Sujet n°4

Algorithmique & Programmation

Structures de données simples

Note : *Le but de ce sujet est de travailler sur des structures de données. Pour créer des structures de données particulières, le langage* ***Python*** *utilise des notions de la programmation objet : les* ***classes****. L'objectif de ce cours n'étant pas d'apprendre l'objet, nous nous contenterons d'une description très sommaire de la notion de* ***classe****, juste suffisante à la réalisation de ces TP.*

1 Minimum vital sur une classe

Jusqu'à présent, vous avez vu qu'il est possible de stocker des valeurs en mémoire en utilisant des variables. Chaque variable possède un type qui décrit les domaines de la valeur affectée à la variable (entier, flottant, ...) et une valeur. Par exemple une variable qui stocke un entier est de type *entier* et peut stocker une donnée entière.

La notion de classe permet de créer de nouveaux types de variables complexes. Ces types sont dits complexes car ils peuvent contenir plus d'une valeur de plusieurs types différents. Nous pouvons voir une classe comme un type regroupant plusieurs variables, appelées dans ce cas *attributs*. Par exemple, nous pouvons définir une classe `Point` comme le regroupement de deux variables entières `x` et `y`. Il sera alors possible de créer une variable (par exemple `p`) de type `Point` ; cette variable sera capable de contenir deux valeurs `x` et `y`. Le type de la variable est donc la *classe* `Point` et l'*instance* de la variable, les valeurs affectées aux *attributs* `x` et `y` de la variable `p`.

En *Python*, nous définissons une classe à l'aide de la syntaxe suivante :

```
class Point :
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

FIGURE 1 – Exeple de création d'une classe ne *Python*.

On remarque que dans la déclaration de la classe se trouve la déclaration d'une fonction `__init__`. Cette fonction est appelée automatiquement à la création d'une variable `Point` et permet d'initialiser les attributs de la classe. Le paramètre `self` est un paramètre caché et est géré automatiquement par Python qui permet de référencer directement la nouvelle instance de la classe créé. Lss autres paramètres sont à la discrétion du programmeur et peuvent être utilsiés pour initialiser des *attributs* (ce qui est le cas dans l'exemple donné).

Il est alors possible de créer une variable `p` contenant une *instance* de la *classe* `Point` et d'utiliser d'accéder aux données stockées dans la variable comme indiqué dans la figure 2.

Notez que chaque variable `p1` et `p2` possèdent leur propre version des *attributs* `x` et `y` et que modifier les *attributs* de `p1` n'affectent en rien les *attributs* de `p2`.

Enfin, les variables stockant des instances de classes ne stockent en réalité que des *références* (sorte de lien) vers ces classes. Une affectation d'une *instance* dans une variable ne fera donc que recopier la référence et non les données contenus dans la classe. L'exemple suivant illustre ce principe : en affectant

```
p1 = Point(1,2)
p2 = Point(3,4)
print(p1.x," - ",p1.y)
print(p2.x," - ",p2.y)
p1.x=10
print(p1.x)
```

FIGURE 2 – Exemple d'instanciation d'une classe

à une variable `p2` la variable `p3`, une modification d'un *attribut* depuis la variable `p2` est aussi "visible" depuis la variable `p3`. L'inverse est évidemment également vrai.

```
p3=p2
print("p2 : ",p2.x," - ",p2.y)
print("p3 : ",p3.x," - ",p3.y)
p2.x=6
print("p2 : ",p2.x," - ",p2.y)
print("p3 : ",p3.x," - ",p3.y)
```

FIGURE 3 – Illustration d'une référence sur une instance de classe.

2 Exercices sur les structures de données simples

Exercice 1 : *Manipulation de tableau*

Créez un tableau à partir d'une liste *Python*. Pour *simuler* le comportement d'un tableau, vous n'aurez le droit d'utiliser dans cet exercice que la fonction *append* de la liste et les accès aux éléments de la liste par la syntaxe `[]`.

QUESTION 1. Écrire une fonction prenant en paramètre un tableau et insérant un élément à la position indiqué dans un deuxième paramètre. Combien d'opérations sont nécessaires dans le meilleur des cas, dans le pire des cas ?

QUESTION 2. Écrire une fonction concaténant deux tableaux donnés en paramètres. Les données du deuxième tableau seront ajoutées à la fin du premier tableau. Combien d'opérations sont nécessaires dans le meilleur des cas, dans le pire des cas ?

QUESTION 3. Écrire une fonction recherchant une valeur donnée en paramètre dans un tableau également donné en paramètre. Combien d'opérations sont nécessaires dans le meilleur des cas, dans le pire des cas ?

Exercice 2 : *Liste chainée*

Dans cet exercice, vous allez créer à l'aide des classes les structures de données nécessaire à la gestion d'une liste.

QUESTION 1. Créer les structure de données nécessaires au stockage des données d'une liste doublement chainée.

QUESTION 2. Écrire une fonction prenant en paramètre une liste et insérant un élément à la position indiqué dans un deuxième paramètre. Combien d'opérations sont nécessaires dans le meilleur des cas, dans le pire des cas ?

Question 3. Écrire une fonction concaténant deux listes données en paramètres. Les données de la deuxième liste seront ajoutées à la fin de la première liste. Combien d'opérations sont nécessaires dans le meilleur des cas, dans le pire des cas ?

Question 4. Écrire une fonction recherchant une valeur donnée en paramètre dans une liste également donnée en paramètre. Combien d'opérations sont nécessaires dans le meilleur des cas, dans le pire des cas ?

Question 5. Comparez les résultats des trois dernières questions avec celles obtenues dans l'exercice 1.

Exercice 3 : *Pile*

Dans une expression arithmétique au format préfixé, on trouve d'abord l'opérateur $+$, $-$, $*$, $/$, puis les opérandes. Par exemple, l'expression $10 + 2$ sera écrite :

```
+ 10 2
```

Les opérandes peuvent être eux mêmes des expressions. Par exemple, l'expression $(10 + 2)/3$ sera écrite :

$$/ + 10\ 2\ 3$$

Question 1. Quelle est l'expression correspondant à l'écriture préfixée

```
+ - 4 * 3 2 / 5 2
```

Question 2. Écrire sous forme préfixée l'expression suivante

$$((3 - 4)/(6 - 12)) * ((7 + 2)/6)$$

Nous allons implémenter une méthode permettant d'évaluer la valeur d'une expression préfixée entrée sous forme de chaîne de caractère, c'est à dire de calculer sa valeur. Pour cela, nous utiliserons une pile, appelée *pile de résultats*.

Nous parcourons la chaîne contenant l'expression préfixée de droite à gauche. Lorsque nous rencontrons un nombre, nous l'empilons. Lorsque nous rencontrons un opérateur, nous dépilons deux nombres, nous effectuons l'opération, et nous empilons le résultat. A la fin, la pile ne contient plus qu'un nombre : c'est le résultat de l'évaluation à retourner.

Question 3. Observer le comportement de l'algorithme sur les exemples de la question 1. Constater que ça marche.

Question 4. Implémenter les primitives de gestion des piles en utilisant une liste *Pyhton* : `Push(list, value)`, `Pop(list)`, `Top(list)`, `IsEmpty(list)`.

Question 5. Implémenter une fonction d'évaluation d'une expression préfixée. On suppose que l'expression est donnée dans une chaîne de caractère et que tous les termes de l'expression sont séparés par des espaces. On lira l'expression de droite à gauche et on reconstituera les nombres à plusieurs chiffres.