

# Bioinformatique

Manipulations de séquences ADN, ARN, protéines,...



: à compléter (différents formats, bases de données ?)

## Compter les nucléotides d'une séquence ADN

```
<sxh python; title : Counting_DNA_Nucleotides-01.py> #!/usr/bin/env python # -*- coding: utf-8 -*-  
""" On dispose d'un exemple de chaîne ADN (constituée des symboles 'A', 'C', 'G', 'T') Le programme  
utilise plusieurs techniques pour donner les nombres d'occurrences respectifs des différentes bases  
""" adn =  
"AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAGAGTGTCTGATAGCAGC"
```

# utilisation d'une liste et de la méthode .count() bases=["A","C","G","T"] for base in bases:

```
print adn.count(base),
```

print

# Variante : for c in 'ACGT':

```
print adn.count(c),
```

print

# variante un peu moins lisible out = [] for c in 'ACGT':

```
out.append(str(adn.count(c)))
```

print(' '.join(out))

# utilisation de la technique "list comprehension" count=[adn.count(c) for c in 'ACGT'] for val in count:

```
print val,
```

print

# autre "list comprehension", avec impression formatée → version "one line" print "%d %d %d %d" % tuple([adn.count(X) for X in "ACGT"])

# count "à la main", sans utilisation de fonctions/librairie ACGT = "ACGT" count = [0,0,0,0] for c in adn:

```
for i in range(len(ACGT)):  
    if c == ACGT[i]:
```

```
count[i] +=1
```

for val in count:

```
print val,
```

print

# count "à la main", avec .index() ACGT = "ACGT" count = [0,0,0,0] for c in adn:

```
count[ACGT.index(c)] += 1
```

for val in count:

```
print val,
```

print

# utilisation de la librairie collections from collections import defaultdict ncount = defaultdict(int) for c in adn:

```
ncount[c] += 1
```

print ncount['A'], ncount['C'], ncount['G'], ncount['T']

# collections.Counter from collections import Counter for k,v in sorted(Counter(adn).items()): print v, print

# avec un dictionnaire freq = {'A': 0, 'C': 0, 'G': 0, 'T': 0} for c in adn:

```
freq[c] += 1
```

print freq['A'], freq['C'], freq['G'], freq['T']

# avec un dictionnaire et count(), impression différente dico={} for base in bases:

```
dico[base]=adn.count(base)
```

for key,val in dico.items():

```
print "{} = {}".format(key, val)
```

</sxh>

## Trouver un motif

+ lecture de fichier

<sxh python; title : Finding\_a\_Protein\_Motif-01.py> #!/usr/bin/env python # -\*- coding: utf-8 -\*- """ La description complète et les caractéristiques d'une protéine particulière peuvent être obtenues via l'ID "uniprot\_id" de la "UniProt database", en insérant la référence dans ce lien :

[http://www.uniprot.org/uniprot/uniprot\\_id](http://www.uniprot.org/uniprot/uniprot_id)

On peut aussi obtenir la séquence peptidique au format FASTA via le lien :

[http://www.uniprot.org/uniprot/uniprot\\_id.fasta](http://www.uniprot.org/uniprot/uniprot_id.fasta) """

```
from Bio import SeqIO from Bio import Expasy from Bio import SeqIO
```

```
dic = {"UUU": "F", "UUC": "F", "UUA": "L", "UUG": "L",
```

```
"UCU": "S", "UCC": "S", "UCA": "S", "UCG": "S",
"UAU": "Y", "UAC": "Y", "UAA": "STOP", "UAG": "STOP",
"UGU": "C", "UGC": "C", "UGA": "STOP", "UGG": "W",
"CUU": "L", "CUC": "L", "CUA": "L", "CUG": "L",
"CCU": "P", "CCC": "P", "CCA": "P", "CCG": "P",
"CAU": "H", "CAC": "H", "CAA": "Q", "CAG": "Q",
"CGU": "R", "CGC": "R", "CGA": "R", "CGG": "R",
"AUU": "I", "AUC": "I", "AUA": "I", "AUG": "M",
"ACU": "T", "ACC": "T", "ACA": "T", "ACG": "T",
"AAU": "N", "AAC": "N", "AAA": "K", "AAG": "K",
"AGU": "S", "AGC": "S", "AGA": "R", "AGG": "R",
"GUU": "V", "GUC": "V", "GUA": "V", "GUG": "V",
"GCU": "A", "GCC": "A", "GCA": "A", "GCG": "A",
"GAU": "D", "GAC": "D", "GAA": "E", "GAG": "E",
"GGU": "G", "GGC": "G", "GGA": "G", "GGG": "G", }
```

```
aminoacids = ".join(sorted(list(set([v for k,v in dic.items() if v <> "STOP"])))) print aminoacids
```

```
# UniProt Protein Database access IDs proteins = ['A2Z669', 'B5ZC00', 'P07204_TRBM_HUMAN',
'P20840_SAG1_YEAST']
```

```
handle = Expasy.get_sprot_raw(proteins[0]) seq_record = SeqIO.read(handle, "swiss") handle.close()
print print seq_record
```

```
</sxh>
```

## Références

- [Using biological databases to teach evolution and biochemistry](#)
- [Rosalind](#), plateforme d'apprentissage de la programmation en bioinformatique
- [GenBank](#)
- [Biopython](#)
- <https://en.wikipedia.org/wiki/Bioinformatics>
- [https://en.wikipedia.org/wiki/Open\\_Bioinformatics\\_Foundation](https://en.wikipedia.org/wiki/Open_Bioinformatics_Foundation)
- [https://en.wikipedia.org/wiki/FASTA\\_format](https://en.wikipedia.org/wiki/FASTA_format)
- [https://en.wikipedia.org/wiki/List\\_of\\_open-source\\_bioinformatics\\_software](https://en.wikipedia.org/wiki/List_of_open-source_bioinformatics_software)
- <http://www.amberbiology.com/>, "Python For The Life Sciences. A gentle introduction to Python for life scientists" (à paraître)

- références sur la lecture de fichiers :

- [http://www.uniprot.org/help/programmatic\\_access#id\\_mapping\\_python\\_example](http://www.uniprot.org/help/programmatic_access#id_mapping_python_example)
- <http://www.python-simple.com/python-biopython/Lecture-ecriture-sequences.php>

From:

<https://dvillers.umons.ac.be/wiki/> - **Didier Villers, UMONS - wiki**

Permanent link:

<https://dvillers.umons.ac.be/wiki/teaching:progappchim:bioinformatic?rev=1458044954>

Last update: **2016/03/15 13:29**

