TP #3 - Correction

Fonctions et classes

I. Création d'un nouveau projet

Ouvrez l'IDE de votre choix (VSCode ou PyCharm), et initiez un projet vierge (tp3).

Créez un fichier de type python (tp3.py).

II. Fonctions

Exercice 1

Ecrire la fonction nbChiffresDuCarre(n), qui calcule et affiche le nombre de chiffres que comporte l'écriture du carré du nombre n.
Exemple : Le nombre 8 au carré vaut 64 et 64 est composé de 2 chiffres.
Indice : La longueur d'une chaine de caractère peut être obtenue grâce à la fonction len().

```python
# Version courte
def nbChiffresDuCarre(n):
    print(len(str(n * n)))

# Version détaillée
def nbChiffresDuCarre(n):
    # On calcule le carré du nombre n reçu en paramètre
    carre = n * n
    # On caste le nombre entier en chaine de caractère
    string = str(carre)
    # On calcule la longueur de la chaine
    nbChiffres = len(string)
    # On affiche le résultat
    print(nbChiffres)
```

```python
# Appel de la fonction pour tester
nbChiffresDuCarre(100)
```

```
5
```

Exercice 2

Ecrire une fonction estPremier() qui reçoit en paramètre un nombre, et retourne true s'il s'agit d'un nombre premier (divisible que par 1 et lui-même), false sinon.

Indice : Opération modulo (%).

```
def estPremier(n):
    estPremier = True
    for i in range(2, n) :
        if n % i == 0 :
            estPremier = False
    return estPremier
```

```
print(estPremier(7))
```

```
True
```

```
print(estPremier(8))
```

```
False
```

Exercice 3

Nous souhaitons afficher la liste de tous les nombres premiers jusqu'à un nombre n.

Ecrire une fonction qui reçoit ce nombre n en paramètre, et qui, pour l'ensemble des nombres de 0 à n, affiche ceux qui sont premiers (ré-utiliser la fonction précédente).

Pour tester, appelez cette fonction pour n = 10.

```
def nombresPremiers(n):
    for i in range(2, n + 1):
        if estPremier(i) == True:
            print(i)
```

```
nombresPremiers(10)
```

```
2
3
5
7
```

III. Classes

3.1. Gestion des dates

Nous souhaitons créer un nouveau type de donnée : Date
Une date sera composée de 3 entiers :

- Le jour
- Le mois
- L'année

Exercice 4

Créez la classe correspondante.

```
class Date:

    def __init__(self, jour = 1, mois = 1, annee = 1970):
        self.jour = jour
        self.mois = mois
        self.annee = annee
```

Exercice 5

Nous aurons souvent besoin d'afficher des variables de type Date, donc le plus simple est de créer dès maintenant une procédure dans la classe Date qui affiche cette dernière.
Si la date reçue en paramètre était le 10 janvier 2025, alors voici l'affichage que vous devez avoir :
10/1/2025

```
    def afficheDate(self):
        print(str(self.jour) + "/" + str(self.mois) + "/" + str(self.annee))
```

Exercice 6

Nous aurons aussi besoin à de nombreuses reprises de créer des variables de type Date et de les remplir avec des valeurs saisies par l'utilisateur.
Or ces saisies doivent être systématiquement vérifiées pour éviter que les utilisateurs n'entrent des valeurs incohérentes ou des dates impossibles. Pour ne pas avoir à coder plusieurs fois cette demande de saisie et ces vérifications vous allez créer une fonction qui ne reçoit aucun paramètre et qui retourne un objet Date correctement remplit par l'utilisateur.

La fonction créera l'objet vide, demandera à l'utilisateur l'année, puis le mois et enfin le jour de la date à créer. Chaque saisie sera vérifiée puis, si elle est correcte, sera enregistrée dans la variable de type Date. Lorsque les 3 composantes de la date (année, mois, jour) seront enregistrés, la variable sera retournée par la fonction.
Vous choisirez un nom approprié pour cette fonction.
Voici un exemple d'exécution :

```
Entrez l'année : 2025
Entrée le mois : 01
Entrez le jour : 10
10/01/2025
```

Cette fonction aura la responsabilité de vérifier chacune des données saisies pour ne pas que l'utilisateur puisse entrer de Date invalide.

```
def creerDate():
    date = Date();
    annee = 0
    mois = 0
    jour = 0

    while annee < 1900 or annee > 2050 or mois < 1 or mois > 12 or jour < 1 or
jour > 31 :
        annee = int(input("Entrez l'année : "))
        mois = int(input("Entrez le mois : "))
        jour = int(input("Entrez le jour : "))

    date.annee = annee
    date.mois = mois
    date.jour = jour

    date.afficheDate()

    return date;
```

3.2. Gestion des employés

Maintenant que nous avons notre type Date de créé et quelques procédures et fonctions nous permettant de manipuler des variables Date, nous allons pouvoir l'utiliser dans un autre type.
Nous voulons créer un type de donnée permettant de gérer les employés d'une entreprise pour un logiciel de gestion.
Nous allons donc créer un nouveau type de donnée Personne. Chaque Personne aura :

- Un nom
- Un prénom
- Une date de naissance
- Une date d'embauche

Exercice 7

Coder la classe correspondante pour créer ce nouveau type de donnée.
Comme vous l'avez fait pour les dates :

1. créez une procédure permettant d'afficher en console les informations sur une Personne reçue en paramètre.
2. créez une fonction permettant de créer une personne par saisie des informations sur son nom, prénom, date naissance et date d'embauche.

```
class Personne:

    def __init__(self, nom, prenom, annee_naissance, date_embauche):
        self.nom = nom
        self.prenom = prenom
        self.annee_naissance = annee_naissance
        self.date_embauche = date_embauche

    def affichePersonne(self):
        print("Personne : " + str(self.nom) + " " + str(self.prenom))
        print("Date de naissance : ")
        self.annee_naissance.afficheDate()
        print("Date d'embauche : ")
        self.date_embauche.afficheDate()
```

Exercice 8

Créer une fonction permettant de comparer 2 dates :
estPlusRecenteQue(date1, date2)
Cette fonction reçoit 2 dates et retourne true si la date1 est plus recente que la date2, sinon si la date2 est plus recente ou égale à la date1 alors elle retourne false.

```
def estPlusRecentQue(date1, date2):
    if (date1.annee > date2.annee):
        return True
    elif (date1.annee == date2.annee):
        if(date1.mois > date2.mois):
            return True
        elif (date1.mois == date2.mois):
            if(date1.jour > date2.jour):
                return True
    return False
```

Exercice 9

Créer une procédure permettant de comparer l'ancienneté de deux employés en se basant sur leur date d'embauche. Cette procédure affiche une synthèse de la comparaison en console. Exemple :

Personne : Karim Benzema
Date de naissance :
19/12/1987

```
Date d'embauche :
18/11/2020

n'a pas plus d'expérience que :

Personne : Youcef Belaili
Date de naissance :
14/03/1992
Date d'embauche :
26/10/2017
```

```python
def compareAnciennete(emp1, emp2):
    emp1.affichePersonne()

    if(estPlusRecentQue(emp1.date_embauche, emp2.date_embauche)):
        print("\na plus d'expérience que :\n")
    else:
        print("\nn'a pas plus d'expérience que :\n")

    emp2.affichePersonne()
```