TP de Langages de script n° 5 : Objets.

Python est un langage orienté objet. Une *classe* permet de définir un nouveau type et une *instance* est un objet d'une classe. Les objets peuvent contenir des variables appelées *attributs* de la classe. Les fonctions appartenant à une classe sont appelées *méthodes*. Toutes les méthodes doivent avoir au moins un paramètre (que nous appellons toujours `self`). Quand on appelle la méthode la valeur de `self` est automatiquement l'objet à partir du quel on a appelé la méthode. Les attributs sont typiquement initialisés par des constructeurs `__init__`. Une méthode (ou même toute fonction python) peut avoir des valeurs des paramètres par défaut qui sont prises quand on l'appelle avec moins d'arguments. Ici on initialise l'objet `Point` avec 0 et 0 par défaut. La méthode `__del__` (facultative) est invoquée quand l'objet est détruit (soit explicitement, soit à la fin du programme). Il peut y avoir également des variables de classe (ici `nombre_de_points`) qui sont associées directement à la classe et pas à une instance de la classe.

```
import math
class Point():
    """Une classe Point"""
    nombre_de_points = 0
    def __init__(self,x=0,y=0):
        self.x = x
        self.y = y
        Point.nombre_de_points += 1
    def __del__(self):
        print("le point",self.x,self.y,"est détruit")
        Point.nombre_de_points += -1
    def distancenull(self):
        return math.sqrt(pow(self.x,2)+pow(self.y,2))
    def distance(self,p):
        return math.sqrt(pow(self.x-p.x,2)+pow(self.y-p.y,2))

p = Point(3,4)
print(p.__doc__)
p1 = Point(1,2)
p2 = Point()
print("Il y a",Point.nombre_de_points,"points")
print(p)
print(p.distancenull())
print(p.distance(p1))
print(p1.distance(p))
print(p.distance(p2))
del p
print("Il y a",Point.nombre_de_points,"points")
print(p)
```

Ce programme a comme effet d'imprimer :

```
Une classe Point
Il y a 3 points
<__main__.Point object at 0xb744258c>
5.0
2.8284271247461903
2.8284271247461903
5.0
le point 3 4 est détruit
Il y a 2 points
Traceback (most recent call last):
  File "exclass.py", line 31, in <module>
    print(p)
```

```
NameError: name 'p' is not defined
le point 0 0 est détruit
le point 1 2 est détruit
```

Ici, on remarque que l'objet `p` n'existe plus après sa destruction et que les deux autres objets sont détruits implicitement.

Exercice 1 :

Définir une classe `CompteEpargneTemps()` pour gérer le nombre d'heures travaillées par un employé. Le constructeur de cette classe devrait initialiser deux attributs nom et solde, avec les valeurs par défaut "Michel" et 0. Vous devrez définir trois autres méthodes :

- `ajout` qui permet d'ajouter un certain nombre d'heures d'un compte,
- `enlever` qui permet d'enlever un certain nombre d'heures d'un compte,
- `afficher` qui permet d'afficher le nom de l'employé et le solde de son compte epargne temps,
- `nombre` qui retourne le nombre d'employés existants.

Définir ensuite une classe `CompteBonus` qui en plus des fonctionnalités du `CompteEpargneTemps` permet de gérer des bonus : a chaque compte est associé un bonus qui peut être modifié par une méthode dédiée.

Exercice 2 :

Gestion d'un annuaire On souhaite écrire un module contenant des outils pour manipuler des annuaires. Ces annuaires seront stockés dans des fichiers textes au format CSV (Comma Separated Values) : chaque ligne représentera une entrée de l'annuaire, avec 5 champs séparés par des virgules, représentant les 5 caractéristiques du personnage concerné : nom, prénom, profession, adresse et numéro de téléphone. Ainsi, par exemple :

```
Haddock,Archibald,Capitaine,Chateau de Moulinsart,421
Derkozy,Niclas,Marin,Champs-Elaises,234
Odema,Darak,Matelot,Maison Grise,555
```

Ce module devra définir deux classes, `Annuaire` et `Personnage`. Chaque instance de la classe `Annuaire` contiendra une liste d'instances de la classe `Personnage`, qui elles-mêmes auront pour attributs les 5 caractéristiques précédemment citées.

1. Écrire un constructeur pour la classe `Personnage`, prenant en paramètre une chaîne de caractères au format CSV (Rappel : utilisez `split`)
2. Il est possible de définir une méthode spéciale pour la représentation d'un objet par une chaîne de caractères, `__str__`. Lorsque cette méthode est définie, `print` utilise `__str__`. Définir cette méthode pour permettre un affichage agréable pour l'utilisateur.
3. Écrire un constructeur pour la classe `Annuaire`, prenant en paramètre un nom de fichier. Écrire une méthode `__str__` pour `Annuaire` qui retourne un annuaire trié en forme de chaîne de caractères.
4. Écrire une méthode `copie` prenant en paramètre un nom de fichier et y copiant les représentations des entrées de l'annuaire au format CSV, triées par ordre alphabétique. Pour cela, on pourra créer un dictionnaire et en parcourir la liste des clés triée.
5. Écrire une méthode `liste_pages_jaunes` prenant en paramètre une chaîne de caractères et renvoyant la liste des entrées de l'annuaire ayant la profession correspondante. Écrire une méthode `cherche_pages_jaunes` affichant le nombre et la liste des réponses obtenues.
6. Écrire une méthode `liste_pages_blanches` prenant en paramètre une chaîne de caractères et retournant la liste des personnes dont le nom ou le prénom contient cette chaîne. Écrire la méthode `cherche_pages_blanches` correspondante.
7. Écrire une méthode `modifier` prenant un nom en paramètre et demandant à l'utilisateur d'entrer les nouvelles coordonnées de la personne correspondante. Pour cela, on pourra écrire une méthode `sélectionne` qui teste si le nom apparaît dans l'annuaire et, s'il apparaît plusieurs fois, demande à l'utilisateur de choisir parmi les personnes correspondantes.
8. Ajouter des méthodes `ajouter` et `supprimer`.
9. Écrire une interface textuelle simple permettant à un utilisateur de manipuler l'annuaire.