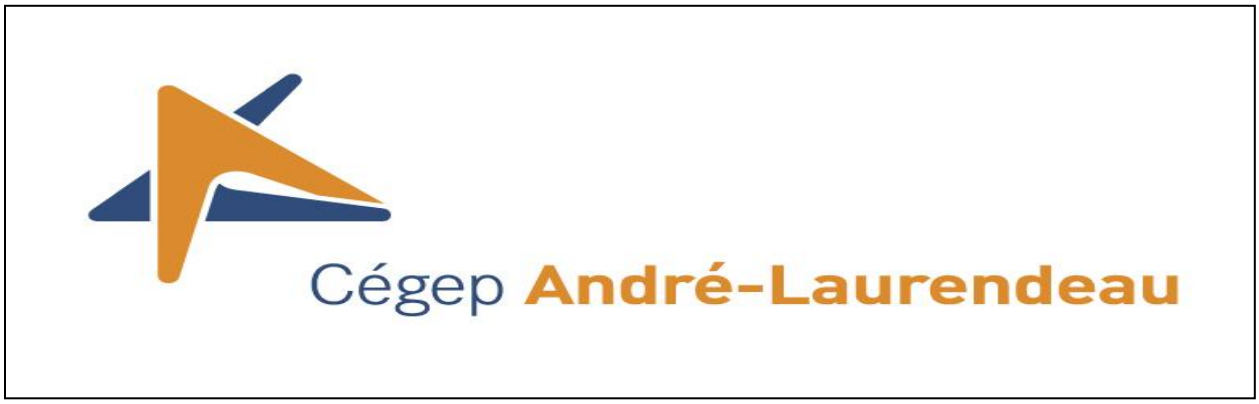


Travail pratique 1 : Langage Python

Conversion de fichiers en génétique

Cours 420-555-AL
Programmation appliquée aux objets connectés

Remettre avant **mardi 16 septembre à minuit**

8%

Objectifs

- Écrire un programme complet en Python.
- Démontrer une maîtrise des notions suivantes : fonctions, lecture et écriture de fichiers, utilisation des expressions régulières (regex), programmation objet, interface graphique.

Directives importantes

- Ce travail pratique **peut** être réalisé en équipe de deux. Les « clones » ou les copies trop semblables valent 0.
- Remettre avant le **mardi 16 septembre à minuit** le code dans un répertoire TP1 de votre repo Github Classroom.
- Au-delà du 16 septembre, une pénalité de 10% par jour de retard sera appliquée jusqu'à concurrence de 5 jours. Plus de 5 jours de retard entraîne la note 0.
- Ce travail compte pour **8%** de votre session.
- Il n'est pas permis d'utiliser de modules autre que ceux de la librairie standard.
- Deux fichiers d'entrées vous sont fournis :
    1. complexe_1.cmp (avec ses deux fichiers de sorties gene_1.sim et gene_2.sim, vous permettant de tester votre code)
    2. complexe_2.cmp (avec ses 101 fichiers de sorties, vous permettant de tester votre code)

" Conversion de fichiers en génétique"

Votre tâche consiste à convertir un fichier d'information génétique d'un format complexe (extension .cmp) vers un format simplifié (extension .sim). Plus spécifiquement, vous allez extraire les gènes qu'on retrouve dans le fichier d'entrée. Ainsi, à partir d'un fichier d'entrée complexe unique contenant un ou plusieurs gènes, nous allons créer un ou plusieurs fichiers simplifiés avec un seul gène par fichier.

Vous trouverez en page suivante un exemple de fichier d'entrée avec deux gènes, les deux fichiers de sortie apparaissant juste au-dessous.

Fichier d'entrée au format complexe (complexe_1.cmp)

```
LOCUS       SCU49845     296 bp    DNA    PLN       21-JUN-1999
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene and Axl2p
            (AXL2) and Rev7p (REV7) genes, complete cds.
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    cerevisiae
SOURCE      Saccharomyces cerevisiae (baker's yeast)
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Saccharomycotina;
            Saccharomycetales; Saccharomycetaceae; Saccharomyces.
FEATURES             Location/Qualifiers
     source          1..5028
                     /organism="Saccharomyces cerevisiae"
                     /db_xref="taxon:4932"
                     /chromosome="IX"
                     /map="9"
     CDS             <1..206
                     /codon_start=3
     gene            10..99
                     /gene="AXL2"
     CDS             687..3158
                     /gene="AXL2"
     gene            complement(150..200)
                     /gene="REV7"
     CDS             complement(3300..4037)
                     /gene="REV7"
ORIGIN
        1 gatcctccat atacaacggt atctccacct caggtttaga tctcaacaac ggaaccattg
       61 ccgacatgag acagttaggt atcgtcgaga gttacaagct aaaacgagca gtagtcagct
      121 ctgcatctga agccgctgaa gttctactaa gggtggataa catcatccgt gcaagaccaa
      181 gaaccgccaa tagacaacat atgtaacata tttaggatat acctcgaaaa taataaaccg
      241 ccacactgtc attattataa ttagaaacag aacgcaaaaa ttatccacta tataat
//
```

Nom de l'organisme décodé

Emplacement d'un gène dans la séquence : positions 10 à 99 soulignées plus bas.

Séquence génétique complète. La séquence soulignée est celle du gène 1.

Fichiers de sortie simplifié (.sim)

Fichier gene_1.sim

```
>Saccharomyces cerevisiae:1:10:99:codant
tatacaacggtatctccacctcaggtttagatctcaacaacggaaccattgccgacatgagacagttaggtatcgtcgag
agttacaagc
```

Fichier gene_2.sim

```
>Saccharomyces cerevisiae:2:150:200:complémentaire
atgttgtctattggcggttcttggtcttgcacggatgatgttatccaccct
```

Mode textuel

Le format des fichiers de sortie est le suivant :

LIGNE 1

> Nom de l'organisme décodé : numéro séquentiel que vous avez attribué au gène : début de la séquence :fin de la séquence :codant ou complémentaire[1]

LES LIGNES SUIVANTES

La séquence du gène avec un maximum de 80 caractères par ligne

Comment savoir si un gène est de type « codant » ou « complémentaire »? Il faut analyser comment, dans le fichier d'entrée, sont spécifiés les intervalles pour retrouver la sous-séquence correspondante. Les gènes codants sont suivis par un intervalle du type **début..fin** ou **<début..>fin**, alors que les gènes complémentaires sont suivis par **complement(début..fin)** ou **complement(<début.>fin)**. Début et fin sont des entiers correspondant au sous-intervalle de la séquence complète de l'organisme (à la toute fin du fichier d'entrée).

Faites attention, la construction de la séquence d'un gène complémentaire nécessite quelques manipulations supplémentaires :

1. Dans la séquence extraite, il faut inverser les bases « a » et « t » ainsi que les bases « g » et « c ».
2. Il faut ensuite inverser la séquence au complet : e.g « atgc » devient « cgta ».

Finalement, votre programme ne pourra être lancé que de l'une des deux manières suivantes :

- python tp1.py -t nom_du_fichier.cmp *mode texte*
- python tp1.py -g *mode graphique (voir la section sur le mode graphique)*

… et devra générer les affichages suivants :

```
Le nombre de lignes lues dans le fichier d'entrée
Le nom de l'organisme décodé
Le nombre de gènes codants et complémentaires trouvés
Le nombre de bases complètes[2]
```

[1] **Gène codant** : gène dont la séquence d'ADN est utilisée directement pour coder une protéine. **Gène complémentaire** : gène dont la séquence est complémentaire à un gène codant et qui ne code généralement pas pour une protéine.

[2] Ceci est en fait la longueur de la séquence totale. C'est le nombre apparaissant sur la première ligne du fichier d'entrée .cmp, tout de suite avant « bp »

```
Gène 1 écrit dans gene_1.sim
Gène 2 écrit dans gene_2.sim
```

Par exemple, pour le fichier complexe_1.cmp :

```
Nombre de lignes lues dans le fichier complexe_1.cmp : 33
Organisme : Saccharomyces cerevisiae
2 gènes trouvés : 1 codant(s) / 1 complémentaire(s)
Séquence complète : 296 bases
Gène 1 écrit dans le fichier complexe_1_gene_1.sim
Gène 2 écrit dans le fichier complexe_1_gene_2.sim
```

N'oubliez pas d'inscrire des commentaires judicieux dans vos programmes. Dans les premières lignes de vos sources Python, vous devriez au minimum inscrire les auteurs, la date et une courte description du programme.

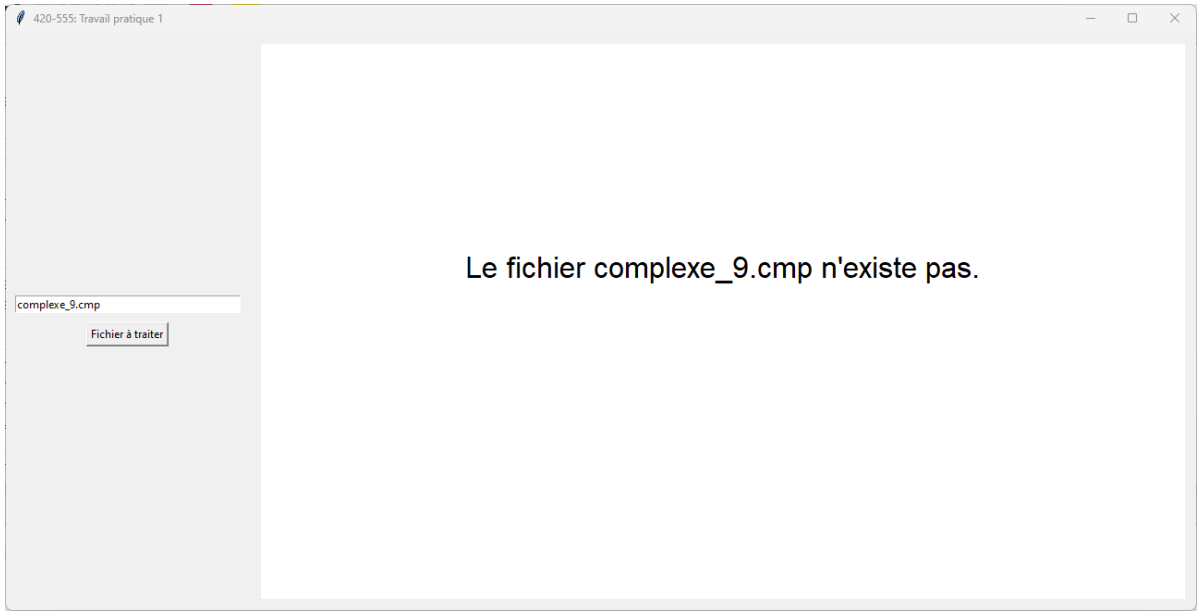
Mode graphique

Utilisez les bibliothèques **turtle** et **tkinter** pour créer une interface graphique à votre travail. L'interface aura l'allure suivante :

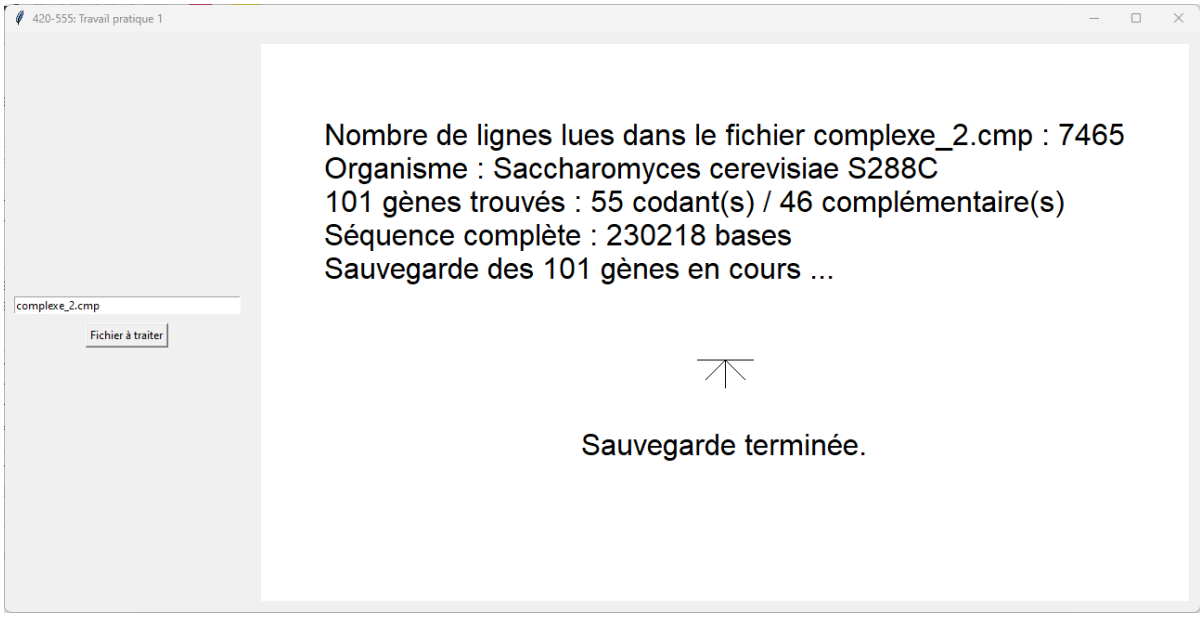
- La zone bleue est occupée par une frame tkinter
- La zone rouge est occupée par un canevas turtle
- Le bouton « **Fichier à traiter** » permet de lancer le traitement du fichier spécifié dans la boîte de texte au-dessus du bouton

Il y a deux cas possibles :

1. Le fichier n'existe pas :



2. Le fichier existe : voir ce clip pour comprendre l'animation.



Barème de correction

Élément	Barème
Mode textuel	**70%**
Ligne de commande avec arguments	/ 5
Fichier .cmp inexistant	/ 5
Le nombre de lignes lues dans le fichier d'entrée .cmp	/ 8
Le nom de l'organisme décodé	/ 10
Le nombre de gènes codants et complémentaires trouvés	/ 10
Le nombre de bases complètes	/ 10
Écriture dans les fichiers .sim	/ 12
Qualité du code (identificateurs, commentaires, etc.)	/ 10
Mode graphique	**30%**
Allure générale	/ 9
Fenêtre tkinter	/ 8
Fenêtre turtle, incluant l'animation	/ 9
Qualité du code (identificateurs, commentaires, etc.)	/ 4
TOTAL	**/ 100**

BON TRAVAIL !