Day-7 sets

Sets

Set est une collection d'articles. Permettez-moi de vous ramener à votre leçon de mathématiqu es élémentaire ou secondaire. La définition des mathématiques d'un ensemble peut également être appliquée dans Python. Set est une collection d'éléments distincts non ordonnés et non in dexés. Dans Python Set est utilisé pour stocker des éléments uniques, et il est possible de trou ver le *union*, *intersection*, *difference*, *symmetric difference*, *subset*, *super set* et *disjoint set* entre les ensembles.

Création d'un ensemble

Nous utilisons la fonction intégrée *set()*.

- Création d'un ensemble vide

```
# syntax
st = set()
```

- Création d'un ensemble avec des éléments initiaux

```
# syntaxe
st = {'item1', 'item2', 'item3', 'item4'}
```

Exemple:

```
# Fruits de syntaxe = {'banana', 'orange', 'mango', 'citron'}
```

Obtenir la longueur du réglage

Nous utilisons la méthode Len () pour trouver la longueur d'un ensemble.

```
# syntaxe st = {'item1', 'item2', 'item3', 'item4'}

Len (ST)
```

Exemple:

```
fruits = {'banana', 'orange', 'mango', 'citron'}
Len (fruits)
```

Accéder aux articles dans un ensemble

Nous utilisons des boucles pour accéder aux articles. Nous verrons cela dans la section Loop

Vérification d'un article

Pour vérifier si un élément existe dans une liste, nous utilisons l'opérateur d'adhésion *in*.

```
# syntaxe st = {'item1', 'item2', 'item3', 'item4'}
```

```
Print ("Set St Contient-il l'élément3?", 'item3' dans ST) # Let ST Contient-il l'élément3? Vrai
```

Exemple:

```
fruits = {'banana', 'orange', 'mango', 'citron'} print ('mango' in fruits) # tr
ue
```

Ajout d'éléments à un ensemble

Une fois un ensemble créé, nous ne pouvons modifier aucun élément et nous pouvons égale ment ajouter des éléments supplémentaires.

- Ajouter un élément à l'aide de *add()*

```
# syntaxe
st = {'item1', 'item2', 'item3', 'item4'}
St.Add ('item5')
```

Exemple:

```
fruits = {'banane', 'orange', 'mango', 'citron'}
fruits.add ('chaux')
```

- Ajouter plusieurs éléments à l'aide de *update()* Le *update()* permet d'ajouter plusieurs él éments à un ensemble. Le *update()* prend un argument de liste.

```
# syntaxe
st = {'item1', 'item2', 'item3', 'item4'}
St.Update (['item5', 'item6', 'item7'])
```

Exemple:

```
Fruits = {'banana', 'orange', 'mango', 'citron'} légumes = ('tomato', 'pommes de terre', 'Cabbage',
'oignon', 'carot') fruits.update (légumes)
```

Supprimer les articles d'un ensemble

Nous pouvons supprimer un élément d'un ensemble en utilisant la méthode *remove()*. Si l'élé ment n'est pas trouvé, la méthode *remove()* augmentera les erreurs, il est donc bon de vérifie r si l'élément existe dans l'ensemble donné. Cependant, la méthode *discard()* n'augmente auc une erreur.

```
# syntaxe st = {'item1', 'item2', 'item3', 'item4'}

St.Remove ('item2')
```

Les méthodes POP () suppriment un élément aléatoire d'une liste et il renvoie l'élément su pprimé.

Exemple:

```
fruits = {'banana', 'orange', 'mango', 'citron'} fruits.pop () # supprime un élém
ent aléatoire de l'ensemble
```

Si nous sommes intéressés par l'article supprimé.

```
fruits = {'banana', 'orange', 'mango', 'citron'} retient_item = fruits.pop ()
```

Effacer les articles dans un ensemble

Si nous voulons effacer ou vider l'ensemble, nous utilisons la méthode *clear*.

```
# syntaxe st = {'item1', 'item2', 'item3', 'item4'}
```

St.Clear ()

Exemple:

```
fruits = {'banana', 'orange', 'mango', 'citron'} fruits.clear () print (fruits) #
set ()
```

Suppression d'un ensemble

Si nous voulons supprimer l'ensemble lui-même, nous utilisons l'opérateur *del*.

```
# syntaxe st = {'item1', 'item2', 'item3', 'item4'}

Del St
```

Exemple:

```
fruits = {'banana', 'orange', 'mango', 'citron'} del fruits
```

Liste de conversion en réglage

Nous pouvons convertir la liste en set et défini sur la liste. La conversion de la conversion en set supprime les doublons et seuls les éléments uniques seront réservés.

```
# syntaxe
LST = ['item1', 'item2', 'item3', 'item4', 'item1'] st = set (lst) # {'item2', 'item4', 'item1', 'ite
m3'} - l'ordre est aléatoire, car les ensembles en général sont non ordonnés
```

Exemple:

```
fruits = ['banane', 'orange', 'mango', 'citron', 'orange', 'banana'] fruits = set (fruits) # {'mango', '
citron', 'banana', 'orange'}
```

Ensembles d'adhésion

Nous pouvons rejoindre deux ensembles à l'aide de la méthode *union()* ou *update()*.

- Union Cette méthode renvoie un nouvel ensemble

```
# syntaxe
st1 = {'item1', 'item2', 'item3', 'item4'} st2 = {'item5', 'item6', 'ite
m7', 'item8'} st3 = st1.Union (st2)
```

Exemple:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}  vegetables = {'tomato', 'potato', 'cabbage','onion',
'carrot'}  print(fruits.union(vegetables)) # {'lemon', 'carrot',  'tomato', 'banana', 'mango', 'orange',
'Cabbage', 'Potato', 'Onion'}
```

- Mettre à jour cette méthode insère un ensemble dans un ensemble donné

```
# syntaxe
ST1 = {'item1', 'item2', 'item3', 'item4'} st2 = {'item5', 'item6', 'item7', 'it
em8'} st1.update (st2) # contenu st2 est ajouté à ST1
```

Exemple:

```
Fruits = {'banana', 'orange', 'mango', 'citron'} légumes = {'tomato', 'pommes de terre', 'Cabbage'
, 'oign 'Orange', 'Cabbage', 'Potato', 'Onion'}
```

Trouver des articles d'intersection

L'intersection renvoie un ensemble d'éléments qui sont dans les deux ensembles. Voir l'exemple

```
# syntaxe
ST1 = {'item1', 'item2', 'item3', 'item4'} st2 = {'item3', 'item2'} st
1.INTERSECTION (ST2) # {'item3', 'item2'}
```

Exemple:

```
Whole_Numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10} se même_numbers = {0, 2, 4, 6, 8, 10} Whol
e_Numbers.INTERSECTION (même_numbers) # {0, 2, 4, 6, 8, 10}.


Python = {'p', 'y', 't', 'h', 'o', 'n'} dragon = {'d', 'r', 'a', 'g', 'o', 'n'} pyth
on.INTERSECTION (Dragon) # {'o', 'n'}
```

Vérification du sous-ensemble et super ensemble

Un ensemble peut être un sous-ensemble ou un super ensemble d'autres ensembles:

- Sous-ensemble: *issubset()*
- Super Set: *issuperset*

```
# syntaxe
st1 = {'item1', 'item2', 'item3', 'item4'}
ST2 = {'item2', 'item3'} st2.issubset (ST
1) # true st1.issuperset (st2) # true
```

Exemple:

```
Whole_Numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10} se même_numbers = {0, 2, 4, 6, 8, 10} who
le_numbers.issubset (même_numbers) # false, car c'est un super sage whole_numbers.issuper
set (même #_numbers) # true) # true) # true)



python = {'p', 'y', 't', 'h', 'o', 'n'} dragon = {'d', 'r', 'a', 'g', 'o', '
n'} python.issubset (dragon) # false
```

Vérification de la différence entre deux ensembles

Il renvoie la différence entre deux ensembles.

```
# syntaxe
ST1 = {'item1', 'item2', 'item3', 'item4'} st2 = {'item2', 'item3'} st2.différence (
st1) # set () st1.différence (st2) # {'item1', 'item4'} => st1 \ st2
```

Exemple:

```
Whole_Numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10} se même_numbers = {0, 2, 4, 6, 8,
10} Whole_numbers.Diférence (même_numbers) # {1, 3, 5, 7, 9}



python = {'p', 'y', 't', 'o', 'n'} dragon = {'d', 'r', 'a', 'g', 'o', 'n'} python.différence (dragon) # {'p', '
y', '} - - Le résultat est non recommandé (caractéristique de sets)) dragon.différence (python)
# {'d', 'r', 'a', 'g'}
```

Trouver une différence symétrique entre deux ensembles

Il renvoie la différence symétrique entre deux ensembles. Cela signifie qu'il renvoie un ens emble qui contient tous les éléments des deux ensembles, à l'exception des éléments présen ts dans les deux ensembles, mathématiquement: (a \ b) ∪ (b \ a)

```
# syntaxe
st1 = {'item1', 'item2', 'item3', 'item4'} st2 = {'item2', 'item3'} # cela signifie (
a \ b) ∪ ( b \ a ) st2.symmetric_diference (st1) # {'item1', 'item4'}
```

Exemple:

```
Whole_Numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10} certains_numbers = {1, 2, 3, 4, 5} Whole
_Numbers.Symmetric_dirence (Some_numbers) # {0, 6, 8, 9, 10}
```

```
python = {'p', 'y', 't', 'h', 'o', 'n'} dragon = {'d', 'r', 'a', 'g', 'o', 'n'} python.symmetric_diffefre '
D', 'H'}
```

Ensembles d'adhésion

Si deux ensembles n'ont pas d'élément ou d'éléments communs, nous les appelons des ense mbles disjoints. Nous pouvons vérifier si deux ensembles sont conjoints ou disjoints en utili sant la méthode *isdisjoint()*.

```
# syntaxe
ST1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'} st2.isdisjoint (st1)
# false
```

Exemple:

```
même_numbers = {0, 2, 4, 6, 8} odd_numbers = {1, 3, 5, 7, 9} se même
```

```
python = {'p', 'y', 't', 'h', 'o', 'n'} dragon = {'d', 'r', 'a', 'g', 'o', 'n'} python.isdisjoint (dragon)
# fail
```

Vous êtes une étoile montante. Vous venez de terminer les défis du jour 7 et vous avez 7 p as en avant pour votre chemin vers la grandeur. Faites maintenant quelques exercices pour vo tre cerveau et vos muscles.

Exercices: Jour 7

```
# sets

it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

A = {19, 22, 24, 20, 25, 26} b = {19, 22,
20, 25, 26, 24, 28, 27} Âge = [22, 19, 24,
25, 26, 24, 25, 24]
```

Exercices: niveau 1

1. Trouvez la longueur de l'ensemble IT_COMPANIES 2. Ajoutez «Tw itter» à IT_COMPANIES 3. Insérez plusieurs sociétés informatiques à l a fois à l'ensemble IT_COMPANIES 4. Supprimer l'une des sociétés de l'ensemble IT_COMPANIES 5. Quelle est la différence entre supprime r et rejeter

Exercices: niveau 2

1. Rejoignez A et B 2. Trouvez une intersection B 3. Est un sous-ensemble de B 4. Les ensembles disjoints A et B sont-i ls 5. rejoindre A avec B et B avec A 6. Quelle est la différen ce symétrique entre A et B 7. Supprimer les ensembles com plètement

Exercices: niveau 3

1. Convertissez les âges en un ensemble et comparez la longueur de la liste et de l'ensemble, qui L'un est plus grand? 2. Expliquez la différence entre les types de données suivants: ch aîne, liste, tuple et ensemble 3.
I am a teacher and I love to inspire and teach people. Combien de mots uniques ont été utilisés dans la phrase? Utilisez les méthodes Split et définissez pour obtenir les mots uniques.

Félicitations!