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

Exercice 1. MyLittleFacebook

Nous poursuivons ici l'exemple vu au cours:

```python
from typing import Dict, Set

# MyLittleFacebook
friendships: Dict[str, Set[str]] = {}

def add_friends(name1: str, name2: str) -> None:
    if name1 in friendships:
        friendships[name1].add(name2)
    else:
        friendships[name1] = {name2}
    if name2 in friendships:
        friendships[name2].add(name1)
    else:
        friendships[name2] = {name1}

add_friends("Alex", "Victor")
add_friends("Alex", "Emelyne")
add_friends("Alex", "Emelyne")
add_friends("Emelyne", "Rose")
```

(a) Ajoutez sous ce code une fonction **`known_people()`** qui nous donne, dans l'ordre alphabétique, tous les noms pour lesquels le dictionnaire contient des relations d'amitié.

Indices: Quel devra être son type de retour? Comment obtenir l'ensemble des clés connues par un dictionnaire? Vous pouvez utiliser la fonction **`sorted()`** de Python.

(b) Ajoutez une fonction **`friends_of()`**, qui prend un paramètre **`name`** de type **`str`** et qui renvoie l'ensemble des amis liés à ce nom. Prenez soin de retourner un set vide lorsque le dictionnaire n'a aucune info sur la personne demandée.

Vous pouvez tester avec ceci:

```python
print(friends_of("Emelyne")) # {'Rose', 'Alex'}
print(friends_of("Rachel"))  # set()
```

(c) Ajoutez une fonction **`are_friends()`**, qui indique si les deux noms passés comme paramètres sont déclarés comme amis. Quel est le type de retour?

Testez avec ceci:

```python
print(are_friends("Alex", "Victor")) # True
print(are_friends("Victor", "Alex")) # True
print(are_friends("Alex", "Rose"))   # False
print(are_friends("Alex", "Rachel")) # False
print(are_friends("Rachel", "Alex")) # False
```

(d) Si l'on passe par la méthode **`add_friends()`**, la modification du dictionnaire est toujours symétrique par rapport aux deux noms passés en paramètre. Mais on pourrait faire exprès de créer une incohérence en allant modifier le dictionnaire de manière asymétrique:

```python
friendships["Alex"].add("Rachel") # Modification asymétrique
```

Pour détecter cette situation problématique, ajoutez une fonction **`is_data_consistent()`**, qui dira si oui ou non l'ensemble du contenu du dictionnaire est cohérent, c'est-à-dire qui renverra **`True`** lorsque A fait partie des amis de B si et seulement si B fait partie des amis de A, et **`False`** sinon.

Indice: vous pouvez itérer sur chaque clé **`k`** liée à sa valeur **`v`** dans un dictionnaire **`d`** comme ceci:

```
for k, v in d.items():
    ...
```

Testez avec ceci:

```
print(f"Consistent before modification? {is_data_consistent()}")  # True
friendships["Alex"].add("Rachel") # Modification asymétrique
print(f"Consistent after modification? {is_data_consistent()}")   # False
```