INF1 – P22

Examen Final - Corrigé

1ère partie : gestion d'associations *(5 points)*

On souhaite représenter une liste des associations de l'UTC. Une association est représentée par un nom, une liste de mots-clés et une liste de membres. Par exemple, l'association "Le coin du joueur a :

- pour mots clés : " *jeux*", "*société*"
- comme membres : "*Donatello*", "*Michelangelo*"

Dans la liste des associations on aura par exemple "*Le coin du joueur*", "*Japon UTC*", etc.

1. Proposez une représentation de la liste de ces associations en utilisant les dictionnaires et les tableaux Python.
2. Écrire une fonction Python `createAsso()` qui permet à un utilisateur de saisir le nom de l'association et d'ajouter des mots clés et des membres. Cette fonction renvoie un dictionnaire.
3. Écrire une fonction Python `createAssos(assos)` qui permet à un utilisateur d'ajouter des associations à une liste `assos` d'associations passée en paramètre. Pensez à utiliser la fonction `createAsso()` proposée à la question précédente.
4. Écrire une fonction Python `searchMemberAssos(assos)` qui prend en entrée la liste `assos` des associations, demande à l'utilisateur le nom d'un membre et renvoie une liste avec le nom des associations dans lesquelles ce membre est impliqué.

```
def printAsso(asso) :
    print('-----------------------')
    print(f'ASSOCIATION : {asso["NAME"]}')
    print(f'              Mots clés : {asso["KEY WORDS"]}')
    print(f'              Membres   : {asso["MEMBERS"]}')

def printAssos(assos) :
    for asso in assos :
        printAsso(asso)

def createItems(item) :
    bContinue = True
    items = []
    print(f'-- Ajout {item}')
    while bContinue :
        it = input(f'Ajout de {item} : ')
        items.append(it)
        bContinue = input('Souhaitez vous ajouter un {item} (oui/non) : ')
== 'oui'
    return items

def createAsso() :
    asso = {}
```

```
    print('---------CREATION ASSO------------')
    asso['NAME'] = input('Nom de l association : ')
    asso['KEY WORDS'] = createItems('MOTS CLE')
    asso['MEMBERS'] = createItems('MEMBRE')
    return asso

def createAssos(assos) :
    bContinue = True
    while bContinue :
        asso = createAsso()
        assos.append(asso)
        bContinue = input('Souhaitez vous ajouter une association (oui/non)
: ') == 'oui'
    return assos

def searchMemberAssos(assos) :
    lMemberAssos = []
    print('---------RECHERCHER LES ASSOS D UN MEMBRE ------------')
    member = input('Quel nom recherchez vous : ')
    for asso in assos :
        if member in asso['MEMBERS'] :
            lMemberAssos.append(asso['NAME'])
    return lMemberAssos

def menu() :
    bContinue = True
    assos = []
    while bContinue :
        print('************* MENU *****************')
        print('1. Ajouter des associations')
        print('2. Ajouter une asso')
        print('3. Trouver les assos d un membre')
        print('4. Afficher toutes les associations')
        print('0. Pour terminer')
        choix = int(input('Votre choix : '))
        if choix == 0 :
            bContinue = False
        elif choix == 1 :
            assos = createAssos(assos)
        elif choix == 2 :
            assos.append(createAsso())
        elif choix == 3 :
            print(searchMemberAssos(assos))
        elif choix == 4 :
            printAssos(assos)
        else:
            print('Choix incorrect')

menu()
```

---------- *{ prendre une nouvelle copie }* ----------

2ème partie : somme de la somme de la… des chiffres *(5 points)*

Soit un entier `n ≥ 1`. On note s(n) le nombre obtenu en faisant la somme des chiffres de `n`, puis la somme des chiffres de ce nombre et en répétant ce processus jusqu'à ce que le résultat soit un entier à seul chiffre.

Par exemple, si `n = 49538` alors la somme des chiffres de n est 4+9+5+3+8=29. Comme cette somme a deux chiffres, on recommence et on obtient 2 + 9 = 11 et comme ce résultat est encore à deux chiffres, on continue pour obtenir finalement s(n) = 2.

1. Écrire une fonction Python `sommeChiffres(n)` permettant de calculer la somme des chiffres d'un entier `n`.

   Indice : on pourra convertir `n` en une chaine de caractères et faire appel aux fonctions prédéfinies de Python : `str(…) et int(…)`

```
def sommeChiffres(n):
    somme=0
    ch=str(n)
    for i in range(0,len(ch)):
        somme=somme+int(ch[i])
    return somme
```

ou

```
def sommeChiffres(n):
    somme = 0
    ch = str(n)
    for car in ch:
        somme = somme + int(car)
    return somme
```

2. Écrire l'algorithme ou le programme Python de la fonction **récursive** `s(n)`.

   Indice : on distinguera le cas où la longueur de la chaine contenant la somme des chiffres de n est égale à 1 et le cas où elle est supérieure à 1

Algorithme

```
fonction s(n:entier) : entier
      variables
            somme : entier

      somme ← sommeChiffres(n)
      si longueur(str(somme))= 1 alors
            retourner somme
      sinon
            retourner s(somme)
      fsi
```

Fonction Python

```
def s(n):
    somme = sommeChiffres(n)
    if len(str(somme))==1:
        return somme
    else:
        return s(somme)
```

3. On souhaite maintenant écrire une **fonction récursive** `srec(n, k)`, k ≥ 1 et et n ≥ 1, qui calcule s(n), n etant un entier composé de k blocs de chiffres égaux à ceux de n. Par exemple, `srec(49538, 3)` vaut s(495384953849538).
   Écrire l'algorithme ou le programme Python de la fonction **récursive** srec(n, k).

   Indice : on distinguera le cas où k = 1 qui n'est autre que s(n) et le cas où k > 1.

```
def srec(n,k):
    if k == 1:
        return s(n)
    else:
        return s(s(n)+srec(n,k-1))
```

---------- ***{ prendre une nouvelle copie }*** ----------

3ème partie : combat de Pokémons *(5 points)*

Un Pokémon représente des créatures dotées de pouvoirs extraordinaires, comme la capacité de cracher du feu ou de maîtriser l'électricité, et ils peuvent se faire capturer par les humains. Le nom Pokémon est issu de la contraction de « Poketto Monsutā », [...], traduisant l'anglais « Pocket Monsters »

source: https://fr.wikipedia.org/wiki/Pok%C3%A9mon

Dans cette partie, l'objectif est de réaliser un combat entre 2 Pokémons. à l'aide de classes et d'objets. On considère pour cela 2 classes : Pokemon et Arene

1. Classe Pokemon :
   a. Créer une classe Pokemon, permettant de définir un Pokémon. Un Pokémon est défini par son nom (une chaîne de caractères), une attaque "attaque1" (un entier), une attaque "attaque2" (un entier) et un degré de vie (un entier) initialement à 0.
   b. Ajouter à cette classe:
      - une méthode `initialiser_vie(self)` qui permet d'initialiser la vie du Pokémon à 100,
      - une méthode booléenne `est_en_vie(self)` qui permet de tester si le Pokémon est en vie ou non.

```
class Pokemon:

   def __init__(self,nom,attaque1, attaque2):
       self.nom = nom
```

```
        self.attaque1 = attaque1
        self.attaque2 = attaque2
        self.vie = 0

    def initialiser_vie(self):
        self.vie = 100

    def est_en_vie(self):
        return self.vie > 0
```

2. Classe Arene:

Le but de cette classe est de prendre en compte deux instances de Pokémon et de réaliser un combat entre les deux.

a. Créer une classe Arene. Une arène est définie simplement par 2 instances de Pokémon.

```
class Arene:
    def __init__(self,pokemonA,pokemonB):
        self.pokemonA = pokemonA
        self.pokemonB = pokemonB
```

b. Écrire les méthodes :
   - `appliquer_attaque(self, attaque, pokemon)` : applique l'attaque sur le pokémon, c'est à dire que la vie de ce pokémon doit diminuer en fonction de l'attaque renseignée,
   - `choisir_une_attaque(self, pokemon)` : demande à l'utilisateur de choisir une attaque ("1" ou "2") pour le pokémon donné en paramètre et retourne la valeur de cette attaque.

```
def appliquer_attaque(self,attaque,pokemon):
        pokemon.vie -= attaque

def choisir_une_attaque(self,pokemon):
        choix = ""
        while(choix != "1" and choix != "2"):
            choix = input(f"choisir une attaque pour le pokemon
{pokemon.nom} ")
        if(choix == "1"):
            return pokemon.attaque1
        return pokemon.attaque2
```

c. A partir de ces fonctions, créer une méthode `combattre(self)` permettant de réaliser le combat entre les 2 pokémons tant que les 2 pokémons sont encore en vie.

La méthode doit afficher le Pokémon gagnant à la fin du combat. On distinguera trois étapes :

a. étape 1 : initialiser la vie des pokémons.
b. étape 2 : faire le combat entre les deux pokémons tant que les deux sont encore en vie. Le combat se fait en tour par tour en alternant le pokémon qui attaque et le pokémon qui défend.
c. étape 3 : afficher le nom du pokémon gagnant.

```
def combattre(self):
        self.pokemonA.initialiser_vie()
        self.pokemonB.initialiser_vie()

        pokemon_qui_attaque = self.pokemonA
        pokemon_qui_defend = self.pokemonB
        while(self.pokemonA.est_en_vie() and self.pokemonB.est_en_vie()):

            attaque = self.choisir_une_attaque(pokemon_qui_attaque)
            self.appliquer_attaque(attaque,pokemon_qui_defend)

            print(f"pokemon: {self.pokemonA.nom} vie: {self.pokemonA.vie}")
            print(f"pokemon: {self.pokemonB.nom} vie: {self.pokemonB.vie}")

            pokemon_tmp = pokemon_qui_defend
            pokemon_qui_defend = pokemon_qui_attaque
            pokemon_qui_attaque = pokemon_tmp

        pokemon_gagnant = self.pokemonA
        if(self.pokemonB.est_en_vie()):
            pokemon_gagnant = self.pokemonB

        print(f"victoire pokemon {pokemon_gagnant.nom}")
```

---------- ***{ prendre une nouvelle copie }*** ----------

4ème partie : fusion de fichiers de personnes *(5 points)*

On souhaite écrire un programme en Python qui permet la manipulation d'informations sur des personnes et leur stockage dans des fichiers texte.

1. Chaque personne est identifiée par son nom et son âge, sous la forme d'une chaine de caractères. Le nom et l'âge sont séparés par un ";" (point-virgule).
   Définir une fonction à deux paramètres (nom et âge) qui retourne cette chaine de caractères.

```
def personne (nom,age):
    return nom + ';' + str(age) + '\n'
```

2. Définir une fonction `ouvre_fich(nom_fich, mode)`, `nom_fich` étant le nom externe du fichier créé ou exploité, `mode` étant le mode d'ouverture du fichier (lecture ou écriture) et qui retourne un fichier ouvert.

```
def ouvre_fich (nom_ext, mode):
    if mode == "r":
        try:
            nom_int = open(nom_ext, "r")
            return nom_int
        except IOError:
            print ("Fichier inexistant : ", nom_ext)
    elif mode == "w":
        nom_int = open(nom_ext, "w")
        return nom_int
    else:
        print ("Erreur de mode")
```

3. Écrire une procédure `cree_fich(nom_fich)` permettant la création du fichier `nom_fich` avec des 'personnes' (voir question 1), le nom et l'âge de chaque personne étant saisis au clavier.

```
def cree_fich (nom_ext):
    f = ouvre_fich (nom_ext, 'w')
    suite = 'ok'
    while suite == 'ok':
        nom = input('Nom : ')
        age = input('Age : ')
        f.write(personne(nom,age))
        suite = input('ok pour continuer')
    f.close()
```

4. Écrire un procédure `fusion(nom_fich1, nom_fich2, nom_fich3)` qui fusionne, suivant l'ordre alphabétique des noms de personne, les deux fichiers de personnes `nom_fich1` et `nom_fich2` dans un troisième fichier `nom_fich3` contenant l'ensemble des données. On supposera que les deux fichiers `nom_fich1` et `nom_fich2` sont déjà ordonnés suivant l'ordre alphabétique des noms de personnes.

```
def fusion (nom_ext1, nom_ext2,nom_ext3):
```

```
    f1 = ouvre_fich(nom_ext1, 'r')
    f2 = ouvre_fich(nom_ext2, 'r')
    f3 = ouvre_fich(nom_ext3, 'w')

    s1 = f1.readline()
    s2 = f2.readline()
    while (s1 != "") and (s2 != ""):
        pos1 = s1.index(';')
        pos2 = s2.index(';')
        if s1[:pos1] < s2[:pos2]:
            f3.write(s1)
            s1 = f1.readline()
        else:
            f3.write(s2)
            s2 = f2.readline()
    if s1 == "":
         while s2 != "":
            f3.write(s2)
            s2 = f2.readline()
    else:
        while s1!= "":
            f3.write(s1)
            s1 = f1.readline()
    f3.close()
    f1.close()
    f2.close()
```

5. Écrire le programme principal qui crée deux fichiers "personnes1" et "personnes2" et les fusionne dans un troisième fichier de nom "personnes"

```
cree_fich("personnes1")
cree_fich("personnes2")
fusion ("personnes1", "personnes2", "personnes")
```