Biopython Basics Cheat Sheet

by Taissir Boukrouba (taissir2002) via cheatography.com/193976/cs/42971/

Installing Biopython

```
pip install biopython
pip install --upgrade biopython
# import the library
import Bio
```

Creating Sequences

```
from Bio.Seq import Seq
my_seq = Seq("AA TGC ACG TTG ")
```

To create a sequence we use the `Seq` function from `Bio` library

Filling Sequences

```
# filling sequences
fragments = [Seq("G TAT "), Seq("TA CT")]
filler = Seq("A"*3)
print( fil ler.jo in( fra gme nts))
#output :
>>> GTATAA ATACT
```

Slicing Sequences

```
# defining sequences
my_seq = Seq("AA GTC CAG TGT ")
my_seq_2 = Seq("AA AA")
# slicing sequences
print( my_ seq [1:6])
print( my_ seq [0::2])
# output :
>>> AGTCC
>>> AGCATT
```

Slicing Sequences is the same as that of a python list (we use `[]`)

Appending Sequences

```
my_seq = Seq("AAGTCCAGTGT")
my_seq_2 = Seq("AA AA")
#appending sequences
print( my_seq + my_seq_2)
# output
```

By **Taissir Boukrouba** (taissir2002)

cheatography.com/taissir2002/

Not published yet.
Last updated 7th April, 2024.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Appending Sequences (cont)

> >>> AAGTCCAGTGTAAAA

Appending sequences is the same as appending strings in python

Sequence Counting

```
from Bio.Seq import Seq
# creating a sequence
seq_ex ample = Seq("AG TAC ACT GGT ")
seq_length = len(se q_e xample)
occ = seq_ex amp le.c ou nt( " C")
print( "The length of the sequence is", len(se q_e xam ple))
print( "The number of occurr ences for nucleotide C is ", occ )
#output :
The length of the sequence is 11
The number of occurr ences for nucleotide C is 2
```

This provides how to get the length of a sequence and the number of occurrences of a specific nucleotide

Finding Sub-sequence Index

```
my_seq = Seq("AAGTCCAGTGT")
index = my_seq.fi nd( " GTC ")
print( f"GTC index is {index }")
# output :
GTC index is 2
```

This returns the start index of the selected sub-sequence

Reading Sequence Files

```
from Bio import SeqIO
records = SeqIO.p ar se( " seq uen ce_ fil e.f ast a", " fas ta")
for record in records :
        pri nt( rec ord.seq)
```

We can also access other attributes from the records :

- `record.seq` : returns one sequence from list of records
- `record.id` : returns the identifier of the sequence
- `record.de scr iption` : returns the sequence description

Writing Sequences into a file

```
from Bio import SeqIO
# Define your sequence as a string
sequence = " ATC GAT CGA TCG ATC GAT C"
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Writing Sequences into a file (cont)

```
> # Defining file name and format
filename = "my_sequence.fasta"
format = "fasta"
# defining the sequence
seq = SeqIO.SeqRecord(SeqIO.Seq(sequence1),
                id="my_id", description="My sequence description")
# Open the file for writing in text mode
with open(filename, "w") as file:
  # Create a SeqRecord object
  record = SeqIO.SeqRecord(seq)
  # Write the record to the file using the specified format
  SeqIO.write(record, file, format)
```

Converting Files

```
# syntax
SeqIO.c on ver t(i np_ file, inp_fo rmat, outp_file, outp_f ormat, alphab et= None)
#example
SeqIO.c on ver t("s equ enc e.g bk", " gen ban k", " seq uen ce_ con ver ted.fa sta ", " fas ta")
```

inp_file : path to input file
inp_format : input file format/extention
outp_file : path to output file
outp_format : output file format/extention
alphabet : specify the correct alphabet (DNA,RNA or Protein) to avoid conversion confusion

Sequence Molecular Weight

```
from Bio.SeqUtils import molecular_weight
from Bio.Seq import Seq
seq_ex ample = Seq("TG TAC CCT GGT ")
mw = molecu lar _we igh t(s eq_ exa mple)
print(mw)
#output :
>>> 3403.1577
```

Molecular weight is a way to guess how heavy a tiny building block of life (like a protein or piece of DNA) is compared to a single carbon atom where the bigger the building blocks , the higher the molecular weight

GC-Content

```
from Bio.SeqUtils import gc_fraction
from Bio.Seq import Seq
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

GC-Content (cont)

```
> # creating a sequence
seq_example = Seq("AGATTCACTGGT")
gc_content = gc_fraction(seq_example)
print(gc_content)
# output :
>>> 0.41
```

G-C content refers to the percentage of **guanine (G)** and **cytosine (C)** molecules out of all the building blocks (called nucleotides) in a strand of DNA or RNA.

Reverse Complement

```
from Bio.Seq import Seq
#creating a sequence
seq_ex ample = Seq("AG TAC ACT GGT ")
print( " Seq uence is :",s eq_ exa mple)
# getting the reverse compliment
rev_comp = seq_ex amp le.r ev ers e_c omp lem ent()
print( " Reverse comple men t:", rev_comp)
#output :
>>> Sequence is : AGTACA CTGGT
>>> The reverse complement : ACCAGT GTACT
```

Reverse complement of a DNA sequence is like a mirror image on the opposite strand.

- **Reverse:** Flips the order of the DNA letters (A, C, G, T) from left to right to right to left.
- **Complement:** Swaps each letter according to its pair: A pairs with T, and C pairs with G.

Transcription & Translation

```
seq_example = Seq("ATGAAGTTTTAG")
transc = seq_ex amp le.t ra nsc ribe()
print( " Tra nsc rip tio n:", transc)
transl = seq_ex amp le.t ra nsl ate()
print( " Tra nsl ati on: ", transl)
#output :
>>> Transc rip tion: AUGAAG UUUUAG
>>> Transl ation: MKF*
```

Transcription and translation are the two main steps that turn the instructions in our genes (DNA) into the building blocks of life (proteins).

- *Transcription* : is going from DNA to RNA (creating a copy)
- *Translation* : is going from RNA to Protein

Accessing NCBI Database using esearch()

```
from Bio import Entrez
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Accessing NCBI Database using esearch() (cont)

```
> handle = Entrez.esearch(db="nucleotide", term="BRCA1 gene", retmax=20)
record = Entrez.read(handle)
```

- `db` : The name of the Entrez database to search ("nucleotide", "protein"....)
- `term` : The search term (e.g., gene name, protein ID ....)
- `retmode (str, optional)` : The format (return mode) to return results in (default: "xml").
- `retmax (int, optional)` : Maximum number of IDs to return (default: 10).
- `sort (str, optional)` : Sorting criteria for results (default: "relevance")

Accessing NCBI Database using efetch()

```
from Bio import Entrez
id_list = ["NM _00 729 4.3 ", " NM_ 000 546.5"]
handle = Entrez.ef etc h(d b="n ucl eot ide ", id=id_ list, rettyp e="g b")
records = Entrez.re ad( handle)
```

`db` : The name of the Entrez database to search ("nucleotide", "protein"....)
`id` (list or str): A single ID or a list of IDs to retrieve
`rettype` (str, optional): The type of information to return (default: "gb" for GenBank format)
`retmode` (str, optional): The format to return results in (default: "xml").

By **Taissir Boukrouba**
(taissir2002)

cheatography.com/taissir2002/

Not published yet.
Last updated 7th April, 2024.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com