Informatique SV L2
TP 4

1 Collection

1.1 Listes

Rappel d'opérations sur les listes :
- Créer une liste vide : liste = [ ]
- Créer une liste avec des éléments : liste = [x1,x2,x3,x4,...]
- Ajouter un élément à la fin de la liste : liste.append(x)
- Ajouter un élément à une certaine position : liste.insert(i,x) #ajoute x au i-éme élément
- Trier une liste : liste.sort()
- Nombre d'éléments dans la liste : len(liste)
- Maximum/minimum : max(liste),min(liste)
- Appartenance : x in liste
- Concaténation : liste1 + liste2
- Nombre d'occurence d'un élément : liste.count(x)
- Supprimer le i-éme élément : liste.pop(i)
- Supprimer un élément : liste.remove(x)

Dans la console, exécuter ces instructions et observez :

```
l=[]
l.append(17)
l.append(54)
l.append(23)
l
l.insert(1,47)
l
l[1 :3]
l[ :2]
l[2 :]
l[2]=47
l
l.count(47)
10 in l
17 in l
len(l)
max(l)
min(l)
l.pop(1)
l
l.remove(47)
l
l2=[1,8,47]
l2
l1=l+l2
l1
```

```
l1.sort()
l1
for n in l1 :
    print n
```

1.2 Éliminer les doublons d'une liste

Écrivez un script qui recopie une liste dans une autre,en omettant les doublons. La liste finale devra être triée. Essayer avec la liste [3,2,7,8,2,4,4,7,1]

1.3 Dictionnaire

Rappel d'opération sur les dictionnaires :

- Créer un dictionnaire vide : d={}
- Créer un dictionnaire avec des éléments : d={cle1 : val1 , cle2 : val2,cle3 : val3,...}
- Obtenir la valeur associé à une clé : d[cle]
- Définir une valeur associé à une clé : d[cle]=val
- Supprime une valeur : del d[cle]
- Vide le dictionnaire : d.clear()
- Nombre de valeur : len(d)
- Retourne Vrai if la clé existe Faux sinon : d.has_key(cle)
- Retourne la liste de toute les clé : d.keys()
- Retourne la liste de toute les valeur : d.values()

Dans la console, exécuter ces instructions et observez :

```
d={}
d['Pingoin']=5
d[78]='Phoque'
d
d[78]
d['Pingoin']
d[5]
del d[78]
d['Elephant']='Gris'
d['Tigre']='Jaune'
d['Pingoin']='Noir'
d
d['Tigre']
len(d)
d.has_key('Pingoin')
d.has_key('Phoque')
d.keys()
d.values()
a2={'Sandrine' :  '0145478596' , 'Mathieu' : '0354789568', 'Tom':'0541257136'}
a2['Tux']='0145914358'
for nom in a2:
    print nom,a2[nom]
noms=a2.keys()
noms
noms.sort()
noms
for nom in noms:
    print nom,a2[nom]
```

1.4 Opération sur l'annuaire

1.4.1 Remplir l'annuaire

Écrire un programme qui crée un annuaire à partir des données entrez par l'utilisateur :

1. Créer un annuaire vide
2. Demander à l'utilisateur un nom, et un numéro de téléphone
3. Ajouter le nom à l'annuaire
4. Recommence à partir de 2 jusqu'à ce que l'utilisateur donne un nom vide (il appuie sur la touche entrée)
5. Afficher l'annuaire

1.4.2 Rechercher dans un annuaire

A partir du dictionnaire : annuaire={'Sandrine' : '0145478596' , 'Mathieu' : '0354789568', 'Tom':'0541257136'}, écrivez et tester le programme suivant :

1. Demander à l'utilisateur un nom à rechercher
2. Afficher le résultat si le nom existe, une erreur sinon
3. Recommencer a partir de 1 jusqu'à ce que l'utilisateur donne un nom vide

2 Lire dans un fichier

Lire un fichier texte ligne par ligne :

```
f=open('notes.txt')
for l in f :
    print l
```

Tester le programme ci-dessus.

2.1 Phrase la plus longue

Écrivez un script qui recherche et affiche la phrase (ligne) la plus longue d'un texte. Tester avec le fichier climat.txt

2.2 Statistique sur les notes à partir d'un fichier

2.2.1 Lire les notes à partir d'une fichier

Soit un fichier dont les lignes contiennent une note. Écrire un programme qui place ces notes dans une liste. Avant d'jaouter la note dans la liste, convertir la ligne lu en float, en utilisant la fonction float(l).

2.2.2 Statistique

Reprendre les fonctions de statistiques sur les notes. Écrire un programme qui lit les note à partir du fichier et affiche les statistiques de ces notes.

Facultatif : Récrire la fonction mediane(liste) en utilisant l'opération liste.sort()

3 Traduction d'une séquence d'ADN

Dans la suite on considéra qu'une séquence d'ADN est stocké dans une chaîne de caractère par les lettres a,c,t et g. Chaque fonction sera testé en écrivant un programme principale ad-hoc, en prenant comme entrée la chaine contenu dans le fichier seqADN.py

3.1 Vérification si une séquence d'ADN est correcte

Écrire une fonction seqADNCorrecte(chaine) qui renvoi la valeur Vrai (True) si une séquence d'ADN est correcte (chaîne constitué uniquement des lettre a,c,t et g) Faux (False) sinon.

3.2 Statistique

Écrire une fonction statBaseADN(chaine) qui renvoi une liste comprenant le pourcentage de base a,c,t et g.

3.3 Traduction

1. Un dictionnaire que vous trouverez dans le fichier codeGen.py, fait la correspondance entre un codon et un acide-aminé. Essayez-le avec quelques codon.
2. Écrire une boucle qui affiche les codons un par un, c'est à dire en affichant les séquences de 3 bases. Utiliser la méthode de découpage de mot vu dans le TP 3.
3. Au lien d'afficher les codons, modifier le programme pour afficher une chaînes d'acide aminé en utilisant le dictionnaire.
4. Écrire une fonction traduction(seq) qui, à partir d'une séquence d'ADN, renvois la protéine correspondante.

3.4 Lecture de la séquence à partir d'un fichier

Écrire un programme qui utilise les 3 fonctions précédentes en lisant les séquences d'ADN à partir d'un fichier. Le programme devra :

- Lire une ligne contenant la séquence
- Vérifier si la séquence est correcte
- Afficher les statistiques
- Traduire la chaîne en protéine

Tester le programme avec seq.txt

Facultatif :

- Ajouter la possibilité de donner le nom de fichier comme paramètre de lancement du programme. Ex *python traduction.py seq.txt*
- Ajouter les statistiques sur les codons.