Définition

Le set est un type de Collection dans Python. Il n'est **pas ordonné**, c'est à dire que l'ordre des éléments n'est pas préservé, et **non modifiable**. On ne peut ni ajouter, ni enlever des éléments après son initialisation. Il s'initialise avec des accolades. Il peut contenir tout type de données primitives (les collections sont exclues).

```
mon_set = {"mon", "super", "set"}
```

Valeurs identique interdites

La grand force du set c'est que les valeurs identiques y sont interdites, on peut ainsi s'assurer qu'une collection ne contient aucun doublon en l'initialisant ou en la convertissant en set.

```
duplicats_interdit = {1, 2, 3, 1}
print(duplicats_interdit) # {1, 2, 3}
```

```
my_list = [1, 2, 3, 1]
print(set(my_list)) # {1, 2, 3}
```

Méthodes de sets entre eux

---

set.difference

Retourne la différence du `set` devant la fonction avec un '.' par rapport les `set` dans les parenthèses

```
set_1 = {'a', 'b', 'c', 'd', 'e', 'f'}
set_2 = {'d', 'e'}
set_3 = {'c', 'd'}

diff = set_1.difference(set_2, set_3)
print(diff) # {'a', 'b', 'f'}
```

set.intersection

Retourne les éléments en communs entre le `set` devant la fonction avec un '.' avec plusieurs `set` dans les parenthèses.

```
set_1 = {5, 10, 15, 20}
set_2 = {5, 35, 40}
set_3 = {5, 50, 15}

intersect = set_1.intersection(set_2, set_3)
print(intersect) # {5}
```

set_1.union(set_2)

Retourne la réunion des deux sets sans doublons.

```
set_1 = {"banane", "kaki", "pomme", "poire"}
set_2 = {"pomme", "orange", "rhubarbe"}

union = set_1.union(set_2)
print(union) # {'rhubarbe', 'kaki', 'orange', 'banane', 'poire', 'pomme'}
```

📌 Liste exhaustives des méthodes de set.

Exercice: "18. Les sets"