



Programmation impérative en Python – SPUF21

Année 2024-2025 – Seconde session

Nom :

Prénom :

Numéro d'étudiant :

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

Durée : 2 heures.

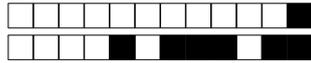
Aucun document n'est autorisé. L'usage de la calculatrice ou de tout autre appareil électronique est interdit.

Les exercices sont indépendants. Au sein d'un même exercice, vous pouvez utiliser les variables et fonctions des questions précédentes, même si vous n'avez pas su les faire; chaque question est donc indépendante.

À part les méthodes et fonctions de base, vous n'avez pas le droit d'utiliser les fonctions et les méthodes « avancées », sauf si l'énoncé vous conseille l'utilisation de certaines d'entre elles.

```
1 # Fonctions autorisées
2 range(...) len(...)
3 print(...) int(...)
4
5 # Méthodes
6 L.append(x)
```

```
1 # Par exemple les méthodes et fonctions suivantes sont entre autres interdites
2 max(...) min(...) sum(...) abs(...) eval(...)
3 s.split(...) s.index(...) L.extend(...)
4
5 # Vous n'avez pas le droit d'utiliser des compréhensions ou des slices
6 # À la place vous devez utiliser des boucles.
7 x in L
8 [ x for x in range(L) ]
9 chaine[début:fin:pas]
```



Exercice 1 Calculs binaires..... 4 points

0 0,5 1 1,5 2 2,5 3 3,5 4

1. Écrivez une fonction `affiche_calcul_binaire`(n) qui affiche le calcul de la représentation binaire de n, de sorte que l'on peut lire *verticalement* (de bas en haut) la représentation en binaire de n. Par exemple, pour $13 = (1101)_2$, on obtiendra dans la console :

```
1 >>> affiche_calcul_binaire(13)
2 1 car 13 = 2 × 6 + 1
3 0 car 6 = 2 × 3 + 0
4 1 car 3 = 2 × 1 + 1
5 1
6 >>> affiche_calcul_binaire(14)
7 0 car 14 = 2 × 7 + 0
8 1 car 7 = 2 × 3 + 1
9 1 car 3 = 2 × 1 + 1
10 1
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. Écrivez une fonction `récursive_binaire`(n) qui renvoie l'écriture binaire de n sous forme de chaîne dans le sens de lecture **usuel**. On pourra utiliser la fonction `str` (rappel : `str(0)` vaut `"0"`).

```
1 >>> binaire(2)
2 '10'
3 >>> binaire(13)
4 '1101'
5 >>> binaire(14)
6 '1110'
```

.....

.....

.....

.....

.....

.....

.....

.....



Exercice 2 Des formules et des arbres 5 points

0 0,5 1 1,5 2 2,5 3 3,5 4 4,5 5

Dans cet exercice nous allons travailler avec des arbres. Pour manipuler ces derniers, vous ne pouvez utiliser que les fonctions suivantes :

- `arbre`(r, Ag, Ad) renvoie un arbre de racine r et de fils Ag (gauche) et Ad (droit);
- `est_feuille`(A) renvoie `True` si A est une feuille, et `False` sinon;
- `racine`(A) renvoie la racine de l'arbre A : '+', '-', '*' ou '/';
- `fg`(A) renvoie le fils gauche de A;
- `fd`(A) renvoie le fils droit de A.

1. Avec les fonctions ci-dessus, définir une variable F correspondant à la formule $\frac{x+y}{2}$ représentée sous forme d'arbre.

.....
.....

2. Écrire une fonction `contient`(v, arbre) qui renvoie `True` si et seulement si une feuille de l'arbre est égale à v. Sinon, la fonction renverra `False`.

```
1 >>> contient('x',F)
2 True
3 >>> contient('a',F)
4 False
5 >>> contient(5,F)
6 False
7 >>> contient(2,F)
8 True
```

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

3. On appelle environnement une liste de couples (nom, valeur).
Écrire une fonction `variable`(env, nom) qui renvoie la valeur associée à nom dans la liste env. Si la variable n'est pas dans la liste on renverra 0. Si le nom apparaît plusieurs fois on renverra la première occurrence.

```
1 >>> env = [ ("a",3), ("x",2) , ("y",5), ("x", 4)]
2 >>> variable(env,"x")
3 2
4 >>> variable(env,"b")
5 0
```




Exercice 3 Collections 6 points

0 0,5 1 1,5 2 2,5 3 3,5 4 4,5 5 5,5 6

Dans cet exercice, on travaille sur des collections avec multiplicités. Par exemple si on se donne les valeurs suivantes : « A A D A B D A », on les stockera sous la forme d'une liste de tuples indiquant pour chaque valeur combien de fois elle apparaît : [("A", 4), ("B", 1), ("D", 2)]. L'ordre des éléments dans la liste n'a pas d'importance.

1. Écrire une fonction `nouvelle_collection()` qui renvoie une collection vide.

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

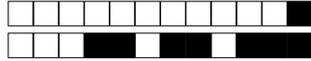
2. Écrire une fonction `lire(x,C)` qui renvoie la multiplicité associée à la valeur x. Si x n'est pas présent dans C, on renverra 0.

```
1 >>> C = [("A",4), ("B",1), ("D",2)]
2 >>> lire("B",C)
3 1
4 >>> lire("U",C)
5 0
```

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

3. Écrire une fonction `ajout(x,n,C)` qui modifie la collection C en ajoutant n occurrences à x.

```
1 >>> C
2 [('A', 4), ('B', 1), ('D', 2)]
3 >>> ajout("D",10,C)
4 >>> C
5 [('A', 4), ('B', 1), ('D', 12)]
6 >>> ajout("E",3,C)
7 >>> C
8 [('A', 4), ('B', 1), ('D', 12), ('E', 3)]
9 >>> ajout("A",-4,C)
10 >>> C
11 [('A', 0), ('B', 1), ('D', 12), ('E', 3)]
```

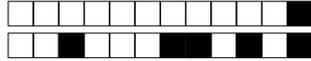


.....
.....
.....
.....
.....
.....
.....
.....

4. Écrire une fonction `ramasse_miette(C)` qui renvoie une nouvelle collection sans les éléments de multiplicité nulle. La fonction affichera aussi le nombre de valeur supprimées.

```
1 >>> C = [("A",4), ("B",0), ("D",0), ("E",-3)]  
2 >>> C = ramasse_miette(C)  
3 2 valeur(s) supprimée(s)  
4 >>> C  
5 [('A', 4), ('E', -3)]  
6 >>> C = ramasse_miette(C)  
7 0 valeur(s) supprimée(s)  
8 >>> C  
9 [('A', 4), ('E', -3)]
```

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



2. Écrire une fonction `rechercher`(chaîne, texte) qui renvoie le nombre d'apparitions de la chaîne dans le texte. On pourra évidemment utiliser la fonction précédente.

```
1 >>> rechercher("toto", "to")
2 0
3 >>> rechercher("thon", "Anthony le python dépasse un thon lors du marathon")
4 4
5 >>> rechercher("outou", "Groutoutou")
6 2
```

.....

.....

.....

.....

.....

.....

.....

.....

3. On se donne un fichier texte `nerval.txt` contenant un poème et représenté ci-contre. Écrire une fonction `début_ligne`(chaîne, nom_fichier) qui affiche toute les lignes commençant par chaîne parmi les lignes du fichier dont le nom est en paramètre.

```
1 >>> début_ligne("Un", "nerval.txt")
2 Un air très vieux, languissant et funèbre,
3 Un coteau vert, que le couchant jaunait,
4 >>> début_ligne("Choucroute", "nerval.txt")
5 >>> début_ligne("Il", "nerval.txt")
6 Il est un air pour qui je donnerais
```

Fantaisie (de Gérard de Nerval)

Il est un air pour qui je donnerais
Tout Rossini, tout Mozart et tout Weber,
Un air très vieux, languissant et funèbre,
Qui pour moi seul a des charmes secrets.

Or, chaque fois que je viens à l'entendre,
De deux cents ans mon âme rajeunit :
C'est sous Louis treize ; et je crois voir s'étendre
Un coteau vert, que le couchant jaunait,

Puis un château de brique à coins de pierre,
Aux vitraux teints de rougeâtres couleurs,
Ceint de grands parcs, avec une rivière
Baignant ses pieds, qui coule entre des fleurs ;

Puis une dame, à sa haute fenêtre,
Blonde aux yeux noirs, en ses habits anciens,
Que, dans une autre existence peut-être,
J'ai déjà vue... - et dont je me souviens !

.....

.....

.....

.....

.....

.....

.....

.....