DS – Term Nsi

POO et RECURSIVITE : Partie A

Exercice 1. : POO : classe *Musicien*

On donne ci-dessous le script incomplet d'une classe nommée *Musicien* permettant d'encapsuler des caractéristiques d'un musicien appartenant à une formation musicale.

```
class Musicien :
    nbr_musiciens = 0
    def __init__(self,nom,prenom,instrument,surnom = None,age = None) :
        self.nom = nom
        self.prenom = prenom
        self.instrument = instrument
        self.surnom = surnom
        self.age = age
        self.__class__.nbr_musiciens +=1


    def __str__(self) :
        if self.surnom != None :
            chaine = f"  - {self.surnom}, {self.instrument}, de son vrai nom :
{self.nom} {self.prenom} "
        else :
            chaine = f"  - {self.nom} {self.prenom}, {self.instrument} "
        return chaine

    def setAge(self,age) :
        self.age = age
        return self.age

    def getAge(self) :
        return self.age
```

```
m1 = Musicien("Cotentin","Aurélien","chant", "Orelsan",40)
m2 = Musicien("Le Carpentier","Matthieu","beatmaker", "Skread",41)
m3 = Musicien("Preau ","Adam","beatmaker","Phazz",31)
m4 = Musicien("Ardan","Edouard","guitariste" , "Eddie Purple")
m5 = Musicien("Dyens","Manu","batteur")
print(m1)
print(m5)
print(m5.setAge(25))
print(m5.getAge())
print(Musicien.nbr_musiciens)
```

L'exécution des lignes suivantes :

permet alors d'afficher dans la console :

```
>>> (executing file "classMusuciens.py")
  - Orelsan, chant, de son vrai nom : Cotentin Aurélien
  - Dyens Manu, batteur
25
25
5
```

1- Compléter le code de la méthode constructeur
2- Compléter le code de la méthode __str__()
3- Compléter le code de la méthode setAge()
4- Compléter le script de la méthode getAge()
5- En poo, quel nom qualificatif donne-t-on aux variables nom, prenom, instrument, surnom et age ?
Ces variables sont appelées attributs d'instance.
6- En poo, quel nom qualificatif donne-t-on à la variable nbr_musiciens ?
Cette variable est appelée attribut de classe.

Exercice 2. : Récursivité

On considère le programme suivant :

```
def f(a,b) :
    if b == 1 :
        return a
    return a + f(a,b-1)

val = f(7,9)
print(val)
```

1- Exécuter ci-dessous ce code « *à la main* »

$$f(7,9)$$
$$7 + f(7,8)$$
$$7 + 7 + f(7,7)$$
$$7 + 7 + 7 + f(7,6)$$
$$7 + 7 + 7 + 7 + f(7,5)$$
$$7 + 7 + 7 + 7 + 7 + f(7,4)$$
$$7 + 7 + 7 + 7 + 7 + 7 + f(7,3)$$
$$7 + 7 + 7 + 7 + 7 + 7 + 7 + f(7,2)$$
$$7 + 7 + 7 + 7 + 7 + 7 + 7 + 7 + f(7,1)$$
$$7 + 7 + 7 + 7 + 7 + 7 + 7 + 7 + 7 = 7 \times 9 = 63$$

2- Donner ci-dessous une version **non récursive** de ce même code :

```
def F(a,b) :
    return a*b
```

Exercice 3. : Récursivité

On considère le programme suivant :

```
def f(n) :
    if n < 2 :
        return 1
    else :
        return f(n-1) + f(n-2)

val = f(7)
print(val)
```

1- Exécuter ci-dessous ce code « *à la main* »

$f(7)$ $f(6) + f(5)$ $f(5) + f(4) + f(4) + f(3)$ $f(4) + f(3) + f(3) + f(2) + f(3) + f(2) + f(2) + f(1)$ $f(3) + f(2) + 3 \times (f(2) + f(1)) + 3 \times (f(1) + f(0)) + 1$	$4 \times (f(2) + f(1)) + 4 \times (f(1) + f(0)) + 1$ $4 \times (f(1) + f(0) + 1) + 4 \times (1 + 1) + 1$ $4 \times (1 + 1 + 1) + 4 \times (1 + 1) + 1$ $12 + 8 + 1$ 21

DS – Term Nsi

POO et RECURSIVITE : Partie B

⇨ Se loguer avec l'identifiant : **exam02.eleve** Mot de passe :

Le code à réaliser sera appelé *ds_mon_nom.py* . Il est à déposer en fin d'épreuves dans le répertoire : ***Examens(Z :) /exam02/copies/***

⇨ **Copier** le dossier *NSI-15nov2022* : ***Examens(Z :) /exam04/sujets/NSI-15nov2022*** sur le disque dur de l'ordinateur (sur le *bureau* ou dans le répertoire *Mes documents*) et le décompresser.

Exercice 4. : POO : classes *Musicien* et *Groupe*

⇨ Ouvrir le fichier *groupeEnonce.py* . Le renommer avec le nom *ds_mon_nom.py.*

Sont définies dans ce fichier 2 classes. La classe *Musicien* étudiée dans la partie A et la classe *Groupe* qui permet de définir les musiciens qui appartiennent à un groupe musical.

⇨ Compléter le code de la classe *Groupe* et celui du dernier *print()* ci-dessous, afin que l'exécution des lignes suivantes :

```
#Main
g = Groupe("Orelsan")
g.setMembres("Cotentin","Aurélien","chant", "Orelsan",40)
g.setMembres("Le Carpentier","Matthieu","beatmaker", "Skread",41)
g.setMembres("Preau","Adam","beatmaker","Phazz",31)
g.setMembres("Ardan","Edouard","guitariste" , "Eddie Purple")
g.setMembres("Dyens","Manu","batteur")
print(g)
moy = g.moyenneAge()
print(moy)
b = Groupe("Beatles")
print(f"Il y a ..... instances de la classe Groupe créées")
```

donne précisément dans la console :

```
>>> (executing file "musique.py")
Composition du groupe d'Orelsan : 5 membres
  - Orelsan, chant, de son vrai nom : Cotentin Aurélien
  - Skread, beatmaker, de son vrai nom : Le Carpentier Matthieu
  - Phazz, beatmaker, de son vrai nom : Preau Adam
  - Eddie Purple, guitariste, de son vrai nom : Ardan Edouard
  - Dyens Manu, batteur

37.3
Il y a 2 instances de la classe Groupe créées
```

```
class Groupe :
    nbr_groupes = 0
    def __init__(self,nom) :
        self.nom = nom
        self.membres = []
        Groupe.nbr_groupes += 1

    def setMembres(self,nom,prenom,instrument,surnom = None,age = None) :
        obj = Musicien(nom,prenom,instrument,surnom,age)
        self.membres.append(obj)

    def moyenneAge(self) :
        s = 0
        n = 0
        for obj in self.membres :
            if obj.age != None :
                s += obj.age
                n += 1
        return round(s/n,1)

    def __str__(self):
        out = f"Composition du groupe {self.nom} : {len(self.membres)} membres \n"
        for obj in self.membres :
            out += obj.__str__()+"\n"
        return out

g = Groupe("Orelsan")
g.setMembres("Cotentin","Aurélien","chant", "Orelsan",40)
g.setMembres("Le Carpentier","Matthieu","beatmaker", "Skread",41)
g.setMembres("Preau","Adam","beatmaker","Phazz",31)
g.setMembres("Ardan","Edouard","guitariste" , "Eddie Purple")
g.setMembres("Dyens","Manu","batteur")
print(g)
moy = g.moyenneAge()
print(moy)
b = Groupe("Beatles")
print(f"Il y a {Groupe.nbr_groupes} instances de la classe Groupe créées")
```

Autre solution pour la méthode __str__():

```
def __str__(self):
    m=len(self.membres)
    out=f"Composition du groupe {self.nom} : {m} membres \n"
    for i in range(m):
        out += (f"{self.membres[i]} \n ")
    return out
```

Exercice 5. : Récursivité

On définit la suite de nombre suivante : 3 ⇨ 6 ⇨ 12 ⇨ 24 ⇨ 48 ⇨ 96 ⇨ 192 ⇨ ….

Le premier de ces nombres est donc $u_1 = 3$. On a ensuite $u_2 = 6$, $u_3 = 12$, …., $u_7 = 192$

On a globalement : $u_n = 2\,u_{n-1}$ avec $u_1 = 3$.

Dans le fichier .py complété précédemment :

1- Coder une fonction *iter*(n) qui renvoie le n^{ième} terme en utilisant un algorithme itératif

2- Coder une fonction *rec*(n) qui renvoie le n^{ième} terme en utilisant un algorithme récursif

```python
def iter(n) :
    u = 3
    for i in range(2,n+1) :
        u = u * 2
    return u

def rec(n) :
    if n == 1 : return 3
    return 2*recur(n-1)

print(iter(7))
print(rec(7))
```

Exercice 6. : Récursivité

La fonction *compteur(l , val)* renvoie le nombre d'éléments de la liste numérique *l* qui sont égaux à la variable *val*.
L'exécution des lignes donnera dans la console :

```python
nbr = compteur([1,7,-8,9,1,20],1)
print(nbr)
print(compteur([1,7,-8,9,1,20],40))
```

```
>>> (executing file "compteur.py")
2
0
```

Dans le fichier .py complété précédemment :

1- Coder une version itérative de cette fonction. On la nommera *compteurIT()* .

```python
def compteurIT(l,num) :
    n = 0
    for elt in l :
        if elt == num : n += 1
    return n
```

2- Coder une version récursive de cette fonction. On la nommera *compteur().* On utilisera la méthode *pop()* qui supprime le dernier élément d'une liste et renvoie sa valeur. Par exemple :

```
>>> l = [1,7,-8,9,1,20]

>>> l.pop()
20

>>> l
[1, 7, -8, 9, 1]
```

```python
def compteur(l,num) :
    if len(l) == 0 : return 0
    else :
        val = l.pop()
        if val == num : return 1 + compteur(l,num)
        else : return 0 + compteur(l,num)
```

Autres solutions :

```python
def compteurRC(l:list,ind:int,c):
    if len(l)==0:
        return c
    else:
        if l[-1]==ind:
            c+=1
        l.pop()
        return compteurRC(l,ind,c)
```