ICC, partie Programmation — Série d'exercices 6

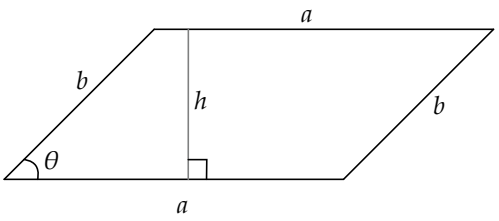
GC/MX, 25 octobre 2019

À faire individuellement ou par petits groupes de deux ou trois.

Tout d'abord, rendez le compilateur (en fait, le *linter*) moins strict à propos des annotations de types en Python. Pour ce faire, ouvrez votre workspace et allez éditer le fichier **`__workspace__.code-workspace`**. Réglez l'option **`python.linting.pylintEnabled`** à **`true`** et l'option **`python.linting.mypyEnabled`** à **`false`**. Ceci désactive la vérification plus «aggressive» de **`mypy`**, un *linter* assez strict, pour la remplacer par celle de **`pylint`**, plus «tolérant».

Exercice 1. Une classe simple

(a) Dans un nouveau fichier, déclarez une classe **`Parallelogram`** pour modéliser un parallélogramme. Celui-ci est défini par 3 **`float`**s: les longueurs a et b, et l'angle θ comme représenté ci-dessous.



Déclarez la méthode **`__init__`** de façon à demander ces 3 valeurs lors de la création d'un objet de type **`Parallelogram`** et les stocker dans des variables d'instance. Vous devriez pouvoir créer un parallélogramme ainsi (où l'angle est exprimé en degrés):

```
p = Parallelogram(a=20, b=10, theta=45)
```

(b) Ajoutez la méthode **`__repr__`** de façon à retourner sous forme de string la représentation du parallélogramme qui correspond au code nécessaire pour le créer. Par exemple, **`print(p)`** avec **`p`** comme défini ci-dessus doit vous retourner **`Parallelogram(a=20, b=10, theta=45)`**.

(c) Ajoutez les méthodes **`perimeter`**, **`height`** et **`area`** pour calculer, respectivement, le périmètre, la hauteur (h dans la figure ci-dessus) et l'aire du parallélogramme. Ce code devrait fonctionner:

```
p = Parallelogram(a=20, b=10, theta=45)
print(p)             # Parallelogram(a=20, b=10, theta=45)
print(p.perimeter()) # 60
print(p.height())    # about 7.07
print(p.area())      # about 141.4
```

Exercice 2. Compréhension de code et `___repr___`

Nous allons explorer en Python la liste des marées noires qui se sont produites depuis le début du XX^{e} siècle. Pour ce faire, créez un nouveau fichier **`oilspills.py`** et collez-y le code de départ de la page Moodle.

(a) Explorez le code de **`oilspills`** tel que vous l'avez importé:

- la classe **`OilSpillEvent`** modélise les données d'une marée noire. En regardant les champs de cette classe, vous pouvez facilement repérer par quels éléments un événement est défini et quels sont les types de ces champs.
- la variable **`all_events`** est une liste où chaque élément est un objet de type **`OilSpillEvents`**.

(b) Créez un nouveau fichier **`.py`** et ajoutez-y un import approprié de manière à pouvoir y faire référence à la liste **`all_events`** du fichier **`oilspills.py`**. Affichez le nombre d'événements que vous trouvez dans cette liste selon ce format:
`Il y a 120 événements.`

(c) Avec une boucle **`for-in`**, affichez tous les éléments de **`all_events`** un par un.

(d) Dans la classe **`OilSpillEvent`**, modifiez la méthode **`__repr__`** pour construire un string différent lorsque **`min_tonnage`** et **`max_tonnage`** coïncident, qui se terminerait donc par exemple par

`Spilled 3800 barrels.`

à la place de

`Spilled between 3800 and 3800 barrels.`

Exercice 3. Statistiques et filtres

(a) Dans votre fichier qui n'est pas **`oilspills.py`**, écrivez une fonction

```
def print_statistics(events:List[OilSpillEvent]) -> None
```

qui calcule la quantité totale déversée (minimum et maximum) par les événements fournis en paramètre et la période sur laquelle ça s'est passé (dates de début et de fin). À la fin, cette fonction doit afficher ces statistiques, selon les calculs faits. Par exemple, si on lui passe tous les événements connus, on devrait voir ceci:

`Entre le 14.12.1907 et le 01.07.2011, 120 événements ont causé le déversement d'entre 6460121 et 7690591 barils de brut.`

Votre code ne doit pas partir du principe que l'array **`events`** est trié chronologiquement.

Indice 1: pour comparer deux dates, vous pouvez directement utiliser les opérateurs **`<`** et **`>`**. *Indice 2:* pour formater une date selon le modèle «jour.mois.année», vous pouvez appeler la méthode **`strftime`** sur un objet de type **`datetime`** comme ceux stockés dans les variables d'instance des objets de type **`OilSpillEvent`**. Cette méthode demande un argument de type **`str`** qui dénote le format de date à utiliser pour la conversion. Ici, par exemple, on utilise **`"%d.%m.%Y"`**:

```
my_date = ... # de type datetime
my_date_as_string = my_date.strftime("%d.%m.%Y")
```

(b) Écrivez une méthode

```
def filter_by_country(country:str, events:List[OilSpillEvent]) -> List[OilSpillEvent]
```

qui retourne une nouvelle liste avec uniquement les événements de l'array **`events`** qui se sont produits dans le pays donné avec le paramètre **`country`**. Testez votre méthode avec les pays **`"United States"`** et **`"Australia"`**. Utilisez la valeur de retour de **`filter_by_country`** pour afficher des statistiques avec **`print_statistics`** sur les éléments filtrés.

Pour **`"United States"`**, vous devriez avoir:
`Entre le 14.03.1910 et le 01.07.2011, 51 événements ont causé le déversement d'entre 2137916 et 2391355 barils de brut.`

Pour **`"Australia"`**, vous devriez avoir:
`Entre le 21.07.1991 et le 03.04.2010, 4 événements ont causé le déversement d'entre 21513 et 47544 barils de brut.`

Le dernier exercice est sur la page suivante.

Exercice 4. Filtre par date

(a) Ajoutez ce code à la fin de votre fichier (récupérable depuis Moodle). À chaque itération, cette boucle calcule deux dates: le début d'une décennie (**`from_date`**) et la fin de la même décennie (**`to_date`**). Votre but est le calculer la valeur de la variable **`num_events`** de façon à ce que le code affiche le nombre d'événements qui se sont passés entre ces deux dates.

```
from datetime import datetime

year_increment = 10
for year in range(1900, 2010, year_increment):
    end_year = year + year_increment
    from_date = datetime(year, 1, 1)
    to_date = datetime(end_year, 1, 1)

    num_events = 0

    # à compléter

    print(f"Entre {year} et {end_year}, il y a eu {num_events} événement(s).")
```

(b) Faites en sorte que, sur la console, le texte **`événement(s)`** s'affiche directement soit au pluriel soit au singulier selon le nombre d'événements.