSAAQFAPSHMERHRWMAFLSRKTRRGGVCCACQ*IYDDS
RQRDIWLLHR*RTQRNAISGVNCRIPRTIESLNASCARGHQAKGTPAWASCAASLLIMLDWHAASLSL*GRERLAPCA*VRRV*ASVF*WPGTWQ*VE
DAGSRKAAVRIPRVVVLVRPTEEPV*TPSGRKTALGR*FPFRCDPSQAGHP*V
```

Biopython - databases



To retrieve automatically sequence data from NCBI online databases we should import the specific subpackage 'Entrez':

In [25]:

```python
from Bio import Entrez
```

What databases do we have access to?

In [26]:

```python
Entrez.email = "anonymous@example.com"
handle =  Entrez.einfo()
record = Entrez.read(handle)
record["DbList"]
```

Out[26]:

```
['pubmed', 'protein', 'nuccore', 'nucleotide', 'nucgss', 'nucest', 'structure', 'sparcle', 'genome
', 'annotinfo', 'assembly', 'bioproject', 'biosample', 'blastdbinfo', 'books', 'cdd', 'clinvar', '
clone', 'gap', 'gapplus', 'grasp', 'dbvar', 'gene', 'gds', 'geoprofiles', 'homologene', 'medgen',
'mesh', 'ncbisearch', 'nlmcatalog', 'omim', 'orgtrack', 'pmc', 'popset', 'probe', 'proteinclusters
', 'pcassay', 'biosystems', 'pccompound', 'pcsubstance', 'pubmedhealth', 'seqannot', 'snp', 'sra',
 'taxonomy', 'unigene', 'gencoll', 'gtr']
```

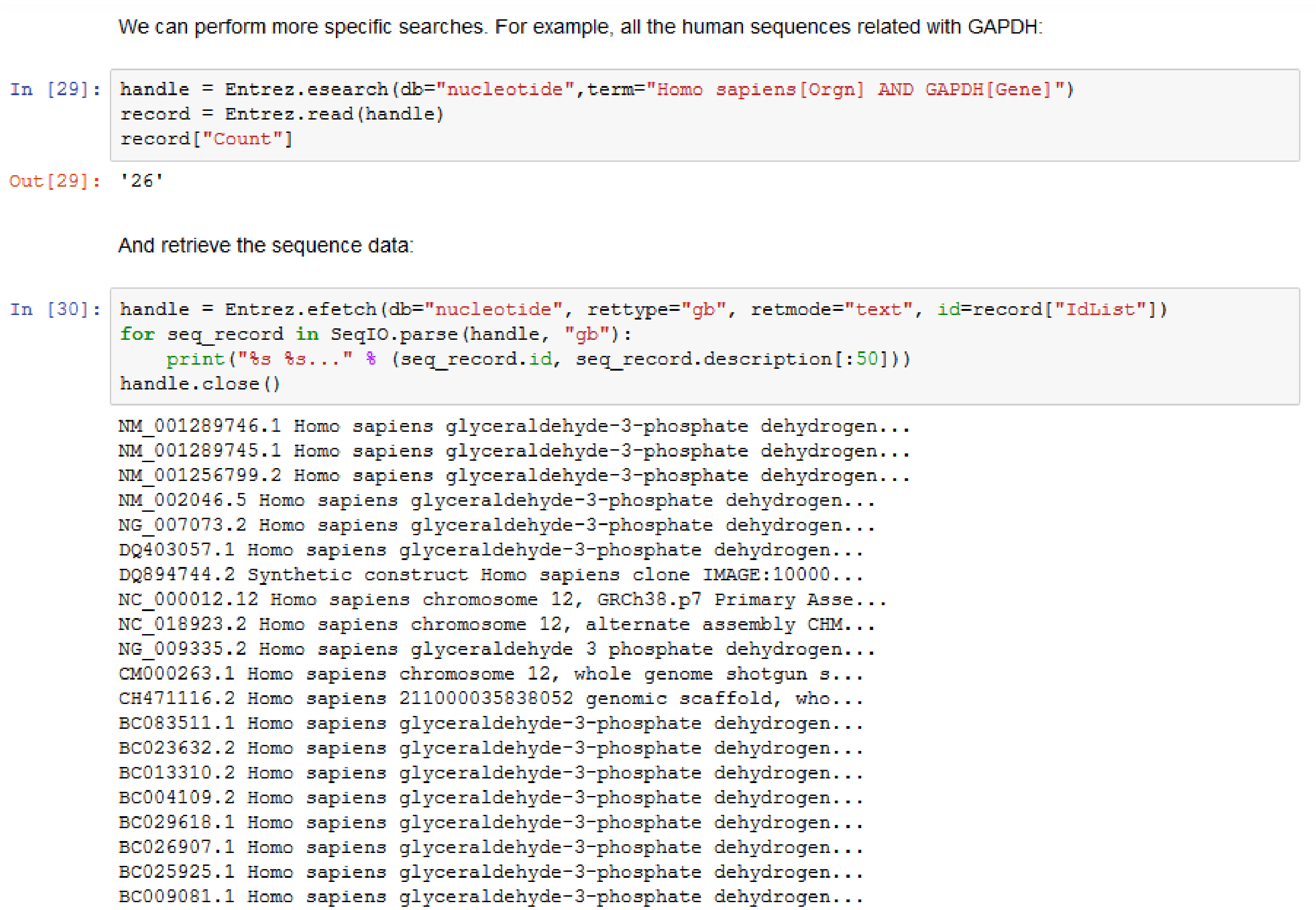
Now let's retrieve several the DNA sequences with IDs 6273291, 6273290 and 6273289 in Genbank format:

In [27]:

```python
Entrez.email = "anonymous@example.com"
handle = Entrez.efetch(db="nucleotide", rettype="gb", retmode="text", id="6273291,6273290,6273289")
for seq_record in SeqIO.parse(handle, "gb"):
    print("%s %s..." % (seq_record.id, seq_record.description[:50]))
handle.close()
```

```
AF191665.1 Opuntia marenae rpl16 gene; chloroplast gene for c...
AF191664.1 Opuntia clavata rpl16 gene; chloroplast gene for c...
AF191663.1 Opuntia bradtiana rpl16 gene; chloroplast gene for...
```

Biopython - databases



We can perform more specific searches. For example, all the human sequences related with GAPDH:

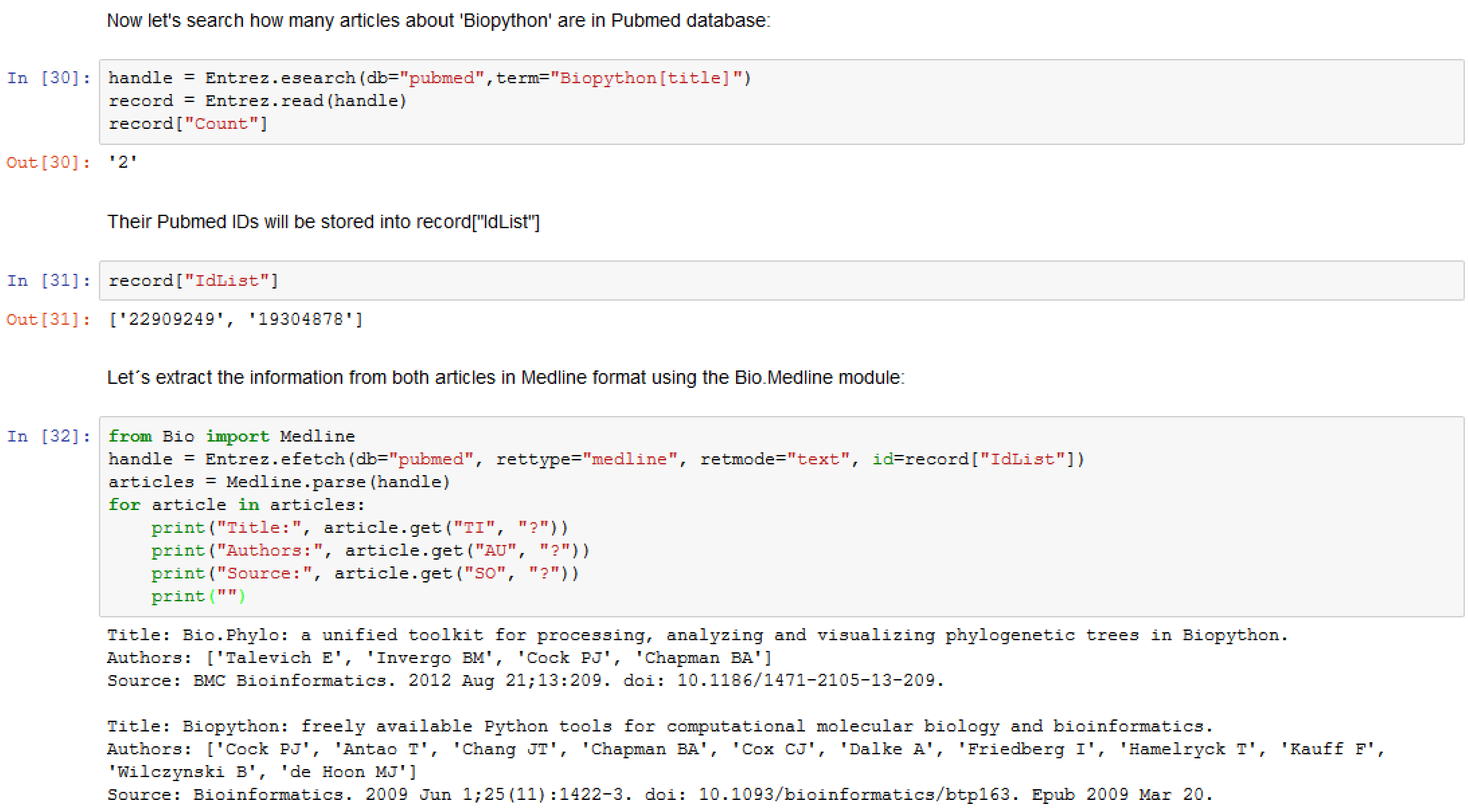
```
In [29]: handle = Entrez.esearch(db="nucleotide",term="Homo sapiens[Orgn] AND GAPDH[Gene]")
         record = Entrez.read(handle)
         record["Count"]

Out[29]: '26'
```

And retrieve the sequence data:

```
In [30]: handle = Entrez.efetch(db="nucleotide", rettype="gb", retmode="text", id=record["IdList"])
         for seq_record in SeqIO.parse(handle, "gb"):
             print("%s %s..." % (seq_record.id, seq_record.description[:50]))
         handle.close()
```

```
NM_001289746.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
NM_001289745.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
NM_001256799.2 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
NM_002046.5 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
NG_007073.2 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
DQ403057.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
DQ894744.2 Synthetic construct Homo sapiens clone IMAGE:10000...
NC_000012.12 Homo sapiens chromosome 12, GRCh38.p7 Primary Asse...
NC_018923.2 Homo sapiens chromosome 12, alternate assembly CHM...
NG_009335.2 Homo sapiens glyceraldehyde 3 phosphate dehydrogen...
CM000263.1 Homo sapiens chromosome 12, whole genome shotgun s...
CH471116.2 Homo sapiens 211000035838052 genomic scaffold, who...
BC083511.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
BC023632.2 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
BC013310.2 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
BC004109.2 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
BC029618.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
BC026907.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
BC025925.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
BC009081.1 Homo sapiens glyceraldehyde-3-phosphate dehydrogen...
```

Biopython - databases



Now let's search how many articles about 'Biopython' are in Pubmed database:

```
In [30]: handle = Entrez.esearch(db="pubmed",term="Biopython[title]")
         record = Entrez.read(handle)
         record["Count"]
```

```
Out[30]: '2'
```

Their Pubmed IDs will be stored into record["IdList"]

```
In [31]: record["IdList"]
```

```
Out[31]: ['22909249', '19304878']
```

Let´s extract the information from both articles in Medline format using the Bio.Medline module:

```
In [32]: from Bio import Medline
         handle = Entrez.efetch(db="pubmed", rettype="medline", retmode="text", id=record["IdList"])
         articles = Medline.parse(handle)
         for article in articles:
             print("Title:", article.get("TI", "?"))
             print("Authors:", article.get("AU", "?"))
             print("Source:", article.get("SO", "?"))
             print("")
```

```
Title: Bio.Phylo: a unified toolkit for processing, analyzing and visualizing phylogenetic trees in Biopython.
Authors: ['Talevich E', 'Invergo BM', 'Cock PJ', 'Chapman BA']
Source: BMC Bioinformatics. 2012 Aug 21;13:209. doi: 10.1186/1471-2105-13-209.

Title: Biopython: freely available Python tools for computational molecular biology and bioinformatics.
Authors: ['Cock PJ', 'Antao T', 'Chang JT', 'Chapman BA', 'Cox CJ', 'Dalke A', 'Friedberg I', 'Hamelryck T', 'Kauff F',
'Wilczynski B', 'de Hoon MJ']
Source: Bioinformatics. 2009 Jun 1;25(11):1422-3. doi: 10.1093/bioinformatics/btp163. Epub 2009 Mar 20.
```