



Programmation impérative en Python – SPUF21

Année 2023-2024 – Partiel

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 et mots-clés autorisés
```

```
6 L.append(x)
```

```
7 x in L
```

```
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(...)
```

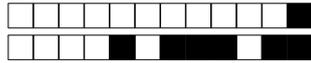
```
4
```

```
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 for x in range(L) ]
```

```
8 chaine[début:fin:pas]
```



Exercice 1 Écrire et tester son code 3,5 points

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

1. Écrire une fonction `est_strictelement_positif(L)` qui renvoie `True` si tous les éléments de `L` sont strictement positifs. On supposera que `L` est une liste d'entiers et on renverra `True` lorsque `L` est vide.

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

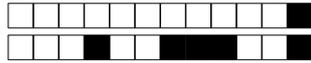
2. Écrire (au moins) trois tests avec `assert` pour la fonction précédente. Vos tests devront être suffisamment diversifiés.

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

3. On cherche à afficher tous les multiples de 10 jusqu'à 50. La fonction suivante est-elle correcte? Justifiez et en cas d'erreur, proposez une correction.

```
1 def étrange():  
2     i=0  
3     while i<=50:  
4         if i%10 == 0:  
5             print(i)  
6         else:  
7             i = i+1
```

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



Exercice 3 Des suites et des boucles 4 points

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

1. Écrire de manière récursive la fonction **u**(n) correspondant à la suite récursive définie par :

$$\begin{cases} u_0 &= 1 \\ u_{n+1} &= 0,9 \times u_n + 2 \end{cases}$$

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

On se propose dans les trois questions suivantes de calculer la suite de manière itérative afin de trouver sa limite. On ne devra pas utiliser la fonction u(n) précédemment définie.

2. Écrire la fonction **suisvant**(x) qui calcule et renvoie le prochain terme de la suite. Par exemple, si $x = u_3$, alors **suisvant**(x) sera égal à u_4 .

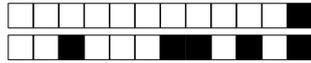
```
1 >>> suisvant(1) # 0,9 * 1 + 2
2 2.9
3 >>> suisvant(10) # 0,9 * 10 + 2
4 11.0
```

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

3. Écrire la fonction **absolue**(x) qui calcule et renvoie la valeur absolue du nombre x.

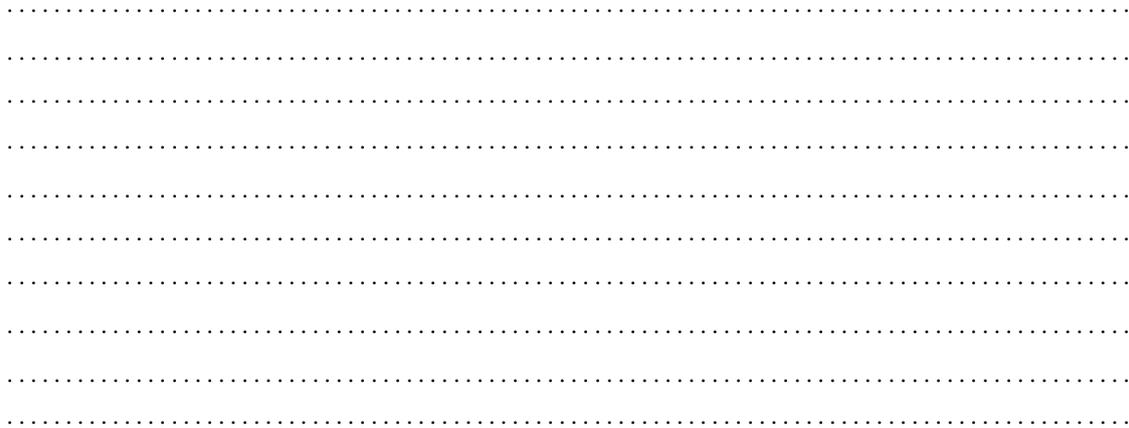
```
1 >>> absolue(3)
2 3
3 >>> absolue(-5)
4 5
```

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



2. Vous souhaitez seulement avoir la fin de liste. Écrire une fonction `tail(liste, n)` qui renvoie les `n` derniers éléments de la liste `liste`.

```
1 >>> L = [1,2,3,4,5,6,7,8,9]
2 >>> tail(L,3)
3 [7, 8, 9]
4 >>> tail(L,0)
5 []
6 >>> tail(L,1000)
7 [1, 2, 3, 4, 5, 6, 7, 8, 9]
```



Il est temps maintenant de créer une IA commentant le cours de la bourse et donnant des conseils pertinents d'investissement.

3. Écrire une fonction `journaliste_ia(produit,historique,n)` qui prend l'historique du cours des crypto-monnaies et affiche un commentaire pour les `n` dernières valeurs concernant le produit donné en paramètre. Il y aura trois types de message : un quand le prix augmente, un autre quand le prix diminue et un dernier quand le prix stagne. On respectera l'affichage proposé dans l'exemple.

```
1 historique = [("2023-10-01", 0.4, "#cc"), # 0,4€ en octobre
2             ("2023-11-01", 0.5, "#cc"), # 0,5€ en novembre --> Ça monte
3             ("2023-12-01", 1, "#cc"), # 1€ en décembre --> Ça monte
4             ("2024-02-01", 0.5, "#bc"), #
5             ("2024-01-01", 0.3, "#cc"), # 0,3€ en janvier ---> Ça baisse
6             ("2024-02-01", 0.3, "#cc"), # 0,3€ en février ---> Ça ne bouge pas
7             ("2024-03-01", 0.5, "#bc"), #
8             ("2024-03-01", 0.7, "#cc")] # 0.7€ en mars ----->Ça monte
```

```
1 >>> journaliste_ia("cc",historique, 4)
2 2023-12-01 Ça monte, il faut acheter !
3 2024-01-01 Ça chute, vendez tout !
4 2024-02-01 Ça ne bouge pas, investissez ailleurs
5 2024-03-01 Ça monte, il faut acheter !
```




+1/10/51+