1 Listes

1.1 Longueur d'une liste

Écrire une fonction $longueur(L)$ qui prend en paramètre une liste L et renvoie sa longueur. On n'utilisera pas la fonction *len*.

1.2 Maximum d'une liste

♡ Écrire une fonction $maximum(L)$ qui prend en paramètre une liste de nombres, L, et renvoie un couple composé de l'indice du maximum et du maximum de la liste L. Si le maximum est rencontré plusieurs fois, on renverra le plus petit indice.

2 Chaînes de caractères

La mémoire d'un ordinateur conserve toutes les données sous forme binaire, il n'existe pas de méthode pour stocker directement les caractères. Chaque caractère posséde son équivalent numérique, c'est le code ASCII. Voici un extrait du tableau permettant de faire cette conversion.

48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	|
53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	⌂
56	8	80	P	104	h	128	Ç
57	9	81	Q	105	i	129	ü
58	:	82	R	106	j	130	é
59	;	83	S	107	k	131	â
60	<	84	T	108	l	132	ä
61	=	85	U	109	m	133	à
62	>	86	V	110	n	134	å
63	?	87	W	111	o	135	ç
64	@	88	X	112	p	136	ê
65	A	89	Y	113	q	137	ë
66	B	90	Z	114	r	138	è
67	C	91	[	115	s	139	ï
68	D	92	\	116	t	140	î
69	E	93	]	117	u	141	ì
70	F	94	^	118	v	142	Ä
71	G	95	_	119	w	143	Å

Les fonctions *ord* et *chr* permettent de passer d'une colonne de ce tableau à l'autre. Par exemple :

$>>> ord('b')$
98
$>>> chr(84)$
$'T'$

Dans tout l'exercice, on appelle *mot* une chaîne de caractères uniquement composée de lettres de 'a' à 'z', majuscules ou minuscules.

1. Écrire une fonction qui étant donné un mot renvoie la liste des codes ASCII des lettres qui le composent. On utilisera une boucle for.
2. Soit c un mot donné. Écrire une liste en compréhension qui répond à la question précédente.
3. Écrire une fonction qui étant donné un mot renvoie le même mot mais qu'avec des lettres minuscules.
4. On suppose dans cette question que tous les mots sont écrits en minuscules. Écrire une fonction $compare(c1,\ c2)$ qui renvoie $True$ si le mot $c1$ est placé avant $c2$ dans l'ordre alphabétique et $False$ sinon.

3 Tuples

3.1 Pair et impair

Écrire une fonction qui prend en argument un n-uplet composé d'entiers et renvoie un couple de deux listes : la première contenant les nombres pairs et la seconde les nombres impairs.

3.2 Représentation des nombres complexes

On représente le nombre complexe $a + ib$ sous la forme d'un couple (a, b). Écrire une fonction $mult(z1,\ z2)$ qui prend en paramètres deux nombres complexes $z1$ et $z2$ et renvoie le produit $z_1 z_2$.

4 Dictionnaires

4.1 Statistiques sur une chaîne de caractères

Écrire une fonction *stat* qui prend en paramètre une chaîne de caractères et renvoie un dictionnaire dont les clés sont les différentes lettres du texte et les valeurs sont le nombre d'occurrences de chaque lettre. On suppose le texte écrit en lettres majuscules non accentuées. Il peut y avoir dans le texte différents caractères de ponctuation mais ils n'apparaitront pas dans le dictionnaire.

4.2 Rapidité de la recherche d'un élément

Le but de cette question est de mesurer le temps mis par un algorithme pour chercher un élément dans une liste par rapport à une recherche dans un dictionnaire.

1. Former une liste L dont les éléments sont de la forme $[i,\ i**2]$ pour i allant de 0 à $10^6 - 1$. Mélanger cette liste avec la fonction $shuffle$ du module $random$.
2. Créer le dictionnaire d correspondant, c'est-à-dire que si la clé est i alors la valeur correspondante est $i**2$.
3. Écrire une fonction $recherche1(k)$ qui renvoie le deuxième élément de la sous-liste de la liste L dont le premier élément est k.
4. Écrire une fonction $recherche2(k)$ qui renvoie la valeur qui correspond à la clé k dans le dictionnaire d.
5. Mesurer le temps d'exécution de ces deux fonctions pour différentes valeurs de k. Conclure.

   Pour les mesures, on utilise la fonction *time* du module *time*. Le mode d'utilisation est le suivant :

```
import time as t
a = t.time() # temps avant exécution
# ici la fonction à exécuter
b= t.time() # temps à la fin de l'exécution
print(b - a) # on obtient le temps écoulé
```