

TP N°4 : Programmation Orientée Objet

Exercice 1

1. Définir une classe Python nommé « *Time* » définie par les variables d'instances : *heures*, *minutes* et *secondes*.
2. Créer un objet *instant* de type *Time*.
3. Créer une **fonction** *affiche_heure* qui serve à visualiser les attributs d'un objet de la classe *Time*. Tester cette fonction sur l'objet *instant*. Exemple : `affiche_heure(instant)` → (12 : 32 : 45).
4. Créer une **méthode** *affiche_heure*, à l'intérieur de la classe *Time*. Tester la nouvelle méthode sur l'objet *instant*.

Exercice 2

L'objectif de cet exercice est la manipulation des polynômes creux à une seule variable. Un polynôme creux est un polynôme dont certains coefficients sont nuls. Un polynôme est construit à partir de monômes. Un monôme est une expression de la forme ax^n où a ($a \neq 0$) est le coefficient du monôme et n ($n > 0$) son degré.

Un monôme est représenté par un dictionnaire à un élément dont la clé est le degré n et la valeur est le coefficient a .

Exemple : Le monôme $8x^2$ est représenté par le dictionnaire `{2 : 8}`.

Un polynôme creux est alors défini comme une association de monômes de degrés différents.

Exemples :

Le polynôme $-x^4 + 8x^2 - 5x$ est représenté par le dictionnaire `{2 : 8, 1 : -5, 4 : -1}`.

Le dictionnaire `{0 : 1, 5 : 1, 8 : 1}` représente le polynôme $x^8 + x^5 + 1$.

On se propose de construire la classe *PolynomeCreux* à coefficients réels dont le squelette (à compléter) est défini par :

```
class PolynomeCreux :
    """Manipulation des polynômes creux à une seule variable """
    def __init__(self):
        self.data = {} #initialisation à un polynôme nul
    def ajout_monome(self, monome ={}):
        """
        Cette méthode ajoute un monôme saisi au clavier si le paramètre
        monome est nul ou ajoute le monôme nommé sinon
        """
        if len(monome) == 0 :
            #réponse à la question 1
        else : #si monome est non vide
            degre = list(monome.keys())[0] # extraction du degré
            coeff = list(monome.values())[0] # extraction du coefficient
            try :
                assert degre >= 0
                assert type(degre) == int
                assert type(coeff) == int or type(coeff) == float
                assert len(monome) == 1
                self.data.update(monome) # self.data[degre] = coeff
            except :
                print ("Erreur d'ajout monome")
    def degree(self):
        #réponse à la question 2
    def __call__(self, x0):
        #réponse à la question 3
    def __add__(self, other):
        #other est un polynôme creux
        #réponse à la question 4
    def __mul__(self, other):
        #other est un polynôme creux
        #réponse à la question 5
```

```
def __str__(self):      #réponse à la question 6
def primitive(self):    #réponse à la question 7
```

Travail demandé :

1. Compléter le script de la méthode **ajout_monome**. On rappelle que cette méthode ajoute un monôme saisi au clavier (en faisant les contrôles nécessaires) si le paramètre monome est nul ou ajoute le monôme nommé *monome* sinon.
2. Écrire le script de la méthode, nommée **degree**, qui retourne le degré du polynôme.
3. Écrire le script de la méthode, nommée **__call__** qui retourne la valeur du polynôme pour un réel x_0 donné.
4. Écrire le script de la méthode, nommée **__add__**, qui retourne le polynôme somme de deux polynômes. Remarque : aucun monôme nul ne doit apparaître dans le polynôme résultat.
5. Écrire le script de la méthode, nommée **__mul__**, qui retourne le polynôme produit de deux polynômes. Remarque : aucun monôme nul ne doit apparaître dans le polynôme résultat.
6. Écrire le script de la méthode, nommée **__str__**, qui retourne la chaîne représentant l'expression du polynôme ordonné par ordre décroissant.
Pour le polynôme représenté par $\{4 : 4, 0 : 4, 12 : 6, 9 : 1, 7 : -1\}$, la chaîne retournée est : "6*x**12 + x**9 - x**7 + 4*x**4 + 4"
7. Écrire le script de la méthode, nommée **primitive**, qui retourne le polynôme représentant la primitive. On suppose que la constante d'intégration est nulle.
8. On définit, l'intégrale d'un polynôme creux P en x entre les bornes a et b , par : $S = \int_a^b P dx$.
Écrire le script de la fonction, nommée **integrale**, permettant de retourner la valeur de S à partir d'un polynôme P , de type PolynomeCreux, et des bornes d'intégration a et b réels.

Exercice 3

1. Créer une classe nommée **forme_géométrique** ayant comme attribut le *nombre de côtés* de la forme ainsi que son *nom*.
2. Un **rectangle** est une forme géométrique ayant 4 cotés et se nomme « rectangle ». Un rectangle a également une longueur et une largeur.
Créer la classe rectangle et définir les méthodes suivantes :
 - (a) Le *constructeur* de la classe rectangle.
 - (b) Une méthode *périmètre* permettant de calculer et retourner la valeur de son périmètre.
 - (c) Une méthode *mesures* permettant d'afficher ses mesures.
3. Dérivez de la classe rectangle la classe **carré**. Implémenter la méthode **__repr__** de la classe carré.

Exercice 4

1. Créer une classe **Personne** définie par deux attributs d'instances **nom** et **prénom**.
2. Un **employé** est une personne physique qui occupe un **poste** dans une entreprise et ayant un **salaire**. Créer une classe Employé dérivée de la classe Personne.
3. Une **entreprise** est une organisation économique qui a pour but de produire des biens et des services destinés au marché. Une entreprise embauche des employés et verse leurs salaires.
 - (a) Créer la classe Entreprise.
 - (b) Créer une méthode permettant de rechercher et retourner un employé par son nom ou son prénom.
 - (c) Créer une méthode permettant de une liste des employés qui occupent par poste passé en paramètre.
 - (d) Créer une méthode permettant de calculer le coût total des salaires.
 - (e) Créer une méthode permettant de trier (sur place) une liste des employés par postes et par salaires.
 - (f) Créer une méthode permettant d'afficher l'employé le plus rémunéré pour un poste particulier.
 - (g) Créer une méthode permettant de tracer sur un graphe l'histogramme représentant la distribution des salaires par tranches. La méthode accepte comme paramètre la valeur de la tranche salariale.