+1/1/60+

Programmation impérative en Python — SPUF21

Année 2020-2021 — 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.

```
# Fonctions autorisées
len(…)
range(…)
print(…)

# Méthodes autorisées
L.append(x)
'{} = {}'.format('1+1',2)
```

```
# Par exemple les méthodes et fonctions suivantes sont entre autres interdites
max(…) min(…) sum(…)
s.split(…) s.index(…) L.extend(…)

# Vous n'avez pas le droit d'utiliser des compréhensions ou des slices
# sur les chaînes. À la place vous devez utiliser des boucles.
[ x for x in range(L) ]
chaine[début:fin:pas]
```

+1/2/59+

Exercice 1 Questions de cours . . . . . 3 points

☐ 0 ☐ 0,5 ☐ 1 ☐ 1,5 ☐ 2 ☐ 2,5 ☐ 3

1. Qu'affiche le code suivant ?

```
L1 = [1,2,3]
L2 = L1
L2.append(4)
print(L1,L2)
L1 = L1 + [3]
print(L1,L2)
L2.append(7)
print(L1,L2)
```

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2. Écrire une fonction `copier_liste(L)` qui renvoie une liste ayant le même contenu que `L` mais indépendante de cette dernière.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

3. Votre fonction permet-elle de copier des matrices (des listes de listes) ? Pourquoi ?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Exercice 2 Tapis . . . . . 1 points

☐0 ☐0,5 ☐1

Écrire la fonction `tapis(n)` qui reproduit le motif ci-dessous de largeur et hauteur `n`.

```
>>> tapis(6)
.O.O.O
O.O.O.
.O.O.O
O.O.O.
.O.O.O
O.O.O.
>>> tapis(11)
.O.O.O.O.O.
O.O.O.O.O.O
.O.O.O.O.O.
O.O.O.O.O.O
.O.O.O.O.O.
O.O.O.O.O.O
.O.O.O.O.O.
O.O.O.O.O.O
.O.O.O.O.O.
```

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Exercice 3 Arithmétique . . . . . 5 points

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

1. Écrire une fonction `liste_diviseur(n)` qui renvoie la liste de tous les diviseurs de `n` (`1` et `n` compris).

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2. Écrire une fonction `moyenne_diviseur(N)` qui renvoie la moyenne du nombre de diviseurs des entiers de `1` à `N`. Par exemple `1` a `1` diviseur, `2` et `3` ont `2` diviseurs chacun et `4` en a `3`. Ainsi, pour `N = 4`, `moyenne_diviseur(N)` devra renvoyer la moyenne de 1, 2, 2 et 3 c'est à dire `2.0`.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

3. On définit par récurrence la suite u_n par

$$\begin{cases} u_0 = u_1 & = 1 \\ u_{n+1} & = k \times u_{k-1} \text{ où } k \text{ est le nombre de diviseurs de } n+1 \end{cases}$$

Écrire la fonction `u(n)` correspondante, calculant u_n de manière récursive.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

4. En mathématiques, un entier est dit parfait s'il est égal à la somme de ses diviseurs **stricts**. Par exemple 6 est parfait car c'est la somme de ses trois diviseurs : $1 + 2 + 3 = 6$. Écrire une fonction `est_parfait(n)` qui renvoie `True` si `n` est parfait et `False` sinon. On rappelle que l'usage de la fonction `sum` est interdite.

5. Écrire un programme `nombres_parfaits(k)` qui affiche les `k` premiers nombres parfaits.

Exercice 4 Jeux de 32 cartes . . . . . 5 points

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

On souhaite modéliser les cartes à jouer. On commence par définir **de manière globale** deux listes pour représenter les huit valeurs et les quatre couleurs (ce qui donne 32 cartes possibles) :

```
liste_valeur  = ['Sept', 'Huit', 'Neuf', 'Dix', 'Valet', 'Dame', 'Roi', 'As']
liste_couleur = ['Cœur', 'Trèfle', 'Carreau', 'Pique']
```

On crée une classe `Carte` comme indiqué ci-dessous. Les deux attributs doivent correspondre à des chaînes de caractères appartenant aux deux listes ci-dessus.

```
class Carte:
    def __init__(self,valeur,couleur):
        self.valeur  = valeur
        self.couleur = couleur
```

1. Définir une variable `vp` contenant un objet de la classe `Carte` représentant un « Valet de Pique ».

. . . . . .

. . . . . .

2. Écrire une fonction `créer_tas_de_carte()` qui renvoie une liste contenant les 32 cartes possibles (sept de cœur, dix de trèfle, etc.)

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

3. On veut s'assurer que l'utilisateur choisit des valeurs ou des couleurs uniquement parmis celles décrites dans les listes (`liste_valeur` et `liste_couleur`). Écrire une **méthode** `est_valide` qui renvoie `True` si les attributs `valeur` et `couleur` sont dans les listes correspondantes et `False` sinon.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

4. Pour pouvoir comparer rapidement les cartes entre elles, on souhaite créer un dictionnaire `Val` dont les clés sont les chaînes de `liste_valeur` et qui leur associe leur indice dans cette liste. Par exemple on aura :

```
>>> Val['Sept']  # Car liste_valeur[0]=='Sept'
0
>>> Val['Huit']  # Car liste_valeur[1]=='Huit'
1
>>> Val['As']    # Car liste_valeur[7]=='As'
7
```

Pour éviter de définir les 8 éléments à la main, écrire une fonction `créer_dictionnaire(L)` qui renvoie le dictionnaire `Val` vérifiant `Val[chaîne]==i` si et seulement si `L[i]==chaîne`.

5. On définit alors ce dictionnaire de manière globale avec l'instruction suivante.

```
>>> Val = créer_dictionnaire(liste_valeur)
```

En utilisant ce dernier, écrire une fonction `plus_grand(carte1,carte2)` qui prend deux objets de la classe Carte et qui renvoie `True` si la valeur de la première carte est supérieure ou égale à la valeur de la seconde. Sinon, la fonction renverra `False`.

Exercice 5 Trier un jeu de carte . . . . . 3 points

☐0 ☐0,5 ☐1 ☐1,5 ☐2 ☐2,5 ☐3

Cette partie est la suite de la précédente, mais reste indépendante. Tout ce dont vous aurez besoin est la fonction booléenne `plus_grand(carte1,carte2)` qui permet de comparer deux cartes.

1. Écrire une fonction `indice_du_minimum(L)` qui renvoie l'indice du minimum d'une liste de cartes L.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2. Écrire une fonction `supprimer_indice(L,i)` qui renvoie une nouvelle liste obtenue en supprimant l'élément `L[i]` à la liste L. Par exemple `supprimer_indice([0,2,4,6],1)` renverra `[0,4,6]`. On utilisera une boucle `for`.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

3. En utilisant les deux fonctions précédentes, écrire une fonction `trier_liste_cartes(L)` qui renvoie une nouvelle version de `L` dans laquelle les éléments sont triés par ordre croissant.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Exercice 6 Jouons au poker . . . . . 3 points

☐ 0 ☐ 0,5 ☐ 1 ☐ 1,5 ☐ 2 ☐ 2,5 ☐ 3

Une main est une liste de cartes (généralement cinq). L'objectif de cette partie et de vérifier si la main vérifie certaines propriétés (contient-elle deux cartes de même valeur) ? Pour simplifier le code, on supposera que la main est triée par ordre croissant (par exemple avec la fonction de l'exercice précedent).

1. Écrire une fonction `couleur(main)` qui renvoie `True` si et seulement si les cartes de la liste `main` sont toutes de la même couleur (Pique, Carreau, etc.). La fonction renverra `False` sinon.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2. Écrire une fonction `paire(main)` qui renvoie `True` si et seulement si la main contient deux cartes de même valeur (par exemple deux Valets). La fonction renverra `False` sinon.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

3. Écrire une fonction `suite(main)` qui renvoie `True` si et seulement si la main est constituée de cartes dont les valeurs se suivent (exemple : 8, 9, 10, Valet, Dame). La fonction renverra `False` sinon. On pourra utiliser le dictionnnaire `Val` défini lors de l'exercice 4.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .