TP2 sur les séquences

En cas de difficultés, pensez à relire la leçon avant de demander de l'aide au professeur.

1. Tuple

. Exercice 1 : Accès

Aide : regarder l'exercice 1 du TP précédent

1. Créer le tuple `t1` contenant les valeurs : 5, k, 12, p, 14, 8, p
   - Afficher sa longueur, son élément d'indice 3 et son élément d'indice 5
2. Créer le tuple `t2` contenant les valeurs : 2, 11, 5, f, nuit, d
   - Afficher sa longueur puis les éléments d'indice 1 à 4

. Exercice 2 : Recherche de valeurs

1. Écrire l'algorithme correspondant à ce code.

```
def occurrences ( v, t : tuple) :
    oc = 0
    for elt in t :
        if elt == v : oc = oc +1
    return oc

print( occurrences( 0 , (2,6,0,4,7,0,3)) )
```

.....
.....
.....
.....
.....
.....

2. Recopier et exécuter le code ci-dessus. Quel affichage obtient-on ? ……
3. Écrire le docstring (« chaîne de documentation ») de cette fonction.
4. Tester cette fonction avec d'autres paramètres.

2. Liste

. Exercice 3 : Poids maximum pour une livraison

Aide : regarder l'exercice 4 du TP précédent

1. Écrire une fonction `poidsTotalValide(Lpoids)` qui prend en arguments la liste des entiers représentant les poids des paquets à envoyer. Si le poids total est strictement inférieur à 105kg, la fonction doit alors renvoyer `True`, sinon elle doit renvoyer `False`.
2. Vérifier que `poidsTotalValide( [20, 45, 30] )` renvoie `True`
   et que `poidsTotalValide( [30, 25, 20, 30] )` renvoie `False`.
3. Importer le module random
4. Que fait cette ligne ? `L=[randint(1,30) for i in range(randint(1,12)) ]`
   .....
5. Utiliser la ligne précédente pour effectuer d'autres tests sur la fonction `poidsTotalValide`.

. Exercice 4 : ADN

Une molécule d'ADN est formée d'environ six milliards de nucléotides.
L'ordinateur est donc un outil indispensable pour l'analyse de l'ADN.
Dans un brin d'ADN il y a seulement quatre types de nucléotides qui sont notés A,C,T ou G.
Une séquence d'ADN est donc un long mot de la forme :TAATTACAGACCTGAA...

1. Que fait cette ligne ? `seqADN = list("CTCCGTT")`
   .....

2. Que fait cette ligne ? `print(seqADN[2:5])`

   .....

3. Recopier et compléter la fonction `position` pour qu'elle renvoie le premier indice à partir duquel les éléments de la liste `seqADN` sont les mêmes que ceux de la liste `code`. Autrement dit, cette fonction permet de connaître la première position où un code est présent dans une séquence ADN.

```
def position( code, seqADN) :
    for i in range( ..... ):
        if seqADN[i] == code[0]:
            if code == seqADN[i: i + ..... ] : return i
    return None
```

4. Recopier cette fonction et vérifier que `position( list("CCG"), list("CTCCGTT") )` renvoie 2 .
5. Importer le module random
6. Que fait cette ligne ? `L=[ choice(('A','C','T','G')) for i in range(100) ]`

   .....

7. Utiliser la ligne précédente pour effectuer d'autres tests sur la fonction `position`.

3. Pour les plus rapides

. Exercice 5 : Le problème des chapeaux (facultatif)

Lors d'une soirée au théâtre, chacune des n personnes invitées dépose son chapeau au vestiaire avant le spectacle. A l'issue de celui-ci, survient une panne d'électricité. Dans ces conditions, chaque personne récupère un chapeau au hasard. Quelle est la probabilité qu'au moins une personne récupère son propre chapeau ? Il s'agit ici de déterminer une valeur approchée de la probabilité cherchée à l'aide d'un grand nombre de simulations

1. Créer une fonction `melange( l1 )` qui renvoie une copie mélangée de la liste l1
   Aide : utiliser shuffle(x) du module random
2. Créer une fonction `commun( l1, l2 )` qui renvoie soit le booléen `True` (si les deux listes l1 et l2 ont au moins une valeur commune à la même position) soit le booléen `False`.
3. Demander à l'utilisateur la valeur de n. Stocker-la dans une variable.
4. Créer la liste `chapeaux` contenant les entiers de 1 à n.
5. Faites un premier test : afficher la liste `chapeaux`, la liste renvoyée par `melange( chapeaux)` ainsi que la valeur renvoyée par un appel de la fonction `commun` avec ces deux listes en paramètre. Corriger si besoin les fonctions `melange` et `commun`.
6. Compléter le programme pour qu'il fasse 1000 essais et affiche la fréquence des simulations pour lesquelles au moins une personne récupère son chapeau.

. Exercice 6 : Distance de Hamming (facultatif)

1. Écrire une fonction `hamming( t1, t2 )` qui prend en paramètres deux tuples et qui renvoie le nombre d'indices auxquels les deux tableaux diffèrent.
   Aide : Boucler sur les indices du tuple le plus petit pour comparer ses valeurs à celle de l'autre.
   Considérer que les tableaux diffèrent pour les indices où un seul tuple est défini.
2. Tester cette fonction avec différents couples de tuples.
3. Écrire le docstring (« chaîne de documentation ») de cette fonction.