N. S. I. Term. - DS03 (jeudi 25 novembre 2021) – v1

Piles-Files – Programmation orientée-objet - Récursivité

1 Récursivité

Exercice 1. [sauf 1/3 temps]

On considère la fonction récursive suivante :

```
def f(n):
    if n < 4:
        print('hello', n)
        f(n + 1)
```

1. Quels affichages obtient-on avec l'appel `f(2)` ?
2. Quels affichages obtient-on avec l'appel `f(5)` ?

Exercice 2.

On considère la fonction récursive suivante :

```
def f(L, i):
    if i == len(L):
        return 0
    else:
        return L[i] + f(L, i + 1)
```

1. Si L = [2, 1, 5], quelle est la valeur de `len(L)` ?
2. Quel(s) est(sont) les cas de base pour la fonction récursive `f` ?
(2bis) : En fonction de la liste *L* passée en argument, quelles valeurs peut prendre le paramètre *i* ? (et quelles genre de listes *L* peut-on accepter en entrée ?)
3. Que renvoie l'appel `f([2, 1, 5], 3)` ?
4. Que renvoie l'appel `f([2, 1, 5], 2)` ?
5. Que renvoie l'appel `f([2, 1, 5], 0)` ?
(Q6) : que fait la fonction `f` ? que renvoie l'appel `f([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 0)` ?

Exercice 3.

On considère la suite $(u_n)_{n\geq 0}$ suivante, définie par $u_0 = 1$, et

$$(*) \text{ pour tout } n \geq 0,\ u_{n+1} = 1 + 2u_n.$$

1. Que valent les quatre premiers termes de la suite $(u_n)_{n\geq 0}$? (donner les calculs à effectuer).

On veut écrire une fonction **récursive**, `u`, de paramètre n, telle que, si n est un entier positif, `u(n)` renvoie la valeur du terme de rang n de la suite $(u_n)_{n\geq 0}$.

2. Pour cette fonction récursive, quelle(s) valeur(s) de n correspondra(ont) au(x) cas de base ?
3. Si on sait que $u_8 = 511$, que vaut u_9 ?
4. D'après la formule $(*)$, quelle valeur suffit-il connaître pour calculer u_{13} ? Quelle valeur suffit-il connaître pour calculer u_n ?
5. Écrire la fonction récursive `u`.

2 Programmation orientée-objet

Exercice 4.

On veut définir une classe implémentant le type Pile, dans laquelle une pile est représentée par un tableau Python.

1. Définir le constructeur, `__init__` , de la classe `Pile`.

2. Définir une fonction `creer_pile_vide()`, à l'aide de la classe `Pile`. La fonction renvoie une pile vide.

3. Combien de méthodes doit-on définir pour la classe `Pile` pour implémenter le modèle théorique de pile ? Que font chacune d'elles ?

4. Implémenter ces trois méthodes.

On suppose que l'on a défini une classe File, munie des méthodes `.est_vide()`, `.enfiler(x)`, `.defiler()`.

Répondre aux questions suivantes, sachant que les files sont implémentées par des objets de la classe File, et en utilisant uniquement les méthodes de la classe File.

5. Par quelles instructions peut-on créer la file `f0`, dont les éléments sont 6, 7 et 8 (entrés dans la file dans cet ordre).

6. Par quelles instructions peut-on créer la file `f1`, dont les éléments sont les entiers 1 à 100, enfilés du plus petit au plus grand ?

7. Écrire une fonction `vider(f)`, permettant de vider une file f, en affichant chaque élément défilé.

Exercice 5.

*Pour chacune les questions **1.** et **3.** un constructeur initialisant les attributs des objets devra être défini.*

1. Définir en Python une classe nommée Carre, dont les objets, représentant des carrés, ont pour attribut la mesure du côté d'un carré, et munie de deux méthodes :
- une méthode `perimetre`, permettant de retourner le périmètre du carré ;
- une méthode `aire`, permettant de retourner son aire.

2. Définir alors un carré de côté 5, puis afficher son périmètre et son aire.

3. **[Sauf 1/3 temps]** Définir en Python une classe nommée Triangle, dont les objets, représentant des triangles, ont pour attributs les mesures des trois côtés d'un triangle, et munie d~~e trois~~ une méthode~~s~~ :
- ~~une méthode `perimetre`, permettant de retourner le périmètre du triangle ;~~
- ~~une méthode `aire`, permettant de retourner son aire, calculée à l'aide de la méthode de Héron, par la formule~~

$$\mathcal{A} = \sqrt{p(p-a)(p-b)(p-c)}, \quad \text{où} \quad p = \frac{a+b+c}{2}.$$

- une méthode `est_rectangle`, renvoyant `True` si le triangle est rectangle et `False` sinon.

a. Définir alors un triangle, nommé `t1`, de côtés 5,4 ; 9 et 7,2~~, et affichant son périmètre, son aire~~

b. Écrire l'appel à la méthode rectangle permettant de savoir si ce triangle est rectangle ou non.

Exercice 6.

Définir en Python une classe, `Temps`, qui permet de définir ~~un horaire~~ une durée au format « hh : mm : ss », dont les objets ont pour attribut un nombre d'heures, un nombre de minutes et un nombre de secondes, et qui dispose des méthodes suivantes :

- `affiche`, qui affiche ~~l'horaire~~ la durée dans un format du type « 12 h 37 min 45 s » ;
- `__add__`, qui renvoie une durée (instance de la classe `Temps`) ~~calcule~~ qui est la somme de deux durées ~~horaires~~ de la classe `Temps` ;
- ~~`__sub__`, qui calcule la différence de deux horaires de la classe `Temps` (si la différence est négative, les nombres d'heures, de minutes et de secondes seront des entiers négatifs).~~

On devra obtenir, par exemple, si t1 correspond à « 9 : 05 : 10 » et t2 à « 11 : 02 : 20 » :

```
>>> t1.affiche()
'9 h 5 min 10 s'
>>> (t1 + t2).affiche()
'20 h 7 min 30 s'
>>> (t1 - t2).affiche()
'-1 h -57 min -10 s'
```

Exercice 7.

On considère la classe `mystere`, définie de la façon suivante :

```
class mystere:
    def __init__(self):
        self.Liste = []

    def add(self, x):
        if len(self.Liste) == 0 or x >= self.Liste[len(self.Liste)-1]:
            self.Liste.append(x)
        elif x < self.Liste[0]:
            self.Liste = [x] + self.Liste
        else:
            for i in range(len(self.Liste)):
                if x > self.Liste[i] and x <= self.Liste[i + 1]:
                    self.Liste = self.Liste[:i+1] + [x] + self.Liste[i+1:]
                break
```

On rappelle que l'instruction `break` (dont l'**utilisation est déconseillée en Python**) permet de terminer prématurément une boucle (comme le fait une instruction `return`, lorsqu'elle est exécutée dans une boucle dans une fonction).

1. Que fait le programme suivant ? (répondre en une ou deux lignes – on détaillera en question **2.**)

```
L = mystere()

for i in [78, 89, 10, 50, 7]:
    L.add(i)

print(L.Liste)
```

2. Détailler ensuite :

- le rôle de la première instruction ;
- l'état de `L` à la fin de la première itération de la boucle `for` ;
- l'état de `L` au début et à la fin de la troisième itération de la boucle `for` ;
- de quel algorithme connu se rapproche ce que réalise ici l'instruction `L.add(i)` à chaque itération ?

3. [sauf 1/3 temps] Ajouter à la classe `mystere` une méthode `ecc`, qui crée et renvoie une liste dont chaque élément est la somme de cet élément et de tous ceux qui le précèdent.
Par exemple, si l'attribut `Liste` de l'objet vaut `[1, 2, 4, 5]`, l'appel à la méthode `ecc` doit renvoyer un objet dont l'attribut `Liste` a pour valeur `[1, 3, 7, 12]`.