

Devoir n°1**Algorithmique et POO****1h15**

Pour travailler sur des dates, on a créé la classe Date dont le code est écrit ci-dessous :

```
1 class Date:
2     def __init__(self, jour, mois, annee):
3         self.jour = ...
4         self.mois = ...
5         self.annee = ...
6         self.nb_jours_par_mois = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
7
8
9
10    def get_jour(self):
11        return self.jour
12
13    def get_mois(self):
14        return self.mois
15
16    def get_annee(self):
17        return ...
18
19    def set_jour(self, jour):
20        self.jour = jour
21
22    def set_mois(self, mois):
23        self.mois = ...
24
25    def set_annee(self, annee):
26        self.annee = annee
27
28    def est_bissextile(self):
29        ...
```

Partie A : accès et modification des données

Le constructeur de la classe Date prend en paramètres trois entiers représentant le jour, le mois et l'année, puis les affecte respectivement aux attributs jour, mois et annee.

1. Recopier et compléter les lignes 3 à 5 du code précédent.
2. Indiquer à quelle date correspond l'instance de la classe Date suivante : `d = Date(1, 5, 2000)`
3. Écrire le code permettant de créer une instance `d` de la classe Date qui représente la date du 19 juin 2024.
4. La méthode `get_annee` renvoie la valeur de l'attribut `annee`. Recopier et compléter les lignes 16 et 17 du code précédent.
5. La méthode `set_mois` modifie l'attribut `mois` en lui affectant la valeur passée en argument. Recopier et compléter les lignes 22 et 23 du code précédent.

L'attribut `nb_jours_par_mois` contient une liste qui correspond au nombre de jours pour chaque mois. Le mois de février contient généralement 28 jours mais lors des années bissextiles il en contient 29.

6. La classe Date dispose d'une méthode `est_bissextile`, qui utilise uniquement l'attribut `annee`, et qui renvoie `True` si l'année de l'instance courante est bissextile et `False` sinon. On veut compléter la méthode `__init__` pour ajuster le nombre de jours par mois pour les années bissextiles. Recopier et compléter les lignes 7 et 8 suivantes :

```
6 self.nb_jours_par_mois = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
7 if ... :
8     self.nb_jours_par_mois[...] = 29
```

Partie B : sur l'année de l'instance courante

Pour déterminer le nombre de jours au cours d'une année, il faut savoir si elle est bissextile. On rappelle qu'une année est bissextile si elle est divisible par 4 mais pas par 100 ou si elle est divisible par 400.

7. Écrire le code de la méthode `est_bissextile`.

On rappelle qu'un entier a est divisible par l'entier n si $a \% n == 0$.

On dote la classe `Date` de la méthode `nb_jours_passes` qui renvoie le nombre de jours passés dans l'année de l'instance courante.

```
1 def nb_jours_passes(self):
2     nb_jours = self.jour
3     mois = self.mois - 2
4     while mois >= 0:
5         nb_jours = nb_jours + self.nb_jours_par_mois[mois]
6         mois = mois - 1
7     return nb_jours
```

8. Indiquer quel sera l'affichage en console après l'exécution des deux instructions suivantes :

```
>>> d1 = Date(20, 3, 2001)
>>> d1.nb_jours_passes()
```

On dote la classe `Date` de la méthode `nb_jours_restants` qui renvoie le nombre de jours restants dans l'année de l'instance courante, soit 366 ou 365 moins le nombre de jours déjà passés, selon que l'année est bissextile ou non.

9. Recopier et compléter le script de la méthode `nb_jours_restants` ci-après :

```
1 def nb_jours_restants(self):
2     j = 365
3     if ...:
4         j = 366
5     return j - ...
```

Partie C : entre deux dates

On dote la classe `Date` de la méthode `nb_jours_depuis` qui prend en paramètre une autre instance `other` de la classe `Date` et qui renvoie le nombre de jours écoulés entre la date de l'instance `other` et la date de l'instance courante.

```
1 def nb_jours_depuis(self, other):
2     if other.get_annee() > self.get_annee():
3         return -1
4     if other.get_annee() == self.get_annee():
5         if other.nb_jours_passes() > self.nb_jours_passes():
6             return -1
7         if other.nb_jours_passes() == self.nb_jours_passes():
8             return 0
9     nb_jours = self.nb_jours_passes() + other.nb_jours_restants()
10    for annee in range(other.get_annee()+1, self.get_annee()):
11        d_suisant = Date(1, 1, annee)
12        if d_suisant.est_bissextile():
13            nb_jours += 366
14        else:
15            nb_jours += 365
16    return nb_jours
```

On crée les instances de la classe `Date` suivantes :

```
>>> d1 = Date(15, 6, 2024)
>>> d2 = Date(15, 6, 2024)
>>> d3 = Date(15, 7, 2024)
>>> d4 = Date(15, 6, 2025)
>>> d5 = Date(15, 6, 2022)
```

10. Indiquer quels seront alors les affichages en console après l'exécution de chacune des instructions suivantes (on précise que l'année 2024 est bissextile).

```
>>> d1.nb_jours_depuis(d2)
>>> d1.nb_jours_depuis(d3)
>>> d1.nb_jours_depuis(d4)
>>> d1.nb_jours_depuis(d5)
```

Le *timestamp* est le nombre de secondes qui se sont écoulées depuis le 1^{er} janvier 1970 à 00h00. Il s'agit de la date de la mise en marche du système d'exploitation UNIX. Par exemple, le 01/01/2024 à 00:00:00 correspond au timestamp de 1704063600.

11. Recopier et compléter le code de la méthode `timestamp` qui renvoie le nombre de secondes qui se sont écoulées depuis le 1^{er} janvier 1970.

```
1 def timestamp(self):
2     d = ...
3     return self.nb_jours_depuis(d) * 24 * 3600
```

Fin du sujet

Correction

Partie A : accès et modification des données

1. Lignes 3 à 5. [1]

```
1 self.jour = jour
2 self.mois = mois
3 self.annee = annee
```

2. 1^{er} mai 2000 [0,5]

3. d = Date(19, 6, 2024) [0,5]

4. Lignes 16 et 17 [0,5]

```
1 def get_annee(self):
2     return self.annee
```

5. Lignes 22 et 23 [0,5]

```
22 def set_mois(self, mois):
23     self.mois = mois
```

6. Lignes 7 et 8 [1]

```
1 self.nb_jours_par_mois = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
2 if self.est_bissextile():
3     self.nb_jours_par_mois[1] = 29
```

B si manque « self »

Partie B : sur l'année de l'instance courante

7. Méthode est_bissextile [1,5]

```
def est_bissextile(self) :
    if self.annee % 400 == 0 : return True
    if self.annee % 100 == 0 : return False
    if self.annee % 4 == 0 : return True
    return False
```

D si ordre de conditions pas bon

8. 31 + 28 + 20 = 79. L'affichage indiquera 79. [0,5]

9. méthode nb_jours_restants [1]

```
1 def nb_jours_restants(self):
2     j = 365
3     if self.est_bissextile():
4         j = 366
5     return j - self.nb_jours_passees()
```

Partie C : entre deux dates

10. affichage console nb_jours_depuis [1,5]

```
>>> d1.nb_jours_depuis(d2) → 0
>>> d1.nb_jours_depuis(d3) → -1
>>> d1.nb_jours_depuis(d4) → -1
>>> d1.nb_jours_depuis(d5) → 731 (365 + 366)
```

11. méthode timestamp [0,5]

```
1 def timestamp(self):
2     d = Date(1,1,1970)
3     return self.nb_jours_depuis(d) * 24 * 3600
```