Frédéric
LURET

Python
Banque de questions

Version du 15 décembre 2024

Université de TOURS

0 Table des matières

1 Site 8 Comprehension Set

Set de comprehension

Question 1

DEPART DES Ensembles Générer un ensemble de carrés de nombres de 1 à 10
Exemple de sortie
{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}

Code python :

```python
squares = {x**2 for x in range(1, 11)}
print(squares)
```

q606.py

Crée un ensemble nommé carrés qui contient les carrés des nombres de 1 à 10. Voici comment fonctionne le code :
x**2 for x in range(1, 11) : Il s'agit d'une compréhension d'ensemble. Elle parcourt les nombres de 1 à 10 (inclus) en utilisant range(1, 11) et calcule le carré de chaque nombre x en utilisant x**2. La compréhension de l'ensemble rassemble ces valeurs au carré dans un ensemble. print(carrés) : Cette ligne de code affiche les carrés de l'ensemble sur la console.

Question 2

Créer un ensemble de nombres pairs de 1 à 20
Exemple de résultat
{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}

Code python :

```python
evens = {x for x in range(2, 21, 2)}
print(evens)
```

q607.py

Ce code Python génère un ensemble nommé evens contenant des nombres pairs compris entre 2 et 20 (inclus). Voici comment fonctionne le code :
evens = {x for x in range(2, 21, 2)} : Cette ligne utilise une compréhension d'ensemble pour créer l'ensemble des pairs. Elle parcourt les nombres compris entre 2 et 20 (inclus) avec un pas de 2. Cette plage comprend tous les nombres pairs de cette plage. La

compréhension de l'ensemble {...} rassemble ces nombres pairs et forme un ensemble avec des éléments distincts. Comme les ensembles n'autorisent pas les valeurs dupliquées, seuls les nombres pairs distincts sont inclus dans l'ensemble. print(evens) : Cette ligne affiche l'ensemble evens sur la console.

Question 3

Générer un ensemble de caractères à partir d'une chaîne de caractères

Exemple de sortie

Bonjour à tous !

{'e', 'r', 'o', 'H', 'w', 'l', 'd'}

Code python :

```python
string = "Hello, world!"
chars = {char for char in string if char.isalpha()}
print(string)
print(chars)
```

q608.py

Ce code Python traite une chaîne, string, et crée un ensemble, chars, contenant des caractères alphabétiques uniques (lettres) de la chaîne. Voici comment fonctionne ce code :

string = "Hello, world !" : Cette ligne initialise une variable nommée string et lui affecte la chaîne "Hello, world !" chars = {char for char in string if char.isalpha()} : Cette ligne initialise une variable nommée chars et lui affecte un ensemble créé à l'aide d'une compréhension d'ensemble. Elle itère sur chaque caractère char dans la chaîne et l'ajoute à l'ensemble s'il s'agit d'un caractère alphabétique (une lettre). ... : Cette notation est utilisée pour créer un ensemble. char pour char dans la chaîne : Cette partie de la compréhension itère sur chaque caractère de la chaîne. if char.isalpha() : Cette vérification conditionnelle permet de s'assurer que seuls les caractères alphabétiques (lettres) sont inclus dans l'ensemble. Elle utilise la méthode isalpha() pour déterminer si un caractère est une lettre. print(string) : Cette ligne de code imprime la chaîne de caractères originale sur la console. print(chars) : Cette ligne de code imprime le jeu de caractères sur la console.

Question 4

Créer un ensemble de longueurs de mots dans une phrase

Exemple de sortie

Voici un exemple de phrase.

{1, 2, 4, 6, 9}

Code python :

```python
sentence = "This is a sample sentence."
word_lengths = {len(word) for word in sentence.split()}
print(sentence)
print(word_lengths)
```

q609.py

Ce code Python traite une phrase, sentence, et crée un ensemble, word_lengths, contenant les longueurs des mots dans la phrase. Voici comment fonctionne le code : sentence = "Ceci est un exemple de phrase" : Cette ligne initialise une variable nommée sentence et lui affecte la chaîne de caractères "Ceci est un exemple de phrase". word_lengths = {len(word) for word in sentence.split()} : Cette ligne initialise une variable nommée word_lengths et lui affecte un ensemble créé à l'aide d'une compréhension d'ensemble. Elle divise la phrase en mots à l'aide de split() et itère sur chaque mot, en ajoutant sa longueur (nombre de caractères) à l'ensemble. {...} : Cette notation est utilisée pour créer un ensemble. len(word) for word in sentence.split() : Cette partie de la compréhension passe en revue chaque mot de la phrase après l'avoir divisée et calcule la longueur de chaque mot à l'aide de len(word). print(sentence) : Cette ligne de code imprime la phrase originale sur la console. print(longueur_des_mots) : Cette ligne de code affiche sur la console le jeu de mots_longueurs.

Question 5

Générer un ensemble de nombres premiers de 1 à 50
Exemple de résultat
{2, 3, 5, 37, 7, 41, 11, 43, 13, 47, 17, 19, 23, 29, 31}

Code python :

```python
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

prime_numbers = {x for x in range(1, 51) if is_prime(x)}
print(prime_numbers)
```

q610.py

Ce code Python définit une fonction is_prime pour vérifier si un nombre donné est premier ou non. Il utilise ensuite une compréhension d'ensemble pour créer un ensemble appelé prime_numbers contenant tous les nombres premiers de 1 à 50. Voici une explication pas à pas du code :
def is_prime(n) : Cette ligne définit une fonction nommée is_prime qui prend un entier n comme argument. Elle renvoie True si n est un nombre premier et False dans le cas contraire. if n <= 1 : : C'est la première condition. Si n est inférieur ou égal

à 1, il ne s'agit pas d'un nombre premier. Les nombres premiers sont supérieurs à 1, la fonction renvoie donc False dans ce cas. for i in range(2, int(n**0.5) + 1) : : Cette boucle itère de 2 à la racine carrée de n (arrondie à l'entier le plus proche) plus 1. Il s'agit d'une optimisation visant à réduire le nombre de diviseurs à vérifier. Les nombres premiers n'ont pas d'autres diviseurs que 1 et eux-mêmes, et nous n'avons besoin de vérifier que jusqu'à la racine carrée de n. if n % i == 0 : : Cette condition vérifie si n est divisible par i. Si c'est le cas, alors n n'est pas un nombre premier et la fonction renvoie False. return True : Si la fonction ne renvoie pas False dans les conditions précédentes, cela signifie que n n'est divisible par aucun nombre dans l'intervalle donné et qu'il s'agit d'un nombre premier. Dans ce cas, la fonction renvoie True. nombres_premiers = {x for x in range(1, 51) if is_prime(x)} : Cette ligne crée un ensemble appelé nombres_premiers à l'aide d'une compréhension d'ensemble. Elle parcourt les nombres de 1 à 50 et les inclut dans l'ensemble si la fonction is_prime renvoie True pour ce nombre. Cet ensemble de compréhension rassemble tous les nombres premiers de 1 à 50. print(nombres_premiers) : Cette ligne imprime l'ensemble prime_numbers sur la console, en affichant tous les nombres premiers compris entre 1 et 50.

Question 6

Créer un ensemble de lettres minuscules

Exemple de résultat

{'a', 'e', 'u', 'z', 'y', 'j', 'k', 't', 'x', 'd', 'r', 'v', 'o', 'h', 'f', 'i', 'c', 'g', 'l', 'p', 'b', 'n', 'm', 'q', 's', 'w'}.

Code python :

```python
lowercase_letters = {chr(x) for x in range(ord('a'), ord('z')+1)}
print(lowercase_letters)
```

q611.py

Crée un ensemble nommé lowercase_letters contenant toutes les lettres minuscules de l'alphabet anglais. Voici comment fonctionne le code :

{chr(x) for x in range(ord('a'), ord('z')+1)} : Il s'agit d'une compréhension d'ensemble qui itère sur une plage de points de code Unicode correspondant aux lettres minuscules anglaises. chr(x) : Cette fonction convertit un point de code Unicode x en un caractère correspondant. for x in range(ord('a'), ord('z')+1) : Elle parcourt une plage de points de code, en commençant par le point de code 'a' (ord('a')) jusqu'au point de code 'z' (ord('z')), inclus. lettres minuscules = {...} : Cette partie du code initialise une variable nommée lettres minuscules et lui assigne l'ensemble créé par la compréhension de l'ensemble. print(lettres_case_inférieures) : Cette ligne de code imprime l'ensemble lowercase_letters sur la console.

Question 7

Générer un ensemble de lettres majuscules

Exemple de résultat
{'U', 'M', 'F', 'D', 'A', 'I', 'R', 'Y', 'V', 'N', 'T', 'P', 'X', 'O', 'C', 'L', 'W', 'Q', 'K', 'J', 'H', 'Z', 'E', 'G', 'S', 'B' }.

Code python :

```python
uppercase_letters = {chr(x) for x in range(ord('A'), ord('Z')+1)}
print(uppercase_letters)
```

q612.py

Ce code Python crée un ensemble nommé uppercase_letters contenant toutes les lettres majuscules de l'alphabet anglais. Voici comment fonctionne ce code :
{chr(x) for x in range(ord('A'), ord('Z')+1)} : Il s'agit d'une compréhension d'ensemble qui itère sur une plage de points de code Unicode correspondant aux lettres majuscules de l'alphabet anglais. chr(x) : Cette fonction convertit un point de code Unicode x en un caractère correspondant. for x in range(ord('A'), ord('Z')+1) : Elle parcourt une plage de points de code, en commençant par le point de code 'A' (ord('A')) jusqu'au point de code 'Z' (ord('Z')), inclus. lettres_majuscules = {...} : Cette partie du code initialise une variable nommée uppercase_letters et lui assigne l'ensemble créé par la compréhension de l'ensemble. print(uppercase_letters) : Cette ligne de code imprime l'ensemble uppercase_letters sur la console.

Question 8

Créer un ensemble de nombres pairs au carré et de nombres impairs au cube de 1 à 10
Exemple de résultat
{64, 1, 4, 36, 100, 16, 343, 729, 27, 125}

Code python :

```python
result = {x**2 if x % 2 == 0 else x**3 for x in range(1, 11)}
print(result)
```

q613.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé result. L'ensemble contient les carrés des nombres pairs et les cubes des nombres impairs dans l'intervalle de 1 à 10. Voici comment fonctionne le code :
result = {x**2 if x % 2 == 0 else x**3 for x in range(1, 11)} : Cette ligne initialise un ensemble nommé result à l'aide d'une compréhension d'ensemble. for x in range(1, 11) : Cette partie du code met en place une boucle qui parcourt les nombres de 1 à 10 (inclus). {x**2 if x % 2 == 0 else x**3} : Pour chaque nombre x, cette partie calcule le carré (x**2) si x est pair (c'est-à-dire si x % 2 == 0 est vrai) et le cube (x**3) si x est impair. print(result) : Cette ligne de code affiche le résultat sur la console.

Question 9

Générer un ensemble de multiples communs de 3 et 5 jusqu'à 100
Exemple de sortie
{75, 45, 15, 90, 60, 30}

Code python :

```python
common_multiples = {x for x in range(1, 101) if x % 3 == 0 and x % 5 == 0}
print(common_multiples)
```

q614.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé common_multiples. L'ensemble contient les multiples communs de 3 et 5 dans l'intervalle de 1 à 100. Voici comment fonctionne le code :
common_multiples = {x for x in range(1, 101) if x % 3 == 0 and x % 5 == 0} : Cette ligne initialise l'ensemble common_multiples à l'aide d'une compréhension de l'ensemble. for x in range(1, 101) : Cette partie du code met en place une boucle qui parcourt les nombres de 1 à 100 (inclus). {x} : Pour chaque nombre x, cette partie l'inclut dans l'ensemble s'il est un multiple de 3 (x % 3 == 0) et également un multiple de 5 (x % 5 == 0). print(common_multiples) : Cette ligne de code affiche l'ensemble common_multiples sur la console.

Question 10

Créer un ensemble de chaînes inversées à partir d'un autre ensemble
Exemple de sortie
{'cerise', 'pomme', 'banane'}
{'ananab', 'elppa', 'yrrehc'}

Code python :

```python
words = {"apple", "banana", "cherry"}
reversed_words = {word[::-1] for word in words}
print(words)
print(reversed_words)
```

q615.py

Ce code Python utilise la compréhension d'un ensemble pour créer un ensemble nommé mots_inversés contenant les versions inversées des mots de l'ensemble original mots. Voici comment fonctionne ce code :
words = {"apple", "banana", "cherry"} : Cette ligne initialise un ensemble nommé words contenant trois mots : "pomme", "banane" et "cerise". mots_inversés = {mot[: :-1] pour mot dans mots} : Cette ligne initialise l'ensemble reversed_words à l'aide d'une compréhension de l'ensemble. pour mot dans mots : Cette partie du code met en place une boucle qui parcourt chaque mot de l'ensemble des mots. {word[: :-1]} : Pour chaque mot, cette partie inclut son inverse (chaîne inversée) dans l'ensemble mots_inversés. L'opération de découpage word[: :-1] est utilisée pour inverser le

mot. print(words) : Cette ligne de code affiche le jeu de mots original sur la console. print(mots_inversés) : Cette ligne de code affiche le jeu de mots inversés sur la console.

Question 11

Générer un ensemble de racines carrées positives à partir d'un ensemble de nombres positifs

Exemple de résultat

{16, 1, 4, 9, 25}

{1.0, 2.0, 3.0, 4.0, 5.0}

Code python :

```python
import math
positive_numbers = {1, 4, 9, 16, 25}
positive_sqrt = {math.sqrt(x) for x in positive_numbers}
print(positive_numbers)
print(positive_sqrt)
```

q616.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé positive_sqrt qui contient les racines carrées des nombres de l'ensemble positive_numbers. Voici comment fonctionne le code :

nombres_positifs = {1, 4, 9, 16, 25} : Cette ligne initialise un ensemble nommé nombres_positifs contenant cinq entiers positifs. positive_sqrt = {math.sqrt(x) for x in positive_numbers} : Cette ligne initialise l'ensemble positive_sqrt à l'aide d'une compréhension de l'ensemble. for x in positive_numbers : Cette partie du code met en place une boucle qui parcourt chaque nombre de l'ensemble positive_numbers. {math.sqrt(x)} : Pour chaque nombre, cette partie inclut sa racine carrée, calculée à l'aide de la fonction math.sqrt(), dans l'ensemble positive_sqrt. print(nombres_positifs) : Cette ligne de code affiche sur la console le jeu de nombres positifs d'origine. print(positive_sqrt) : Cette ligne de code affiche l'ensemble positive_sqrt sur la console.

Question 12

Créer un ensemble de mots en majuscules à partir d'une phrase

Exemple de sortie

Voici un exemple de phrase.

{'SAMPLE', 'THIS', 'SENTENCE.', 'A', 'IS'}

Code python :

```python
sentence = "This is a sample sentence."
uppercase_words = {word.upper() for word in sentence.split()}
print(sentence)
print(uppercase_words)
```

q617.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé mots_uppercase qui contient les versions en majuscules des mots de la phrase d'entrée. Voici comment fonctionne le code :

sentence = "Ceci est un exemple de phrase" : Cette ligne initialise une chaîne nommée sentence contenant un exemple de phrase. uppercase_words = {word.upper() for word in sentence.split()} : Cette ligne initialise l'ensemble uppercase_words à l'aide d'une compréhension de l'ensemble. for word in sentence.split() : Cette partie du code divise la phrase en mots à l'aide de la méthode split() et met en place une boucle pour parcourir les mots. {word.upper()} : Pour chaque mot, cette partie inclut sa version en majuscules (convertie à l'aide de la méthode upper()) dans l'ensemble uppercase_words. print(sentence) : Cette ligne de code imprime la phrase originale sur la console. print(mots_uppercase) : Cette ligne de code imprime le jeu de mots en majuscules sur la console.

Question 13

Générer un ensemble de caractères non-voyelles à partir d'une chaîne de caractères

Exemple de sortie

Bonjour à tous !

{' ', '!', 'w', ',', 'r', 'H', 'l', 'd' }

Code python :

```python
string = "Hello, world!"
non_vowels = {char for char in string if char.lower() not in 'aeiou'}
print(string)
print(non_vowels)
```

q618.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé non_voyelles qui contient les caractères non-voyelles de la chaîne d'entrée. Voici comment fonctionne ce code :

string = "Hello, world !" : Cette ligne initialise une chaîne nommée string contenant le texte "Hello, world !". non_voyelles = {char for char in string if char.lower() not in 'aeiou'} : Cette ligne initialise l'ensemble non_voyelles à l'aide d'une compréhension d'ensemble. for char in string : Cette partie du code met en place une boucle qui parcourt chaque caractère de la chaîne. {char} : Pour chaque caractère, cette partie l'inclut dans l'ensemble non_voyelles s'il ne s'agit pas d'une voyelle. La condition char.lower() not in 'aeiou' vérifie que la version minuscule du caractère ne se trouve pas dans la chaîne 'aeiou', filtrant ainsi les voyelles. print(string) : Cette ligne de code

imprime la chaîne originale sur la console. print(non_voyelles) : Cette ligne de code imprime le jeu de non_voyelles sur la console.

Question 14

Créer un ensemble de numéros uniques à partir d'une liste
Exemple de sortie
[1, 2, 3, 2, 4, 5, 1]
{1, 2, 3, 4, 5}

Code python :

```python
numbers = [1, 2, 3, 2, 4, 5, 1]
unique_numbers = {x for x in numbers}
print(numbers)
print(unique_numbers)
```

q619.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé nombres_uniques qui contient les éléments uniques de la liste nombres. Voici comment fonctionne ce code :
nombres = [1, 2, 3, 2, 4, 5, 1] : Cette ligne initialise une liste nommée numbers contenant plusieurs entiers, dont certaines valeurs répétées. nombres_uniques = {x for x in numbers} : Cette ligne initialise l'ensemble unique_numbers à l'aide d'une compréhension d'ensemble. pour x dans nombres : Cette partie du code met en place une boucle qui parcourt chaque élément de la liste des nombres. {x} : Pour chaque élément, cette partie l'inclut dans l'ensemble unique_numbers. Cependant, comme les ensembles ne permettent pas de dupliquer les éléments, seuls les éléments uniques sont inclus dans l'ensemble résultant. print(nombres) : Cette ligne de code imprime la liste originale des nombres sur la console. print(nombres_uniques) : Cette ligne de code affiche l'ensemble unique_numbers sur la console.

Question 15

nérer un ensemble de valeurs ASCII de caractères à partir d'une chaîne de caractères
Exemple de sortie
Bonjour à tous !
{32, 33, 100, 101, 72, 108, 44, 111, 114, 119}

Code python :

```python
string = "Hello, world!"
ascii_values = {ord(char) for char in string}
print(string)
print(ascii_values)
```

q620.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé ascii_values qui contient les valeurs ASCII des caractères de la chaîne d'entrée. Voici comment fonctionne le code :
string = "Hello, world!" : Cette ligne initialise une chaîne nommée string contenant le texte "Hello, world!". ascii_values = {ord(char) for char in string} : Cette ligne initialise l'ensemble ascii_values à l'aide d'une compréhension d'ensemble. for char in string : Cette partie du code met en place une boucle qui parcourt chaque caractère de la chaîne. {ord(char)} : Pour chaque caractère, cette partie inclut sa valeur ASCII, calculée à l'aide de la fonction ord(), dans l'ensemble ascii_values. print(string) : Cette ligne de code imprime la chaîne de caractères originale sur la console. print(ascii_values) : Cette ligne de code imprime le jeu de valeurs ascii sur la console.

Question 16

Générer un ensemble de tuples contenant des nombres et leurs carrés
Exemple de sortie
{(2, 4), (4, 16), (1, 1), (3, 9), (5, 25)}

Code python :

```python
num_squares = {(x, x**2) for x in range(1, 6)}
print(num_squares)
```

q621.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé num_squares qui contient des tuples avec des paires de nombres et leurs carrés. Voici comment fonctionne le code :
{(x, x**2) for x in range(1, 6)} : Cette compréhension d'ensemble crée un ensemble en itérant à travers les valeurs de x de 1 à 5 (inclus). Pour chaque valeur de x, elle génère un tuple (x, x**2) contenant le nombre original x et son carré x**2. Le résultat est un ensemble de ces tuples. print(num_squares) : Cette ligne de code affiche l'ensemble num_squares sur la console.

Question 17

Créer un ensemble de voyelles à partir d'une chaîne de caractères
Exemple de sortie
Bonjour à tous !
{'e', 'o'}

Code python :

```python
string = "Hello, world!"
vowels = {char.lower() for char in string if char.lower() in 'aeiou'}
print(string)
print(vowels)
```

q622.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé voyelles qui contient des voyelles minuscules à partir de la chaîne d'entrée. Voici comment fonctionne le code :

string = "Hello, world !" : Cette ligne initialise une chaîne nommée string contenant le texte "Hello, world !". voyelles = {char.lower() for char in string if char.lower() in 'aeiou'} : Cette ligne initialise l'ensemble des voyelles à l'aide d'une compréhension de l'ensemble. for char in string : Cette partie du code met en place une boucle qui parcourt chaque caractère de la chaîne. {char.lower()} : Pour chaque caractère, cette partie inclut sa version minuscule dans l'ensemble des voyelles, mais uniquement s'il s'agit d'une voyelle. La condition char.lower() dans 'aeiou' vérifie si la version minuscule du caractère est l'une des voyelles minuscules. print(string) : Cette ligne de code imprime la chaîne de caractères originale sur la console. print(voyelles) : Cette ligne de code imprime le jeu de voyelles sur la console.

Question 18

Générer un ensemble de nombres qui sont des carrés parfaits de 1 à 100

Exemple de sortie

{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}

Code python :

```python
perfect_squares = {x for x in range(1, 101) if int(x**0.5)**2 == x}
print(perfect_squares)
```

q623.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé perfect_squares qui contient des nombres carrés parfaits compris entre 1 et 100. Voici comment fonctionne le code :

{x for x in range(1, 101) if int(x**0.5)**2 == x} : Cette compréhension d'ensemble crée un ensemble en parcourant les valeurs de x de 1 à 100 (inclus). Pour chaque valeur de x, elle vérifie si la valeur entière de la racine carrée de x (int(x**0.5)) au carré (**2) est égale à x. Si cette condition est vraie, elle inclut x dans l'ensemble perfect_squares. print(perfect_squares) : Cette ligne de code affiche l'ensemble perfect_squares sur la console.

Question 19

Générer un ensemble de caractères qui sont des chiffres à partir d'une chaîne de caractères

Exemple de sortie
12345Bonjour67890
{'2', '8', '7', '3', '9', '0', '1', '5', '4', '6'}

Code python :

```python
string = "12345Hello67890"
digits = {char for char in string if char.isdigit()}
print(string)
print(digits)
```

q624.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé digits qui contient tous les caractères digitaux uniques de la chaîne d'entrée. Voici comment fonctionne le code :
string = "12345Hello67890" : Cette ligne initialise une chaîne nommée chaîne contenant un mélange de chiffres et de caractères non numériques. digits = {char for char in string if char.isdigit()} : Cette ligne initialise l'ensemble de chiffres à l'aide d'une compréhension de l'ensemble. for char in string : Cette partie du code met en place une boucle qui parcourt chaque caractère de la chaîne. {char.isdigit()} : Pour chaque caractère, cette partie inclut le caractère dans le jeu de chiffres s'il s'agit d'un chiffre. La condition char.isdigit() vérifie si le caractère est un chiffre ou non. print(string) : Cette ligne de code imprime la chaîne de caractères originale sur la console. print(digits) : Cette ligne de code imprime le jeu de chiffres sur la console.

Question 20

Créer un ensemble de nombres qui sont des puissances de 2 de 1 à 10
Exemple de résultat
{32, 64, 2, 128, 4, 256, 512, 1024, 8, 16}

Code python :

```python
powers_of_2 = {2**x for x in range(1, 11)}
print(powers_of_2)
```

q625.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé puissances_de_2 qui contient les puissances de 2 pour x allant de 1 à 10. Voici comment fonctionne le code :
{2**x for x in range(1, 11)} : Cette compréhension d'ensemble crée un ensemble en parcourant les valeurs de x de 1 à 10 (inclus). Pour chaque valeur de x, elle calcule la puissance de 2 correspondante, 2**x, et l'inclut dans l'ensemble powers_of_2. print(puissances_de_2) : Cette ligne de code affiche l'ensemble powers_of_2 sur la console.

Question 21

Générer un ensemble d'éléments communs à partir de deux listes
Exemple de sortie
[1, 2, 3, 4, 5]
[3, 4, 5, 6, 7]
{3, 4, 5}

Code python :

```python
list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]
common_elements = {x for x in list1 if x in list2}
print(list1)
print(list2)
print(common_elements)
```

q626.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé common_elements qui contient les éléments communs à deux listes, list1 et list2. Voici comment fonctionne ce code :
list1 = [1, 2, 3, 4, 5] et list2 = [3, 4, 5, 6, 7] : Ces lignes initialisent deux listes, list1 et list2, avec des valeurs entières. common_elements = {x for x in list1 if x in list2} : Cette ligne initialise l'ensemble common_elements à l'aide d'une compréhension d'ensemble. for x in list1 : Cette partie du code met en place une boucle qui parcourt chaque élément x de la liste1. {x for x in list1 if x in list2} : Pour chaque élément x dans list1, cette partie inclut x dans l'ensemble common_elements s'il se trouve également dans list2. print(list1) : Cette ligne de code affiche list1 sur la console. print(list2) : Cette ligne de code affiche la liste 2 sur la console. print(common_elements) : Cette ligne de code affiche le jeu d'éléments communs sur la console.

Question 22

Générer un ensemble de caractères non alphanumériques à partir d'une chaîne de caractères
Exemple de sortie
Bonjour à tous !
{'!', ' ', ','}

Code python :

```python
string = "Hello, world!"
non_alphanumeric = {char for char in string if not char.isalnum()}
print(string)
print(non_alphanumeric)
```

q627.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé non_alphanumeric qui contient tous les caractères non alphanumériques de la chaîne d'entrée. Voici comment fonctionne ce code :

string = "Hello, world!" : Cette ligne initialise une chaîne nommée string contenant des lettres, des espaces et des signes de ponctuation. non_alphanumeric = {char for char in string if not char.isalnum()} : Cette ligne initialise l'ensemble non_alphanumérique à l'aide d'une compréhension d'ensemble. for char in string : Cette partie du code met en place une boucle qui parcourt chaque caractère de la chaîne. {char.isalnum()} : Pour chaque caractère, cette partie inclut le caractère dans l'ensemble non_alphanumérique s'il n'est pas alphanumérique. La condition char.isalnum() vérifie si le caractère est alphanumérique ou non. print(string) : Cette ligne de code imprime la chaîne de caractères originale sur la console. print(non_alphanumerique) : Cette ligne de code imprime le jeu de caractères non alphanumériques sur la console.

Question 23

Créer un ensemble de caractères qui sont des consonnes à partir d'une chaîne de caractères

Exemple de sortie

Bonjour à tous !

{'H', 'r', 'w', 'd', 'l' }

Code python :

```python
string = "Hello, world!"
consonants = {char for char in string if char.isalpha() and
    ↪  char.lower() not in 'aeiou'}
print(string)
print(consonants)
```

q628.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé consonnes qui contient tous les caractères consonants de la chaîne d'entrée. Voici comment fonctionne ce code :

string = "Hello, world!" : Cette ligne initialise une chaîne nommée chaîne contenant des lettres, des espaces et des signes de ponctuation. consonnes = {char for char in string if char.isalpha() and char.lower() not in 'aeiou'} : Cette ligne initialise l'ensemble des consonnes à l'aide d'une compréhension de l'ensemble. for char in string : Cette partie du code met en place une boucle qui parcourt chaque caractère de la chaîne. {char.isalpha() et char.lower() not in 'aeiou'} : Pour chaque caractère, cette partie inclut le caractère dans le jeu de consonnes s'il s'agit d'un caractère alphabétique (c'est-à-dire char.isalpha()) et non d'une voyelle minuscule (c'est-à-dire char.lower() not in 'aeiou'). print(string) : Cette ligne de code affiche la chaîne de caractères originale sur la console. print(consonnes) : Cette ligne de code affiche le jeu de consonnes sur la console.

Question 24

Créer un ensemble de chaînes de caractères avec des caractères en majuscules
Exemple de sortie
{'banane', 'pomme', 'cerise'}
{'APPLE', 'BANANA', 'CHERRY'}

Code python :

```python
strings = {"apple", "banana", "cherry"}
uppercase_strings = {word.upper() for word in strings}
print(strings)
print(uppercase_strings)
```

q629.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé uppercase_strings qui contient les versions en majuscules des mots de l'ensemble d'entrée strings. Voici comment fonctionne ce code :
strings = {"apple", "banana", "cherry"} : Cette ligne initialise un ensemble nommé strings avec trois éléments de chaîne. uppercase_strings = word.upper() for word in strings : Cette ligne initialise l'ensemble uppercase_strings à l'aide d'une compréhension de l'ensemble. for word in strings : Cette partie du code met en place une boucle qui parcourt chaque mot (chaîne) de l'ensemble strings. {word.upper()} : Pour chaque mot, cette partie inclut la version en majuscules du mot dans l'ensemble uppercase_strings à l'aide de la méthode .upper(). print(strings) : Cette ligne de code imprime les chaînes du jeu original sur la console. print(uppercase_strings) : Cette ligne de code imprime le jeu de chaînes en majuscules sur la console.

Question 25

Créer une série de mots avec leurs caractères triés.
Exemple de sortie
{'banane', 'pomme', 'cerise'}
{'aaabnn', 'aelpp', 'cehrry'}

Code python :

```python
words = {"apple", "banana", "cherry"}
sorted_chars = {''.join(sorted(word)) for word in words}
print(words)
print(sorted_chars)
```

q630.py

Ce code Python utilise la compréhension d'un ensemble pour créer un nouvel ensemble appelé sorted_chars, qui contient les mots de l'ensemble d'entrée words, mais dont les caractères sont triés par ordre alphabétique. Voici comment fonctionne ce code :

words = {"apple", "banana", "cherry"} : Cette ligne initialise un ensemble nommé words avec trois éléments de type chaîne de caractères. sorted_chars = ''.join(sorted(word)) for word in words : Cette ligne initialise l'ensemble sorted_chars à l'aide d'une compréhension d'ensemble. for word in words : Cette partie du code met en place une boucle qui parcourt chaque mot (chaîne) de l'ensemble words. {''.join(sorted(word))} : Pour chaque mot, cette partie trie les caractères du mot par ordre alphabétique à l'aide de la fonction sorted(), puis ''.join() est utilisé pour concaténer les caractères triés en une seule chaîne. print(words) : Cette ligne de code imprime les mots du jeu original sur la console. print(sorted_chars) : Cette ligne de code affiche le jeu de caractères triés sur la console.

Question 26

Générer un ensemble de tuples contenant des nombres pairs et impairs de 1 à 10

Exemple de sortie

{(3, 4), (5, 4), (3, 10), (9, 2), (5, 10), (9, 8), (1, 6), (7, 4), (7, 10), (5, 6), (3, 6), (9, 4), (9, 10), (1, 2), (1, 8), (7, 6), (3, 2), (5, 2), (3, 8), (5, 8), (9, 6), (1, 4), (1, 10), (7, 2), (7, 8)}

Code python :

```python
even_odd_pairs = {(x, y) for x in range(1, 11, 2) for y in range(2, 12, 2)}
print(even_odd_pairs)
```

q631.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé even_odd_pairs. Il contient des paires de nombres dont le premier (x) est impair et compris entre 1 et 10, et le second (y) est pair et compris entre 2 et 12. Voici comment fonctionne le code :

{(x, y) for x in range(1, 11, 2) for y in range(2, 12, 2)} : Il s'agit d'une compréhension d'ensemble qui crée des paires (x, y) pour x dans l'intervalle des nombres impairs de 1 à 10 (1, 3, 5, 7, 9) et pour y dans l'intervalle des nombres pairs de 2 à 12 (2, 4, 6, 8, 10, 12). pour x dans range(1, 11, 2) : Cette partie du code met en place la boucle externe, qui parcourt les nombres impairs de 1 à 10 avec un pas de 2. for y in range(2, 12, 2) : Cette partie du code met en place la boucle interne, itérant à travers les nombres pairs de 2 à 12 avec un pas de 2. (x, y) : Pour chaque combinaison de x et y, il crée un tuple contenant les valeurs de x et y. print(even_odd_pairs) : Cette ligne de code affiche l'ensemble even_odd_pairs sur la console.

Question 27

Créer un ensemble de mots dont les voyelles sont remplacées par des traits de soulignement.

Exemple de résultat

{'banane', 'cerise', 'pomme'}

{'_ppl_', 'ch_rry', 'b_n_n_'}

Code python :

```
words = {"apple", "banana", "cherry"}
vowel_replaced = {''.join(['_' if char.lower() in 'aeiou' else char for
  ↪  char in word]) for word in words}
print(words)
print(vowel_replaced)
```

q632.py

Ce code Python utilise la compréhension d'un ensemble pour créer un nouvel ensemble nommé vowel_replaced, qui contient des mots de l'ensemble d'entrée words avec des voyelles remplacées par des traits de soulignement. Voici comment fonctionne ce code : words = {"apple", "banana", "cherry"} : Cette ligne initialise un ensemble nommé words avec trois éléments de type chaîne de caractères. vowel_replaced = ''.join(['_' if char.lower() in 'aeiou' else char for char in word]) for word in words : Cette ligne initialise l'ensemble vowel_replaced à l'aide d'une compréhension de l'ensemble. for word in words : Cette partie du code met en place une boucle qui parcourt chaque mot (chaîne) de l'ensemble de mots. ''.join(['_' if char.lower() in 'aeiou' else char for char in word]) : Pour chaque mot, cette partie crée une nouvelle chaîne où chaque caractère est remplacé par un trait de soulignement ('_') s'il s'agit d'une voyelle (en minuscules), ou laissé inchangé s'il ne s'agit pas d'une voyelle. print(words) : Cette ligne de code imprime les mots du jeu original sur la console. print(voyelle_remplacée) : Cette ligne de code affiche sur la console le jeu de voyelles_remplacées.

Question 28

Générer un ensemble de tuples contenant des nombres et leurs cubes
Exemple de sortie
{(3, 27), (4, 64), (1, 1), (5, 125), (2, 8)}

Code python :

```
num_cubes = {(x, x**3) for x in range(1, 6)}
print(num_cubes)
```

q633.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé num_cubes. Il contient des paires de nombres où le premier nombre (x) est compris entre 1 et 5, et le second nombre est le cube de x. Voici comment fonctionne le code :
{(x, x**3) for x in range(1, 6)} : Il s'agit d'une compréhension d'ensemble qui crée des paires (x, x**3) pour x dans la plage de nombres allant de 1 à 5. for x in range(1, 6) : Cette partie du code met en place la boucle, itérant sur les nombres de 1 à 5 (inclus). (x, x**3) : Pour chaque valeur de x, elle crée un tuple contenant la valeur de x et son cube, x**3. print(num_cubes) : Cette ligne de code affiche la valeur de num_cubes sur la console.

Question 29

Créer un ensemble de personnages uniques à partir d'une liste de mots
Exemple de sortie
['apple', 'banana', 'cherry']
{'y', 'a', 'e', 'c', 'b', 'p', 'l', 'n', 'h', 'r' }

Code python :

```python
words = ["apple", "banana", "cherry"]
unique_chars = {char for word in words for char in word}
print(words)
print(unique_chars)
```

q634.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé unique_chars, qui contient des caractères uniques extraits d'une liste de mots. Voici comment fonctionne ce code :
words = ["apple", "banana", "cherry"] : Cette ligne initialise une liste nommée words avec trois éléments de type chaîne de caractères. unique_chars = {char for word in words for char in word} : Cette ligne initialise l'ensemble unique_chars à l'aide d'une compréhension de l'ensemble. pour mot dans mots : Cette partie du code met en place une boucle imbriquée qui parcourt chaque mot de la liste des mots. for char in word : Pour chaque mot, cette partie parcourt chaque caractère du mot et ajoute chaque caractère unique à l'ensemble unique_chars. print(words) : Cette ligne de code imprime la liste originale de mots, words, sur la console. print(unique_chars) : Cette ligne de code affiche sur la console le jeu de caractères unique_chars, qui contient les caractères uniques des mots.

Question 30

Générer un ensemble de tuples contenant des nombres et leurs valeurs absolues
Exemple de sortie
[-2, 3, -5, 7, -11]
{(7, 7), (-11, 11), (3, 3), (-5, 5), (-2, 2)}

Code python :

```python
numbers = [-2, 3, -5, 7, -11]
absolute_values = {(x, abs(x)) for x in numbers}
print(numbers)
print(absolute_values)
```

q635.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé valeurs_absolues, qui contient des paires de nombres et leurs valeurs absolues à partir d'une liste de nombres. Voici comment fonctionne ce code :

nombres = [-2, 3, -5, 7, -11] : Cette ligne initialise une liste nommée numbers avec cinq éléments entiers. valeurs_absolues = {(x, abs(x)) for x in numbers} : Cette ligne initialise l'ensemble valeurs_absolues à l'aide d'une compréhension d'ensemble. for x in numbers : Cette partie du code met en place une boucle qui parcourt chaque nombre de la liste des nombres. (x, abs(x)) : Pour chaque nombre, cette partie crée un tuple contenant le nombre lui-même (x) et sa valeur absolue (abs(x)). print(nombres) : Cette ligne de code imprime la liste originale de nombres, numbers, sur la console. print(valeurs_absolues) : Cette ligne de code affiche sur la console le jeu de valeurs absolues, qui contient les paires de nombres et leurs valeurs absolues.

Question 31

Créer une série de mots dont les caractères sont répétés deux fois.

Exemple de sortie

['apple', 'banana', 'cherry']

{'aappppllee', 'bbaannaannaa', 'cchheerrrryy'}

Code python :

```python
words = ["apple", "banana", "cherry"]
duplicated_chars = {''.join([char*2 for char in word]) for word in
↪  words}
print(words)
print(duplicated_chars)
```

q636.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé duplicated_chars, qui contient des mots avec des caractères dupliqués à partir d'une liste de mots. Voici comment fonctionne ce code :

mots = ["pomme", "banane", "cerise"] : Cette ligne initialise une liste nommée words avec trois éléments de type chaîne de caractères. duplicated_chars = {''.join([char*2 for char in word]) for word in words} : Cette ligne initialise l'ensemble duplicated_chars à l'aide d'une compréhension d'ensemble. for word in words : Cette partie du code met en place une boucle qui parcourt chaque mot de la liste des mots. for char in word : Pour chaque mot, cette partie parcourt chaque caractère du mot et le double en utilisant char*2. ''.join(...) : Cette partie du code combine les caractères doublés en une seule chaîne pour chaque mot. print(words) : Cette ligne de code imprime la liste originale de mots, words, sur la console. print(duplicated_chars) : Cette ligne de code imprime sur la console l'ensemble duplicated_chars, qui contient les mots dont les caractères sont dupliqués.

Question 32

Générer un ensemble de tuples contenant des nombres et leurs carrés, mais seulement pour les nombres pairs

Exemple de sortie

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
{(2, 4), (4, 16), (8, 64), (10, 100), (6, 36)}

Code python :

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_num_squares = {(x, x**2) for x in numbers if x % 2 == 0}
print(numbers)
print(even_num_squares)
```

q637.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé even_num_squares, qui contient des tuples de nombres pairs et leurs carrés à partir d'une liste de nombres. Voici comment fonctionne ce code :
nombres = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] : Cette ligne initialise une liste nommée nombres avec dix éléments entiers. even_num_squares = {(x, x**2) for x in numbers if x % 2 == 0} : Cette ligne initialise l'ensemble even_num_squares à l'aide d'une compréhension d'ensemble. for x in numbers : Cette partie du code met en place une boucle qui parcourt chaque nombre de la liste des nombres. if x % 2 == 0 : Cette partie du code filtre les nombres pour n'inclure que les nombres pairs (divisibles par 2). (x, x**2) : Pour chaque nombre pair, cette partie crée un tuple contenant le nombre original et son carré. print(nombres) : Cette ligne de code imprime la liste originale des nombres, numbers, sur la console. print(even_num_squares) : Cette ligne de code affiche sur la console l'ensemble even_num_squares, qui contient des tuples de nombres pairs et leurs carrés.

Question 33

Créer un ensemble de nombres qui sont des cubes parfaits de 1 à 100
Exemple de résultat
{8, 1, 27}

Code python :

```
perfect_cubes = {x for x in range(1, 101) if int(x**(1/3))**3 == x}
print(perfect_cubes)
```

q638.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé perfect_cubes, qui contient des nombres cubiques parfaits compris entre 1 et 100. Voici comment fonctionne le code :
perfect_cubes = {x for x in range(1, 101) if int(x**(1/3))**3 == x} : Cette ligne initialise l'ensemble perfect_cubes à l'aide d'une compréhension de l'ensemble. for x in range(1, 101) : Cette partie du code met en place une boucle qui parcourt chaque nombre compris entre 1 et 100 (inclus). if int(x**(1/3))**3 == x : Cette partie du code filtre les nombres pour n'inclure que ceux qui sont des cubes parfaits. Elle vérifie si la racine cubique de x, convertie en entier et élevée à la puissance 3, est égale à x. Si

c'est le cas, cela signifie que x est un cube parfait. print(perfect_cubes) : Cette ligne de code affiche sur la console l'ensemble perfect_cubes, qui contient les nombres de cubes parfaits.

Question 34

Créer un ensemble de caractères sans espace à partir d'une chaîne de caractères
Exemple de sortie
Bonjour à tous !
{'e', 'r', 'H', 'd', 'w', ',', 'l', ' !', 'o'}

Code python :

```python
string = "Hello, world!"
non_whitespace_chars = {char for char in string if not char.isspace()}
print(string)
print(non_whitespace_chars)
```

q639.py

Ce code Python utilise une compréhension d'ensemble pour créer un nouvel ensemble nommé non_whitespace_chars, qui contient les caractères sans espace de la chaîne "Hello, world !". Voici comment fonctionne ce code :

non_whitespace_chars = {char for char in string if not char.isspace()} : Cette ligne initialise l'ensemble non_whitespace_chars à l'aide d'une compréhension de l'ensemble. for char in string : Cette partie du code parcourt chaque caractère de la chaîne "Hello, world !". if not char.isspace() : Cette partie du code filtre les caractères pour n'inclure que ceux qui ne sont pas des caractères d'espacement. Elle vérifie si le caractère char n'est pas un caractère d'espacement à l'aide de la méthode isspace(). print(string) : Cette ligne de code imprime la chaîne originale, "Hello, world !", sur la console. print(non_whitespace_chars) : Cette ligne de code affiche sur la console le jeu de caractères non_whitespace_chars, qui contient les caractères non blancs de la chaîne.

Question 35

Créer un ensemble de nombres pairs de 1 à 50 qui ne sont pas divisibles par 4
Exemple de résultat
{2, 34, 6, 38, 10, 42, 14, 46, 18, 50, 22, 26, 30}

Code python :

```python
#a corriger
```

pascorrige.py

Question 36

Générer un ensemble d'éléments communs à partir de plusieurs ensembles
Exemple de sortie
{1, 2, 3, 4, 5}
{3, 4, 5, 6, 7}
{5, 6, 7, 8, 9}
{5}

Code python :

```python
set1 = {1, 2, 3, 4, 5}
set2 = {3, 4, 5, 6, 7}
set3 = {5, 6, 7, 8, 9}
common_elements = {x for x in set1 if x in set2 and x in set3}
print(set1)
print(set2)
print(set3)
print(common_elements)
```

q640.py

Dans ce code Python, vous disposez de trois ensembles : set1, set2 et set3, et vous recherchez les éléments communs présents dans les trois ensembles. Voici comment fonctionne le code :
set1 = {1, 2, 3, 4, 5} : Cette ligne définit l'ensemble 1 comme un ensemble contenant les éléments 1, 2, 3, 4 et 5. set2 = {3, 4, 5, 6, 7} : Cette ligne définit l'ensemble 2 comme un ensemble contenant les éléments 3, 4, 5, 6 et 7. set3 = {5, 6, 7, 8, 9} : Cette ligne définit l'ensemble 3 comme un ensemble contenant les éléments 5, 6, 7, 8 et 9. common_elements = {x for x in set1 if x in set2 and x in set3} : Cette ligne crée un ensemble nommé common_elements à l'aide d'une compréhension d'ensemble. Elle parcourt les éléments de l'ensemble 1 et inclut un élément dans l'ensemble common_elements s'il est également présent dans les ensembles 2 et 3. En d'autres termes, il trouve les éléments qui sont communs aux trois ensembles. print(set1) : Cette ligne affiche l'ensemble set1 sur la console. print(set2) : Cette ligne affiche l'ensemble set2 sur la console. print(set3) : Cette ligne affiche l'ensemble set3 sur la console. print(common_elements) : Cette ligne affiche le jeu d'éléments communs sur la console.

Question 37

Créez un ensemble de chaînes de caractères dont les voyelles ont été supprimées.
Exemple de résultat
{'banane', 'cerise', 'pomme'}
{'chrry', 'ppl', 'bnn'}

Code python :

```python
strings = {"apple", "banana", "cherry"}
no_vowels = {''.join([char for char in word if char.lower() not in
    ↪  'aeiou']) for word in strings}
print(strings)
print(no_vowels)
```

q641.py

Vous voulez créer un nouvel ensemble appelé no_vowels qui contient les mêmes mots que strings, mais en supprimant les voyelles de chaque mot. Voici comment fonctionne le code :

strings = {"apple", "banana", "cherry"} : Cette ligne définit un ensemble nommé strings contenant trois mots : "pomme", "banane" et "cerise". no_vowels = {''.join([char for char in word if char.lower() not in 'aeiou']) for word in strings} : Cette ligne utilise une compréhension d'ensemble pour créer l'ensemble no_vowels. Elle parcourt chaque mot de l'ensemble strings et, pour chaque mot, traite les caractères qu'il contient. Il vérifie si chaque caractère (converti en minuscule) n'est pas une voyelle ('a', 'e', 'i', 'o', 'u'). Si le caractère n'est pas une voyelle, il l'inclut dans le mot traité. La partie ''.join(...) est utilisée pour concaténer les caractères en un seul mot. Ainsi, pour chaque mot de strings, ce code crée un nouveau mot sans voyelles et l'ajoute à l'ensemble no_vowels. print(strings) : Cette ligne affiche sur la console le jeu de chaînes original, qui contient les mots avec voyelles. print(no_vowels) : Cette ligne affiche sur la console le jeu no_vowels, qui contient les mots modifiés dont les voyelles ont été supprimées.

Question 38

Générer un ensemble de mots commençant par une voyelle à partir d'une phrase
Exemple de résultat
Voici un exemple de phrase contenant des mots commençant par des voyelles.
{'a', 'is'}

Code python :

```python
sentence = "This is a sample sentence with words starting with vowels."
vowel_start_words = {word for word in sentence.split() if
    ↪  word[0].lower() in 'aeiou'}
print(sentence)
print(vowel_start_words)
```

q642.py

Dans ce code Python, vous avez une phrase et vous voulez créer un ensemble appelé vowel_start_words qui contient les mots de la phrase qui commencent par une voyelle (en majuscules ou en minuscules). Voici comment fonctionne le code :

sentence = "Voici un exemple de phrase avec des mots commençant par des voyelles" : Cette ligne définit une chaîne appelée phrase contenant la phrase d'entrée. vowel_start_words = {word for word in sentence.split() if word[0].lower() in 'aeiou'} : Cette ligne utilise une compréhension d'ensemble pour créer un ensemble appelé mots_voyelles_début.

Voici comment cela fonctionne : sentence.split() divise la phrase en une liste de mots. for word in sentence.split() parcourt chaque mot de la liste. if word[0].lower() in 'aeiou' vérifie si la minuscule du premier caractère du mot est une voyelle. Si c'est le cas, le mot est inclus dans l'ensemble. print(phrase) : Cette ligne affiche la phrase originale sur la console. print(mots_voyelles_début) : Cette ligne affiche sur la console l'ensemble mots_début_voyelles, qui contient les mots de la phrase commençant par des voyelles.

Question 39

Créer un ensemble de voyelles uniques à partir d'une liste de mots

Exemple de résultat

['apple', 'banana', 'cherry']

{'a', 'e'}

Code python :

```python
words = ["apple", "banana", "cherry"]
unique_vowels = {char for word in words for char in word if
  ↪  char.lower() in 'aeiou'}
print(words)
print(unique_vowels)
```

q643.py

Ce code Python traite une liste de mots pour en extraire les voyelles uniques. Voici ce que fait chaque partie du code :

words = ["apple", "banana", "cherry"] : Cette ligne définit une liste appelée words, qui contient trois mots : "pomme", "banane" et "cerise". unique_vowels = {char for word in words for char in word if char.lower() in 'aeiou'} : Cette ligne utilise la compréhension d'un ensemble pour créer un ensemble appelé unique_voyelles. Voici comment cela fonctionne : for word in words parcourt chaque mot de la liste des mots. for char in word parcourt chaque caractère du mot. if char.lower() in 'aeiou' vérifie si la minuscule du caractère est une voyelle ('a', 'e', 'i', 'o', ou 'u'). Si c'est le cas, le caractère est inclus dans l'ensemble. La compréhension de l'ensemble permet de s'assurer que seuls les caractères voyelles uniques sont inclus. print(words) : Cette ligne imprime la liste originale des mots sur la console. print(unique_vowels) : Cette ligne affiche l'ensemble unique_voyelles, qui contient les voyelles uniques extraites des mots de la liste.

Question 40

Créer un ensemble de nombres palindromes de 1 à 100

Exemple de résultat

{1, 2, 3, 4, 5, 6, 7, 8, 9, 33, 11, 44, 66, 77, 99, 22, 55, 88}

Code python :

```python
palindromes = {x for x in range(1, 101) if str(x) == str(x)[::-1]}
print(palindromes)
```

q644.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé palindromes, qui contient des nombres palindromiques compris entre 1 et 100. Voici comment fonctionne le code :

palindromes = {x for x in range(1, 101) if str(x) == str(x)[: :-1]} : Cette ligne initialise l'ensemble palindromes à l'aide d'une compréhension de l'ensemble. for x in range(1, 101) : Cette partie du code parcourt les nombres de 1 à 100 (inclus). if str(x) == str(x)[: :-1] : Cette partie du code vérifie si un nombre est un palindrome. Pour ce faire, elle convertit le nombre x en chaîne de caractères à l'aide de str(x). Ensuite, elle vérifie si la représentation en chaîne du nombre est égale à son inverse, obtenu en découpant str(x)[: :-1]. print(palindromes) : Cette ligne de code imprime le jeu de palindromes sur la console, qui contient les nombres palindromiques compris dans l'intervalle spécifié.

Question 41

Générer un ensemble de mots qui sont des anagrammes à partir d'une liste de mots

Exemple de sortie

['apple', 'banana', 'elppa', 'race', 'care', 'cherry']

{'aaabnn', 'cehrry', 'aelpp', 'acer'}

Code python :

```python
words = ["apple", "banana", "elppa", "race", "care", "cherry"]
anagrams = {''.join(sorted(word)) for word in words}
print(words)
print(anagrams)
```

q645.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé anagrammes, qui contient les lettres triées de chaque mot de la liste words. Voici comment fonctionne le code :

anagrammes = {''.join(sorted(word)) for word in words} : Cette ligne initialise l'ensemble d'anagrammes en utilisant une compréhension de l'ensemble. for word in words : Cette partie du code parcourt chaque mot de la liste words. ''.join(sorted(word)) : Pour chaque mot, le code trie les lettres par ordre alphabétique à l'aide de la fonction sorted, puis les réunit en une seule chaîne. Cela permet de créer un anagramme du mot. print(words) : Cette ligne de code imprime la liste originale des mots, qui est ["apple", "banana", "elppa", "race", "care", "cherry"], sur la console. print(anagrammes) : Cette ligne de code imprime le jeu d'anagrammes, qui contient les lettres triées de chaque mot, sur la console.

Question 42

Créer un ensemble de mots avec leurs caractères triés par ordre décroissant
Exemple de résultat
['apple', 'banana', 'cherry']
{'pplea', 'yrrhec', 'nnbaaa'}

Code python :

```python
words = ["apple", "banana", "cherry"]
sorted_descending = {''.join(sorted(word, reverse=True)) for word in
↪  words}
print(words)
print(sorted_descending)
```

q646.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble appelé sorted_descending, qui contient les lettres de chaque mot de la liste words triées par ordre décroissant (dans l'ordre alphabétique inverse). Voici comment fonctionne ce code :
sorted_descending = {''.join(sorted(word, reverse=True)) for word in words} : Cette ligne initialise l'ensemble sorted_descending à l'aide d'une compréhension de l'ensemble. for word in words : Cette partie du code parcourt chaque mot de la liste words. ''.join(sorted(word, reverse=True)) : Pour chaque mot, le code trie les lettres dans l'ordre alphabétique inverse (ordre décroissant) en utilisant la fonction sorted avec l'argument reverse=True, puis les joint en une seule chaîne. print(words) : Cette ligne de code imprime la liste originale des mots, qui est ["apple", "banana", "cherry"], sur la console. print(sorted_descending) : Cette ligne de code imprime sur la console l'ensemble sorted_descending, qui contient les lettres de chaque mot trié par ordre décroissant.

Question 43

enerate un ensemble de caractères qui apparaissent exactement deux fois dans une chaîne de caractères
Exemple de sortie
Bonjour à tous !
{'o'}

Code python :

```python
string = "Hello, world!"
twice_chars = {char for char in string if string.count(char) == 2}
print(string)
print(twice_chars)
```

q647.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé twice_chars, qui contient les caractères qui apparaissent exactement deux fois dans la chaîne de caractères. Voici comment fonctionne ce code :

twice_chars = {char for char in string if string.count(char) == 2} : Cette ligne initialise l'ensemble twice_chars à l'aide d'une compréhension de l'ensemble. for char in string : Cette partie du code parcourt chaque caractère de la chaîne de caractères. if string.count(char) == 2 : cette ligne vérifie si le nombre de caractères dans la chaîne est égal à 2, ce qui signifie que le caractère apparaît exactement deux fois dans la chaîne. print(string) : Cette ligne de code imprime la chaîne originale, qui est "Hello, world !", sur la console. print(twice_chars) : Cette ligne de code imprime sur la console l'ensemble twice_chars, qui contient les caractères qui apparaissent exactement deux fois dans la chaîne.

Question 44

Créer un ensemble de chaînes de caractères avec tous les caractères en majuscules à partir d'une liste de chaînes de caractères.

Exemple de sortie

['apple', 'banana', 'cherry']

{'Banane', 'Pomme', 'Cerise' }

Code python :

```python
strings = ["apple", "banana", "cherry"]
capitalized_strings = {' '.join([word.capitalize() for word in
    ↪  string.split()]) for string in strings}
print(strings)
print(capitalized_strings)
```

q648.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé chaînes_capitalisées, qui contient des chaînes dont chaque mot est en majuscule. Voici comment fonctionne ce code :

chaînes_capitalisées = {' '.join([word.capitalize() for word in string.split()]) for string in strings} : Cette ligne initialise l'ensemble chaînes_capitalisées à l'aide d'une compréhension de l'ensemble. for string in strings : Cette partie du code parcourt chaque chaîne de la liste des chaînes. string.split() : Elle divise la chaîne actuelle en une liste de mots, en les séparant par des espaces. [word.capitalize() for word in string.split()] : Il parcourt les mots de la liste et met la première lettre de chaque mot en majuscule. ' '.join([word.capitalize() for word in string.split()]) : Il réunit les mots en majuscules en une seule chaîne de caractères, en les séparant par des espaces. print(strings) : Cette ligne de code imprime la liste originale des chaînes de caractères, qui est ["apple", "banana", "cherry"], sur la console. print(chaînes_capitalisées) : Cette ligne de code imprime sur la console le jeu de chaînes capitalisées, qui contient les chaînes dont les mots sont en majuscules.

Question 45

Générer un ensemble de mots comportant au moins une voyelle à partir d'une liste de mots

Exemple de résultat

['apple', 'banana', 'cherry']

{'banane', 'pomme', 'cerise' }

Code python :

```python
words = ["apple", "banana", "cherry", ]
vowel_words = {word for word in words if any(char.lower() in 'aeiou'
    ↪  for char in word)}
print(words)
print(vowel_words)
```

q649.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé mots_voyelles, qui contient des mots de la liste des mots contenant au moins une voyelle (a, e, i, o ou u). Voici comment fonctionne ce code :

mots_voyelles = {mot pour mot dans mots si any(char.lower() in 'aeiou' pour char dans mot)} : Cette ligne initialise l'ensemble mots_voyelles à l'aide d'une compréhension de l'ensemble. for word in words : Cette partie du code parcourt chaque mot de la liste des mots. if any(char.lower() in 'aeiou' for char in word) : Cette partie du code vérifie si au moins un caractère du mot actuel est une voyelle minuscule (a, e, i, o ou u). print(words) : Cette ligne de code imprime la liste originale des mots, qui est ["apple", "banana", "cherry"], sur la console. print(mots_voyelles) : Cette ligne de code imprime sur la console l'ensemble mots_voyelles, qui contient les mots comportant au moins une voyelle.

Question 46

Créer une série de mots dont les caractères sont répétés trois fois à partir d'une liste de mots.

Exemple de résultat

['apple', 'banana', 'cherry']

{'bbbaaannnaaannnaaa', 'aaapppppplllleee', 'ccchhheeerrrrrryyy'}

Code python :

```python
words = ["apple", "banana", "cherry"]
tripled_chars = {''.join([char*3 for char in word]) for word in words}
print(words)
print(tripled_chars)
```

q650.py

Ce code Python utilise la compréhension d'un ensemble pour créer un ensemble nommé tripled_chars, qui contient des mots de la liste des mots dont chaque caractère est répété trois fois. Voici comment fonctionne ce code :
tripled_chars = {''.join([char*3 for char in word]) for word in words} : Cette ligne initialise l'ensemble tripled_chars à l'aide d'une compréhension d'ensemble. for word in words : Cette partie du code parcourt chaque mot de la liste des mots. ''.join([char*3 for char in word]) : Cette partie du code répète trois fois chaque caractère du mot actuel et les joint à l'aide d'une chaîne vide. Cela crée un mot avec des caractères triplés. print(words) : Cette ligne de code imprime la liste originale de mots, qui est ["pomme", "banane", "cerise"], sur la console. print(tripled_chars) : Cette ligne de code imprime sur la console l'ensemble tripled_chars, qui contient des mots dont chaque caractère est répété trois fois.

Question 47

Créer un ensemble de mots dont les caractères sont triés et concaténés
Exemple de sortie
['apple', 'banana', 'cherry']
{'aelpp', 'cehrry', 'aaabnn'}

Code python :

```python
words = ["apple", "banana", "cherry"]
sorted_concatenated = {''.join(sorted(word)) for word in words}
print(words)
print(sorted_concatenated)
```

q651.py

Ce code Python utilise une compréhension d'ensemble pour créer un ensemble nommé sorted_concatenated, qui contient des mots de la liste de mots triés et concaténés. Voici comment fonctionne le code :
sorted_concatenated = {''.join(sorted(word)) for word in words} : Cette ligne initialise l'ensemble sorted_concatenated à l'aide d'une compréhension d'ensemble. for word in words : Cette partie du code parcourt chaque mot de la liste des mots. ''.join(sorted(word)) : Cette partie du code trie les caractères du mot actuel par ordre alphabétique, puis les joint en utilisant une chaîne vide. Cela crée un mot dans lequel les caractères sont triés. print(mots) : Cette ligne de code imprime la liste originale de mots, qui est ["apple", "banana", "cherry"], sur la console. print(sorted_concatenated) : Cette ligne de code imprime sur la console l'ensemble sorted_concatenated, qui contient des mots dont les caractères sont triés par ordre alphabétique.

Question 48

Créer une série de mots avec leurs caractères mélangés à partir d'une liste de mots.
Exemple de sortie

['apple', 'banana', 'cherry']
{'hreyrc', 'nbaana', 'lepap'}

Code python :

```python
import random

words = ["apple", "banana", "cherry"]
shuffled_chars = {''.join(random.sample(word, len(word))) for word in
↪  words}
print(words)
print(shuffled_chars)
```

q652.py

Ce code Python mélange les caractères de chaque mot de la liste des mots à l'aide d'un échantillonnage aléatoire et crée un ensemble nommé shuffled_chars pour stocker les versions mélangées des mots. Voici comment fonctionne le code :
import random : Cette ligne importe le module random, qui fournit des fonctions permettant de générer des nombres aléatoires et d'effectuer des opérations aléatoires. words = ["apple", "banana", "cherry"] : Cette ligne initialise une variable nommée words et lui attribue une liste de trois mots. shuffled_chars = {''.join(random.sample(word, len(word))) for word in words} : Cette ligne initialise un ensemble nommé shuffled_chars à l'aide d'une compréhension d'ensemble. for word in words : Cette partie du code parcourt chaque mot de la liste des mots. random.sample(word, len(word)) : Pour chaque mot, il utilise la fonction random.sample pour mélanger ses caractères. random.sample renvoie un échantillon aléatoire de la séquence d'entrée (dans ce cas, les caractères du mot) de la longueur spécifiée. ''.join(...) : Cette partie du code réunit les caractères mélangés en une seule chaîne. print(words) : Cette ligne de code imprime la liste originale des mots sur la console. print(shuffled_chars) : Cette ligne de code imprime le jeu de caractères mélangés (qui contient les versions mélangées des mots) sur la console.

Question 49

Filtrer les nombres négatifs d'un ensemble
Exemple de sortie
{1, 3, 5, -4, -2}
{1, 3, 5}

Code python :

```python
import random

numbers = {1, -2, 3, -4, 5}
positive_numbers = {x for x in numbers if x >= 0}
print(numbers)
print(positive_numbers)
```

q653.py

Ce code Python utilise une compréhension d'ensemble pour filtrer les nombres positifs d'un ensemble donné et crée un nouvel ensemble nommé nombres_positifs. Voici comment fonctionne le code :

import random : Cette ligne importe le module random, bien qu'il ne soit pas utilisé dans le code. nombres = {1, -2, 3, -4, 5} : Cette ligne initialise un ensemble nommé numbers avec une collection d'entiers, comprenant à la fois des nombres positifs et négatifs. nombres_positifs = {x for x in numbers if x >= 0} : Cette ligne crée un nouvel ensemble nommé nombres_positifs à l'aide d'une compréhension d'ensemble. pour x dans nombres : Cette partie du code parcourt chaque élément de l'ensemble de nombres. if x >= 0 : elle vérifie si chaque élément est supérieur ou égal à zéro, ce qui permet de filtrer les nombres positifs et zéro de l'ensemble de nombres. print(nombres) : Cette ligne affiche le jeu de nombres original sur la console. print(nombres_positifs) : Cette ligne affiche sur la console l'ensemble nombres_positifs (qui ne contient que les nombres positifs et zéro).

Question 50

Générer des paires d'éléments à partir de deux ensembles

Exemple de sortie

{1, 2, 3}

{'b', 'c', 'a'}

{(2, 'b'), (3, 'a'), (3, 'b'), (1, 'b'), (1, 'a'), (2, 'c'), (3, 'c'), (1, 'c'), (2, 'a')}

Code python :

```python
set1 = {1, 2, 3}
set2 = {"a", "b", "c"}
pairs = {(x, y) for x in set1 for y in set2}
print(set1)
print(set2)
print(pairs)
```

q654.py

Ce code Python crée des paires d'éléments à partir de deux ensembles, set1 et set2, et stocke ces paires dans un nouvel ensemble appelé paires. Voici comment fonctionne ce code :

set1 = {1, 2, 3} : Cette ligne initialise un ensemble nommé set1 contenant trois entiers. set2 = {"a", "b", "c"} : Cette ligne initialise un autre ensemble nommé set2

contenant trois chaînes de caractères. pairs = {(x, y) for x in set1 for y in set2} : Cette ligne utilise une compréhension de l'ensemble pour créer des paires d'éléments à partir de l'ensemble 1 et de l'ensemble 2. for x in set1 : Cette partie du code parcourt chaque élément de l'ensemble 1. for y in set2 : Pour chaque élément de l'ensemble 1, il parcourt chaque élément de l'ensemble 2, créant des paires (x, y) pour toutes les combinaisons. print(set1) : Cette ligne affiche l'ensemble 1 original sur la console. print(set2) : Cette ligne affiche l'ensemble 2 original sur la console. print(pairs) : Cette ligne affiche le jeu de paires (qui contient toutes les paires possibles d'éléments de set1 et 'set2) sur la console.

Question 51

Vérification des mots palindromes

Exemple de sortie

['radar', 'hello', 'level', 'world']

{'radar', 'level'}

Code python :

```python
words = ["radar", "hello", "level", "world"]
palindromes = {word for word in words if word == word[::-1]}
print(words)
print(palindromes)
```

q655.py

Ce code Python crée un ensemble nommé palindromes qui contient des mots palindromiques à partir d'une liste de mots donnée. Voici comment fonctionne le code : words = ["radar", "hello", "level", "world"] : Cette ligne initialise une liste nommée words avec quatre mots, dont certains sont palindromes et d'autres non. palindromes = {mot pour mot dans mots si mot == mot[: :-1]} : Cette ligne utilise une compréhension d'ensemble pour créer l'ensemble palindromes, qui ne contient que les mots de la liste words qui sont des palindromes. for word in words : Cette partie du code parcourt chaque mot de la liste des mots. si mot == mot[: :-1] : Elle vérifie si chaque mot est un palindrome. Pour ce faire, il vérifie si le mot est égal à son inverse (mot[: :-1]). print(words) : Cette ligne affiche la liste originale des mots sur la console. print(palindromes) : Cette ligne imprime le jeu de palindromes (qui contient les mots palindromiques de la liste des mots) sur la console.

Question 52

Conversion d'une liste de dictionnaires en un ensemble

Exemple de sortie

[{'a' : 1, 'b' : 2}, {'b' : 2, 'c' : 3}, {'c' : 3, 'd' : 4}]

{frozenset({('a', 1), ('b', 2)}), frozenset({('d', 4), ('c', 3)}), frozenset({('b', 2), ('c', 3)})}.

Code python :

```python
list_of_dicts = [{"a": 1, "b": 2}, {"b": 2, "c": 3}, {"c": 3, "d": 4}]
dict_set = {frozenset(d.items()) for d in list_of_dicts}
print(list_of_dicts)
print(dict_set)
```

q656.py

Ce code Python crée un ensemble nommé dict_set en convertissant une liste de dictionnaires, list_of_dicts, en ensembles de paires clé-valeur. Voici comment fonctionne ce code :
list_of_dicts = [{"a" : 1, "b" : 2}, {"b" : 2, "c" : 3}, {"c" : 3, "d" : 4}] : Cette ligne initialise une liste de dictionnaires nommée list_of_dicts. Chaque dictionnaire contient des paires clé-valeur. dict_set = {frozenset(d.items()) for d in list_of_dicts} : Cette ligne utilise une compréhension de l'ensemble pour créer l'ensemble dict_set. Pour chaque dictionnaire d dans list_of_dicts, elle convertit les paires clé-valeur en un frozenset en utilisant frozenset(d.items()). pour d dans list_of_dicts : Cette partie du code parcourt chaque dictionnaire de la liste_of_dicts. frozenset(d.items()) : Elle convertit les paires clé-valeur du dictionnaire en un frozenset. Un frozenset est utilisé parce qu'il s'agit d'un type immuable et qu'il peut être inclus dans un ensemble. print(list_of_dicts) : Cette ligne affiche la liste originale des dictionnaires sur la console. print(dict_set) : Cette ligne affiche sur la console l'ensemble dict_set (qui contient des frozensets de paires clé-valeur provenant des dictionnaires originaux).

Question 53

Suppression de la ponctuation d'une chaîne de caractères
Exemple de sortie
Bonjour à tous ! Comment ça va ?
{'e', ' ', 'g', 'o', 'h', 'd', 'l', 's', 'r', 'y', 'w', 't', 'H', 'n', 'v', 'i' }

Code python :

```python
import string

text = "Hello, world! How's everything?"
cleaned_text = {char for char in text if char not in
    ↪  string.punctuation}
print(text)
print(cleaned_text)
```

q657.py

Ce code Python supprime les caractères de ponctuation d'un texte donné en utilisant un ensemble de compréhension. Voici comment fonctionne le code :
import string : Cette ligne importe le module string, qui contient une constante string, string.punctuation, qui inclut tous les caractères de ponctuation. text = "Hello, world ! Comment ça va ?" : Cette ligne initialise une chaîne de caractères nommée

text avec un exemple de texte contenant de la ponctuation. cleaned_text = {char for char in text if char not in string.punctuation} : Cette ligne utilise une compréhension d'ensemble pour créer l'ensemble texte_nettoyé. Elle parcourt chaque caractère char du texte et l'inclut dans l'ensemble cleaned_text s'il ne s'agit pas d'un caractère de ponctuation. pour char dans texte : Cette partie du code parcourt chaque caractère de la chaîne de texte. if char not in string.punctuation : Cette partie du code vérifie si chaque caractère ne se trouve pas dans la chaîne string.punctuation, qui contient les caractères de ponctuation. print(text) : Cette ligne imprime le texte original sur la console. print(texte_nettoyé) : Cette ligne imprime le jeu de textes nettoyés, qui contient les caractères du texte original, à l'exception des caractères de ponctuation.

Question 54

Ensemble des mots ayant un préfixe donné
Exemple de sortie
['apple', 'banana', 'cherry', 'date']
ba
{'banane'}

Code python :

```python
words = ["apple", "banana", "cherry", "date"]
prefix = "ba"
words_with_prefix = {word for word in words if word.startswith(prefix)}
print(words)
print(prefix)
print(words_with_prefix)
```

q658.py

Ce code Python crée un ensemble nommé words_with_prefix qui contient des mots d'une liste donnée de mots commençant par un préfixe spécifié. Voici comment fonctionne ce code :
words = ["apple", "banana", "cherry", "date"] : Cette ligne initialise une liste nommée words avec quatre mots. prefix = "ba" : Cette ligne initialise une variable de type chaîne nommée préfixe avec le préfixe que vous souhaitez rechercher. words_with_prefix = {word for word in words if word.startswith(prefix)} : Cette ligne utilise une compréhension de l'ensemble pour créer l'ensemble words_with_prefix. Elle parcourt chaque mot de la liste des mots et l'inclut dans l'ensemble words_with_prefix s'il commence par le préfixe spécifié. pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. if word.startswith(prefix) : Cette partie du code vérifie si chaque mot commence par le préfixe spécifié. print(mots) : Cette ligne imprime la liste de mots originale sur la console. print(prefix) : Cette ligne affiche le préfixe spécifié sur la console. print(mots_avec_préfixe) : Cette ligne affiche l'ensemble words_with_prefix, qui contient les mots commençant par le préfixe spécifié.

Question 55

Fusionner les caractères d'une liste de chaînes
Exemple de sortie
['apple', 'banana', 'cherry']
{'r', 'b', 'p', 'n', 'a', 'e', 'c', 'h', 'y', 'l' }

Code python :

```python
strings = ["apple", "banana", "cherry"]
merged_chars = {char for string in strings for char in string}
print(strings)
print(merged_chars)
```

q659.py

Ce code Python crée un ensemble nommé merged_chars qui contient les caractères distincts d'une liste de chaînes de caractères. Voici comment fonctionne ce code : strings = ["apple", "banana", "cherry"] : Cette ligne initialise une liste nommée strings avec trois chaînes. merged_chars = {char for string in strings for char in string} : Cette ligne utilise une compréhension d'ensemble pour créer l'ensemble merged_chars. Elle parcourt chaque chaîne de la liste des chaînes et inclut chaque caractère dans l'ensemble merged_chars. for string in strings : Cette partie du code parcourt chaque chaîne de la liste des chaînes. for char in string : Elle parcourt chaque caractère de la chaîne en cours. print(strings) : Cette ligne imprime la liste originale des chaînes de caractères sur la console. print(merged_chars) : Cette ligne affiche le jeu de caractères fusionnés, qui contient tous les caractères distincts des chaînes originales.