Frédéric
LURET

Python
Banque de questions

Version du 15 décembre 2024

Université de TOURS

0 Table des matières

1 Site 8 Comprehension Dict

Dictionnary de comprehension

Question 1

DEPARTDICT Créer un dictionnaire de carrés de nombres de 1 à 10
Exemple de sortie
{1 : 1, 2 : 4, 3 : 9, 4 : 16, 5 : 25, 6 : 36, 7 : 49, 8 : 64, 9 : 81, 10 : 100}

Code python :

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

q660.py

Ce code Python crée un dictionnaire nommé carrés où les clés sont des nombres de 1 à 10, et les valeurs sont les carrés de ces nombres. Voici comment fonctionne le code : squares = {x : x ** 2 for x in range(1, 11)} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire squares. Elle parcourt chaque nombre x dans l'intervalle de 1 à 10 et affecte une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x) et la valeur est le carré de ce nombre (x ** 2). for x in range(1, 11) : Cette partie du code parcourt chaque nombre de 1 à 10. print(squares) : Cette ligne imprime le dictionnaire des carrés sur la console.

Question 2

Créer un dictionnaire dont les clés sont les nombres pairs et les valeurs leurs carrés.
Exemple de résultat
{0 : 0, 2 : 4, 4 : 16, 6 : 36, 8 : 64, 10 : 100, 12 : 144, 14 : 196, 16 : 256, 18 : 324, 20 : 400}

Code python :

```python
evens_squared = {x: x ** 2 for x in range(0, 21) if x % 2 == 0}
print(evens_squared)
```

q661.py

Ce code Python crée un dictionnaire nommé evens_squared dont les clés sont des nombres pairs compris entre 0 et 20, et dont les valeurs sont les carrés de ces nombres pairs. Voici comment fonctionne ce code :

evens_squared = {x : x ** 2 for x in range(0, 21) if x % 2 == 0} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire evens_squared. Elle parcourt chaque nombre pair x dans l'intervalle de 0 à 20 et affecte une paire clé-valeur au dictionnaire. La clé est le nombre pair lui-même (x) et la valeur est le carré de ce nombre pair (x ** 2). for x in range(0, 21) : Cette partie du code parcourt chaque nombre de 0 à 20. if x % 2 == 0 : elle vérifie si le nombre est pair en utilisant l'opérateur modulo (%) pour voir s'il n'y a pas de reste lorsque l'on divise par 2. print(evens_squared) : Cette ligne affiche le dictionnaire evens_squared sur la console.

Question 3

Créer un dictionnaire de mots et de leur longueur à partir d'une phrase

Exemple de sortie

Enter String : Python is awesome (Python est génial)

{'Python' : 6, 'is' : 2, 'awesome' : 7}

Code python :

```python
# sentence=input("Enter String : ")
sentence = "Python is awesome"
word_lengths = {word: len(word) for word in sentence.split()}
print(sentence)
print(word_lengths)
```

q662.py

Ce code Python crée un dictionnaire nommé word_lengths dont les clés sont les mots d'une phrase et les valeurs sont les longueurs de ces mots. Voici comment fonctionne ce code :

sentence = "Python is awesome" : Cette ligne initialise une variable de chaîne nommée phrase avec la phrase que vous souhaitez analyser. word_lengths = {word : len(word) for word in sentence.split()} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire word_lengths. Elle divise la phrase en mots à l'aide de .split() et parcourt chaque mot de la phrase. Pour chaque mot, elle assigne une paire clé-valeur au dictionnaire. La clé est le mot lui-même (word), et la valeur est la longueur de ce mot (len(word)). pour mot dans phrase.split() : Cette partie du code parcourt chaque mot de la phrase après l'avoir divisée en mots. print(sentence) : Cette ligne imprime la phrase originale sur la console. print(word_lengths) : Cette ligne affiche le dictionnaire word_lengths, qui contient les longueurs des mots de la phrase originale.

Question 4

Créer un dictionnaire de caractères minuscules à partir d'une chaîne de caractères

Exemple de sortie

Bonjour à tous

{'H' : 'h', 'e' : 'e', 'l' : 'l', 'o' : 'o', 'W' : 'w', 'r' : 'r', 'd' : 'd'}

Code python :

```python
text = "Hello World"
lowercase_chars = {char: char.lower() for char in text if
    ↪  char.isalpha()}
print(text)
print(lowercase_chars)
```

q663.py

Ce code Python crée un dictionnaire nommé lowercase_chars dont les clés sont les caractères alphabétiques d'un texte donné et les valeurs sont les versions minuscules correspondantes de ces caractères. Voici comment fonctionne ce code :
text = "Hello World" : Cette ligne initialise une variable de chaîne nommée texte avec le texte que vous souhaitez analyser. lowercase_chars = {char : char.lower() for char in text if char.isalpha()} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire lowercase_chars. Elle parcourt chaque caractère char du texte et vérifie s'il s'agit d'un caractère alphabétique (en utilisant char.isalpha()). Pour chaque caractère alphabétique, il attribue une paire clé-valeur au dictionnaire. La clé est le caractère lui-même (char) et la valeur est la version minuscule de ce caractère (char.lower()). pour char dans text : Cette partie du code parcourt chaque caractère du texte. if char.isalpha() : Cette partie du code vérifie si le caractère est alphabétique. print(text) : Cette ligne imprime le texte original sur la console. print(lowercase_chars) : Cette ligne affiche le dictionnaire lowercase_chars, qui contient les versions minuscules des caractères alphabétiques du texte original.

Question 5

Créer un dictionnaire des nombres et de leurs cubes
Exemple de résultat
[1, 2, 3, 4, 5]
{1 : 1, 2 : 8, 3 : 27, 4 : 64, 5 : 125}

Code python :

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

q664.py

Ce code Python crée un dictionnaire nommé cubes où les clés sont des nombres d'une liste, et les valeurs sont les cubes de ces nombres. Voici comment fonctionne le code :
nombres = [1, 2, 3, 4, 5] : Cette ligne initialise une liste nommée numbers avec une séquence de nombres que vous souhaitez analyser. cubes = {x : x ** 3 for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire cubes. Elle parcourt chaque nombre x dans la liste des nombres et affecte

une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x) et la valeur est le cube de ce nombre (x ** 3). pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. print(numbers) : Cette ligne imprime la liste originale des nombres sur la console. print(cubes) : Cette ligne imprime le dictionnaire cubes, qui contient les cubes des nombres de la liste originale.

Question 6

Créer un dictionnaire des nombres et de leurs carrés, à l'exclusion des nombres impairs
Exemple de résultat
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
{2 : 4, 4 : 16, 6 : 36, 8 : 64, 10 : 100}

Code python :

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

q665.py

Ce code Python crée un dictionnaire nommé even_squares dont les clés sont les nombres pairs d'une liste et les valeurs sont les carrés de ces nombres pairs. Voici comment fonctionne ce code :
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] : Cette ligne initialise une liste nommée numbers avec une séquence de nombres que vous souhaitez analyser. even_squares = {x : x ** 2 for x in numbers if x % 2 == 0} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire even_squares. Elle parcourt chaque nombre x dans la liste des nombres et vérifie si le nombre est pair (en utilisant x % 2 == 0). Pour chaque nombre pair, il attribue une paire clé-valeur au dictionnaire. La clé est le nombre pair lui-même (x) et la valeur est le carré de ce nombre (x ** 2). pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. if x % 2 == 0 : Cette partie du code vérifie si le nombre est pair (c'est-à-dire si son reste après division par 2 est égal à 0). print(even_squares) : Cette ligne affiche le dictionnaire even_squares, qui contient les carrés des nombres pairs de la liste originale.

Question 7

Créer un dictionnaire des nombres et de leur statut premier
Exemple de résultat
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
{1 : Faux, 2 : Vrai, 3 : Vrai, 4 : Faux, 5 : Vrai, 6 : Faux, 7 : Vrai, 8 : Faux, 9 : Faux, 10 : Faux}

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


numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
prime_status = {x: is_prime(x) for x in numbers}
print(numbers)
print(prime_status)
```

q666.py

Ce code Python définit une fonction is_prime(n) qui vérifie si un nombre donné n est premier. Il utilise ensuite une compréhension de dictionnaire pour créer un dictionnaire nommé prime_status où les clés sont des nombres d'une liste et les valeurs sont des valeurs booléennes indiquant si chaque nombre est premier ou non. Voici comment fonctionne le code :

def is_prime(n) : Ceci définit une fonction is_prime qui prend un seul argument n et vérifie s'il s'agit d'un nombre premier. La fonction renvoie True si n est premier et False dans le cas contraire. Elle utilise un algorithme standard de vérification de la primalité, testant les diviseurs jusqu'à la racine carrée de n. numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] : Cette ligne initialise une liste nommée numbers avec une séquence de nombres que vous souhaitez analyser. prime_status = {x : is_prime(x) for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire prime_status. Elle parcourt chaque nombre x dans la liste des nombres et affecte une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x) et la valeur est le résultat de l'appel à la fonction is_prime sur ce nombre, qui détermine s'il est premier ou non. pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. x : is_prime(x) for x in numbers : Elle utilise la compréhension d'un dictionnaire pour créer des paires clé-valeur où la clé est le nombre x et la valeur est le résultat de la fonction is_prime pour ce nombre. print(nombres) : Cette ligne imprime la liste originale des nombres sur la console. print(prime_status) : Cette ligne affiche le dictionnaire prime_status, qui contient le statut premier (Vrai ou Faux) de chaque nombre de la liste originale.

Question 8

Créer un dictionnaire de caractères et de leurs valeurs ASCII à partir d'une chaîne de caractères

Exemple de sortie

Tuteur Joes

{'T' : 84, 'u' : 117, 't' : 116, 'o' : 111, 'r' : 114, ' ' : 32, 'J' : 74, 'e' : 101, 's' : 115}

Code python :

```python
text = "Tutor Joes"
ascii_values = {char: ord(char) for char in text}
print(text)
print(ascii_values)
```

q667.py

Ce code Python prend une chaîne de texte et crée un dictionnaire ascii_values où les clés sont des caractères de la chaîne, et les valeurs sont les valeurs ASCII correspondantes. Voici comment fonctionne ce code :

text = "Tutor Joes" : Cette ligne initialise une variable texte avec la valeur "Tutor Joes". ascii_values = {char : ord(char) for char in text} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire ascii_values. Elle parcourt chaque caractère char de la chaîne de texte et affecte une paire clé-valeur au dictionnaire. La clé est le caractère lui-même (char) et la valeur est la valeur ASCII de ce caractère, obtenue à l'aide de la fonction ord(char). pour char dans text : Cette partie du code parcourt chaque caractère de la chaîne de texte. char : ord(char) for char in text : Elle utilise la compréhension d'un dictionnaire pour créer des paires clé-valeur où la clé est le caractère char et la valeur est sa valeur ASCII correspondante. print(text) : Cette ligne imprime la chaîne de texte originale sur la console. print(ascii_values) : Cette ligne affiche le dictionnaire ascii_values, qui contient les caractères de la chaîne originale comme clés et leurs valeurs ASCII comme valeurs.

Question 9

Créer un dictionnaire de mots et de leurs voyelles à partir d'une liste de chaînes de caractères.

Exemple de résultat

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

{'apple' : 2, 'banana' : 3, 'cherry' : 1}

Code python :

```python
words = ['apple', 'banana', 'cherry']
vowel_counts = {word: sum(1 for char in word if char.lower() in
    ↪  'aeiou') for word in words}
print(words)
print(vowel_counts)
```

q668.py

Ce code Python prend une liste de mots et crée un dictionnaire vowel_counts dont les clés sont les mots et les valeurs sont les nombres de voyelles (a, e, i, o, u) dans chaque mot. Voici comment fonctionne le code :

words = ['apple', 'banana', 'cherry'] : Cette ligne initialise une liste nommée words avec trois mots. vowel_counts = {word : sum(1 for char in word if char.lower() in 'aeiou') for word in words} : Cette ligne utilise une compréhension de dictionnaire

pour créer le dictionnaire vowel_counts. Elle parcourt chaque mot de la liste des mots et assigne une paire clé-valeur au dictionnaire. La clé est le mot lui-même (word), et la valeur est le résultat de l'expression suivante : sum(1 for char in word if char.lower() in 'aeiou') : Cette partie du code compte le nombre de voyelles dans chaque mot en parcourant chaque caractère char dans le mot. Si char est une voyelle (quelle que soit la casse), il ajoute 1 au compte. La fonction sum calcule le nombre total de voyelles dans le mot. pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. print(words) : Cette ligne imprime la liste de mots originale sur la console. print(vowel_counts) : Cette ligne imprime le dictionnaire vowel_counts, qui contient les mots comme clés et le nombre de voyelles dans chaque mot comme valeurs.

Question 10

Créer un dictionnaire de mots avec les lettres triées
Exemple de résultat
['python', 'programmation', 'langage']
{'python' : 'hnopty', 'programming' : 'aggimmnoprr', 'language' : 'aaegglnu'}

Code python :

```python
words = ['python', 'programming', 'language']
sorted_letters = {word: ''.join(sorted(word)) for word in words}
print(words)
print(sorted_letters)
```

q669.py

Ce code Python prend une liste de mots et crée un dictionnaire sorted_letters où les clés sont des mots, et les valeurs sont les mots avec leurs lettres triées dans l'ordre alphabétique. Voici comment fonctionne ce code :
mots = ['python', 'programmation', 'langage'] : Cette ligne initialise une liste nommée words avec trois mots. sorted_letters = {word : ''.join(sorted(word)) for word in words} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire sorted_letters. Elle parcourt chaque mot de la liste des mots et affecte une paire clé-valeur au dictionnaire. La clé est le mot lui-même (word), et la valeur est le résultat de l'expression suivante : ''.join(sorted(word)) : Cette partie du code trie les lettres de chaque mot par ordre alphabétique et les réunit en une chaîne. La fonction sorted est utilisée pour trier les caractères du mot, et join combine les caractères triés en une seule chaîne. pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. print(words) : Cette ligne imprime la liste de mots originale sur la console. print(lettres_triées) : Cette ligne affiche le dictionnaire sorted_letters, qui contient les mots comme clés et les mots avec leurs lettres triées par ordre alphabétique comme valeurs.

Question 11

Créer un dictionnaire des mots et de leur longueur, mais seulement pour les mots de

plus de 5 lettres.
Exemple de résultat
['apple', 'banana', 'cherry', 'date']
{'banane' : 6, 'cerise' : 6}

Code python :

```python
words = ['apple', 'banana', 'cherry', 'date']
long_word_lengths = {word: len(word) for word in words if len(word) > 5}
print(words)
print(long_word_lengths)
```

q670.py

Ce code Python prend une liste de mots et crée un dictionnaire long_word_lengths où les clés sont les mots dont la longueur est supérieure à 5 caractères, et les valeurs sont les longueurs de ces mots. Voici comment fonctionne ce code :
words = ['apple', 'banana', 'cherry', 'date'] : Cette ligne initialise une liste nommée words avec quatre mots. long_word_lengths = {word : len(word) for word in words if len(word) > 5} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire long_word_lengths. Elle parcourt chaque mot de la liste des mots et affecte une paire clé-valeur au dictionnaire uniquement si la longueur du mot est supérieure à 5 caractères. La clé est le mot lui-même (word) et la valeur est la longueur du mot (calculée à l'aide de len(word)). pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. if len(word) > 5 : Cette partie du code vérifie si la longueur du mot courant est supérieure à 5. print(words) : Cette ligne imprime la liste de mots originale sur la console. print(long_word_lengths) : Cette ligne affiche le dictionnaire long_word_lengths, qui contient les mots dont la longueur est supérieure à 5 comme clés et leurs longueurs respectives comme valeurs.

Question 12

Créer un dictionnaire des caractères et de leur fréquence dans une chaîne de caractères
Exemple de sortie
bonjour le monde
{'l' : 3, ' ' : 1, 'r' : 1, 'h' : 1, 'd' : 1, 'o' : 2, 'e' : 1, 'w' : 1}

Code python :

```python
text = "hello world"
char_frequency = {char: text.count(char) for char in set(text)}
print(text)
print(char_frequency)
```

q671.py

Ce code Python prend une chaîne de texte et crée un dictionnaire char_frequency où les clés sont des caractères uniques dans la chaîne, et les valeurs sont les fréquences de ces caractères dans la chaîne. Voici comment fonctionne le code :

text = "hello world" : Cette ligne initialise la chaîne de caractères text avec le texte "hello world". char_frequency = {char : text.count(char) for char in set(text)} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire char_frequency. Elle effectue les opérations suivantes : set(text) : Cette partie du code crée un ensemble de caractères uniques dans la chaîne de texte. L'utilisation d'un ensemble permet de s'assurer que chaque caractère n'est inclus qu'une seule fois, ce qui élimine les doublons. {char : text.count(char) for char in set(text)} : Cette partie du code parcourt chaque caractère unique char dans le jeu de caractères uniques et attribue une paire clé-valeur au dictionnaire. La clé est le caractère lui-même (char) et la valeur est le résultat de text.count(char), qui compte le nombre d'occurrences de ce caractère dans la chaîne de texte. print(text) : Cette ligne imprime la chaîne de texte originale sur la console. print(char_frequency) : Cette ligne affiche le dictionnaire char_frequency, qui contient des caractères uniques comme clés et leurs fréquences comme valeurs.

Question 13

Créer un dictionnaire de mots et de leurs formes inversées
Exemple de résultat
['apple', 'banana', 'cherry']
{'apple' : 'elppa', 'banana' : 'ananab', 'cherry' : 'yrrehc'}.

Code python :

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

q672.py

Ce code Python prend une liste de mots et crée un dictionnaire reversed_words dont les clés sont les mots originaux et les valeurs les versions inversées de ces mots. Voici comment fonctionne ce code :
mots = ['pomme', 'banane', 'cerise'] : Cette ligne initialise une liste nommée words avec trois mots. reversed_words = {word : word[: :-1] for word in words} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire mots_inversés. Elle parcourt chaque mot de la liste des mots et affecte une paire clé-valeur au dictionnaire. La clé est le mot original (word), et la valeur est le mot inversé en utilisant le découpage (word[: :-1]). pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. mot[: :-1] : Cette partie du code découpe le mot avec un pas de -1, inversant ainsi l'ordre des caractères dans le mot. print(mots) : Cette ligne imprime la liste de mots originale sur la console. print(mots_inversés) : Cette ligne affiche le dictionnaire mots_inversés, qui contient les mots originaux comme clés et leurs versions inversées comme valeurs.

Question 14

Créer un dictionnaire des nombres et de leurs carrés, mais uniquement pour les nombres pairs

Exemple de résultat

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

{2 : 4, 4 : 16, 6 : 36, 8 : 64, 10 : 100}

Code python :

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

q673.py

Ce code Python prend une liste de nombres et crée un dictionnaire even_squares dont les clés sont les nombres pairs de la liste originale, et les valeurs sont les carrés de ces nombres pairs. Voici comment fonctionne ce code :

nombres = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] : Cette ligne initialise une liste nommée numbers avec dix valeurs entières. even_squares = {x : x ** 2 for x in numbers if x % 2 == 0} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire even_squares. Elle parcourt chaque nombre x dans la liste des nombres et affecte une paire clé-valeur au dictionnaire si et seulement si le nombre est pair (c'est-à-dire si x % 2 == 0 est Vrai). La clé est le nombre pair lui-même (x) et la valeur est le carré de ce nombre pair (x ** 2). pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. if x % 2 == 0 : Cette partie du code vérifie si le nombre est pair en testant s'il est divisible par 2 sans reste. print(numbers) : Cette ligne imprime la liste originale des nombres sur la console. print(even_squares) : Cette ligne affiche le dictionnaire even_squares, qui contient les nombres pairs comme clés et leurs valeurs au carré comme valeurs.

Question 15

Créer un dictionnaire des nombres et de leurs facteurs

Exemple de résultat

[1, 2, 3, 4, 5]

{1 : [1], 2 : [1, 2], 3 : [1, 3], 4 : [1, 2, 4], 5 : [1, 5]}

Code python :

```python
numbers = [1, 2, 3, 4, 5]
factors = {x: [i for i in range(1, x + 1) if x % i == 0] for x in
    ↪  numbers}
print(numbers)
print(factors)
```

q674.py

Ce code Python prend une liste de nombres et crée un dictionnaire factors où les clés

sont les nombres de la liste originale, et les valeurs sont des listes de leurs facteurs. Voici comment fonctionne le code :

nombres = [1, 2, 3, 4, 5] : Cette ligne initialise une liste nommée numbers avec cinq valeurs entières. facteurs = {x : [i for i in range(1, x + 1) if x % i == 0] for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire des facteurs. Elle parcourt chaque nombre x dans la liste des nombres et affecte une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x) et la valeur est une liste de ses facteurs. pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. x : [i for i in range(1, x + 1) if x % i == 0] : Cette partie du code utilise une compréhension de liste pour générer une liste de facteurs pour chaque nombre. Elle parcourt les nombres de 1 à x (inclus) et n'inclut que les nombres pour lesquels x % i == 0 est Vrai. Cette condition vérifie si i est un facteur de x. print(nombres) : Cette ligne imprime la liste originale des nombres sur la console. print(facteurs) : Cette ligne affiche le dictionnaire des facteurs, qui contient des nombres comme clés et des listes de leurs facteurs comme valeurs.

Question 16

Créer un dictionnaire de mots et de leurs majuscules

Exemple de sortie

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

{'apple' : 'APPLE', 'banana' : 'BANANA', 'cherry' : 'CHERRY'}.

Code python :

```python
words = ['apple', 'banana', 'cherry']
uppercase_words = {word: word.upper() for word in words}
print(words)
print(uppercase_words)
```

q675.py

Ce code Python prend une liste de mots et crée un dictionnaire uppercase_words dont les clés sont les mots de la liste originale et les valeurs sont les versions en majuscules de ces mots. Voici comment fonctionne ce code :

mots = ['pomme', 'banane', 'cerise'] : Cette ligne initialise une liste nommée words avec trois valeurs de chaîne. uppercase_words = {word : word.upper() for word in words} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire uppercase_words. Elle parcourt chaque mot de la liste des mots et affecte une paire clé-valeur au dictionnaire. La clé est le mot original (word) et la valeur est la version en majuscules du mot (word.upper()). pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. word : word.upper() : Cette partie du code crée la paire clé-valeur. La clé est le mot original et la valeur est la version en majuscules du mot. print(words) : Cette ligne imprime la liste des mots originaux sur la console. print(mots_en_majuscules) : Cette ligne affiche le dictionnaire uppercase_words, qui contient des mots comme clés et leurs versions en majuscules comme valeurs.

Question 17

Créer un dictionnaire des nombres et de leur parité (paire ou impaire)
Exemple de résultat
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
{1 : 'odd', 2 : 'even', 3 : 'odd', 4 : 'even', 5 : 'odd', 6 : 'even', 7 : 'odd', 8 : 'even', 9 : 'odd', 10 : 'even'}

Code python :

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
parity = {x: 'even' if x % 2 == 0 else 'odd' for x in numbers}
print(numbers)
print(parity)
```

q676.py

Ce code Python prend une liste de nombres et crée un dictionnaire de parité où les clés sont les nombres de la liste originale, et les valeurs indiquent si chaque nombre est "pair" ou "impair". 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 valeurs entières. parité = {x : 'even' if x % 2 == 0 else 'odd' for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire de parité. Elle parcourt chaque nombre x dans la liste des nombres et affecte une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x) et la valeur est déterminée à l'aide d'une expression conditionnelle : Si x % 2 == 0, il est considéré comme "pair". Sinon, il est considéré comme "impair". pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. x : "pair" si x % 2 == 0 sinon "impair" : Cette partie du code crée la paire clé-valeur. La clé est le nombre x, et la valeur est soit "pair", soit "impair", en fonction du résultat de l'expression conditionnelle. print(nombres) : Cette ligne imprime la liste originale des nombres sur la console. print(parity) : Cette ligne affiche le dictionnaire de parité, qui contient des nombres comme clés et leur parité (paire ou impaire) comme valeurs.

Question 18

Créer un dictionnaire des nombres et de leurs représentations binaires
Exemple de sortie
[1, 2, 3, 4, 5]
{1 : '0b1', 2 : '0b10', 3 : '0b11', 4 : '0b100', 5 : '0b101'}

Code python :

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

q677.py

Ce code Python prend une liste de nombres et crée un dictionnaire binary_representations dont les clés sont les nombres de la liste originale et les valeurs sont leurs représentations binaires sous forme de chaînes de caractères. Voici comment fonctionne ce code :

nombres = [1, 2, 3, 4, 5] : Cette ligne initialise une liste nommée numbers avec cinq valeurs entières. binary_representations = {x : bin(x) for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire binary_representations. Elle parcourt chaque nombre x de la liste des nombres et affecte une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x) et la valeur est sa représentation binaire sous forme de chaîne, obtenue à l'aide de la fonction bin. pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. x : bin(x) : Cette partie du code crée la paire clé-valeur. La clé est le nombre x et la valeur est sa représentation binaire sous forme de chaîne de caractères. print(nombres) : Cette ligne imprime la liste originale des nombres sur la console. print(binary_representations) : Cette ligne affiche le dictionnaire binary_representations, qui contient des nombres comme clés et leurs représentations binaires comme valeurs.

Question 19

Créer un dictionnaire des nombres et de leurs factoriels
Exemple de résultat
[1, 2, 3, 4, 5]
{1 : 1, 2 : 2, 3 : 6, 4 : 24, 5 : 120}

Code python :

```python
import math

numbers = [1, 2, 3, 4, 5]
factorials = {x: math.factorial(x) for x in numbers}
print(numbers)
print(factorials)
```

q678.py

Ce code Python crée un dictionnaire factorials dont les clés sont des nombres issus d'une liste et les valeurs sont leurs valeurs factorielles calculées à l'aide de la fonction math.factorial. Voici comment fonctionne le code :

nombres = [1, 2, 3, 4, 5] : Cette ligne initialise une liste nommée numbers avec cinq valeurs entières. factorials = {x : math.factorial(x) for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire factorials. Elle

parcourt chaque nombre x dans la liste des nombres et affecte une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x) et la valeur est sa factorielle calculée à l'aide de la fonction math.factorial. pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. x : math.factorial(x) : Cette partie du code crée la paire clé-valeur. La clé est le nombre x et la valeur est sa valeur factorielle calculée à l'aide de math.factorial(x). print(nombres) : Cette ligne imprime la liste originale des nombres sur la console. print(factorials) : Cette ligne affiche le dictionnaire factorials, qui contient des nombres comme clés et leurs valeurs factorielles comme valeurs.

Question 20

Créer un dictionnaire des nombres et de leurs carrés, mais seulement pour les multiples de 3
Exemple de résultat
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
{3 : 9, 6 : 36, 9 : 81}

Code python :

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
multiples_of_3_squares = {x: x ** 2 for x in numbers if x % 3 == 0}
print(numbers)
print(multiples_of_3_squares)
```

q679.py

Ce code Python crée un dictionnaire multiples_of_3_squares où les clés sont des nombres d'une liste, mais seulement pour les nombres qui sont des multiples de 3, et les valeurs sont les carrés de ces nombres. Voici comment fonctionne le code :

nombres = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] : Cette ligne initialise une liste nommée nombres avec dix valeurs entières. multiples_of_3_squares = {x : x ** 2 for x in numbers if x % 3 == 0} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire multiples_of_3_squares. Elle parcourt chaque nombre x dans la liste des nombres et affecte une paire clé-valeur au dictionnaire uniquement si le nombre est un multiple de 3 (c'est-à-dire lorsque x % 3 == 0). La clé est le nombre lui-même (x) et la valeur est le carré de ce nombre (x ** 2). pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. if x % 3 == 0 : cette partie du code vérifie si le nombre x est un multiple de 3. x : x ** 2 : cette partie du code crée la paire clé-valeur. La clé est le nombre x et la valeur est son carré, calculé comme x ** 2. print(nombres) : Cette ligne imprime la liste originale des nombres sur la console. print(multiples_de_3_squares) : Cette ligne affiche le dictionnaire multiples_de_3_squares, qui contient des nombres (multiples de 3) comme clés et leurs valeurs au carré comme valeurs.

Question 21

Créer un dictionnaire des nombres et de leurs puissances de 2

Exemple de sortie
[1, 2, 3, 4, 5]
{1 : 2, 2 : 4, 3 : 8, 4 : 16, 5 : 32}

Code python :

```python
numbers = [1, 2, 3, 4, 5]
powers_of_2 = {x: 2 ** x for x in numbers}
print(numbers)
print(powers_of_2)
```

q680.py

Ce code Python crée un dictionnaire powers_of_2, où les clés sont des nombres d'une liste, et les valeurs sont 2 élevés à la puissance de chaque nombre. Voici comment fonctionne le code :
nombres = [1, 2, 3, 4, 5] : Cette ligne initialise une liste nommée numbers avec cinq valeurs entières. puissances_de_2 = {x : 2 ** x for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire powers_of_2. Elle parcourt chaque nombre x dans la liste des nombres et affecte une paire clé-valeur au dictionnaire. La clé est le nombre lui-même (x), et la valeur est calculée en élevant 2 à la puissance de ce nombre (2 ** x). pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. x : 2 ** x : Cette partie du code crée la paire clé-valeur. La clé est le nombre x, et la valeur est calculée comme 2 ** x, ce qui élève 2 à la puissance du nombre x. print(numbers) : Cette ligne imprime la liste originale des nombres sur la console. print(puissances_de_2) : Cette ligne affiche le dictionnaire powers_of_2, qui contient des nombres comme clés et leurs puissances de 2 correspondantes comme valeurs.

Question 22

Créer un dictionnaire de mots et de leurs voyelles, en excluant les mots sans voyelles.
Exemple de résultat
['apple', 'banana', 'cherry', 'dog', 'ct']
{'apple' : 2, 'banana' : 3, 'cherry' : 1, 'dog' : 1, 'cat' : 1}

Code python :

```python
words = ['apple', 'banana', 'cherry', 'dog', 'ct']
vowel_counts = {word: sum(1 for char in word if char.lower() in
    'aeiou') for word in words if
                    any(char.lower() in 'aeiou' for char in word)}
print(words)
print(vowel_counts)
```

q681.py

Ce code Python définit un dictionnaire vowel_counts, dont les clés sont des mots d'une liste contenant au moins une voyelle (a, e, i, o, u), et dont les valeurs sont les nombres de voyelles dans chaque mot. Voici comment fonctionne le code :

words = ['apple', 'banana', 'cherry', 'dog', 'ct'] : Cette ligne initialise une liste nommée words avec cinq valeurs de chaîne, dont certaines contiennent des voyelles. vowel_counts = {word : sum(1 for char in word if char.lower() in 'aeiou') for word in words if any(char.lower() in 'aeiou' for char in word)} : Cette ligne utilise une compréhension du dictionnaire pour créer le dictionnaire vowel_counts. Elle parcourt chaque mot de la liste des mots, et pour chaque mot, elle vérifie s'il y a au moins une voyelle (n'importe quel caractère voyelle) dans ce mot. pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. any(char.lower() in 'aeiou' for char in word) : Cette partie vérifie si l'un des caractères du mot est une voyelle minuscule ("a", "e", "i", "o", "u"). La partie char.lower() garantit que les voyelles minuscules et majuscules sont comptées. word : sum(1 for char in word if char.lower() in 'aeiou') : Si le mot contient au moins une voyelle, une paire clé-valeur est créée dans le dictionnaire. La clé est le mot lui-même (word), et la valeur est calculée en additionnant 1 pour chaque caractère du mot qui est une voyelle. La fonction char.lower() de la partie 'aeiou' vérifie si le caractère est une voyelle minuscule. print(words) : Cette ligne affiche la liste originale des mots sur la console. print(vowel_counts) : Cette ligne affiche le dictionnaire vowel_counts, qui contient les mots comportant au moins une voyelle et leur nombre de voyelles.

Question 23

Créer un dictionnaire de chaînes de caractères en remplaçant les voyelles par des traits de soulignement
Exemple de résultat
['apple', 'banana', 'cherry']
{'apple' : '_ppl_', 'banana' : 'b_n_n_', 'cherry' : 'ch_rry'}

Code python :

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

q682.py

Le code Python définit un dictionnaire underscored_words, dont les clés sont des mots d'une liste et les valeurs sont les mêmes mots dont les voyelles ont été remplacées par des traits de soulignement. Voici comment fonctionne le code :
words = ['apple', 'banana', 'cherry'] : Cette ligne initialise une liste nommée words avec trois valeurs de chaîne. underscored_words = {word : ''.join(['_' if char.lower() in 'aeiou' else char for char in word]) for word in words} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire underscored_words. Elle parcourt chaque mot de la liste des mots et crée une paire clé-valeur pour chaque mot. for word in words : Cette partie du code parcourt chaque mot de la liste des mots. ''.join(['_' if char.lower() in 'aeiou' else char for char in word]) : Pour chaque mot, il crée une version modifiée du mot dont les voyelles sont remplacées par des

traits de soulignement ("_"). Pour ce faire, il parcourt chaque caractère du mot (for char in word) et utilise une expression conditionnelle pour vérifier si le caractère est une voyelle minuscule ('a', 'e', 'i', 'o', 'u'). S'il s'agit d'une voyelle, il la remplace par un trait de soulignement ; sinon, il conserve le caractère tel quel. La méthode join est utilisée pour concaténer les caractères modifiés en une seule chaîne. print(words) : Cette ligne affiche la liste originale des mots sur la console. print(mots_soulignés) : Cette ligne affiche le dictionnaire underscored_words, qui contient les mots originaux comme clés et leurs versions remplacées par des voyelles comme valeurs.

Question 24

Créer un dictionnaire de mots et de leur longueur, mais uniquement pour les mots commençant par "a".
Exemple de résultat
['apple', 'banana', 'cherry', 'date']
{'apple' : 5}

Code python :

```python
words = ['apple', 'banana', 'cherry', 'date']
a_word_lengths = {word: len(word) for word in words if
    ↪  word.startswith('a')}
print(words)
print(a_word_lengths)
```

q683.py

Le code Python que vous avez fourni crée un dictionnaire a_word_lengths, dont les clés sont les mots d'une liste commençant par la lettre "a", et les valeurs sont les longueurs de ces mots. Voici comment fonctionne le code :
words = ['apple', 'banana', 'cherry', 'date'] : Cette ligne initialise une liste nommée words avec quatre valeurs de chaîne. a_word_lengths = {word : len(word) for word in words if word.startswith('a')} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire a_word_lengths. Elle parcourt chaque mot de la liste des mots et crée une paire clé-valeur pour chaque mot commençant par la lettre "a". pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. 'a_word_lengths' = {word : len(word) for word in words if word.startswith('a')} : Pour chaque mot, il est vérifié si le mot commence par la lettre "a" en utilisant la condition word.startswith('a'). Si la condition est vraie, le mot est inclus dans le dictionnaire. La clé est le mot lui-même, et la valeur est la longueur du mot, calculée à l'aide de la fonction len(word). print(words) : Cette ligne affiche la liste originale des mots sur la console. print(a_word_lengths) : Cette ligne affiche le dictionnaire a_word_lengths, qui contient les mots commençant par "a" comme clés et leurs longueurs correspondantes comme valeurs.

Question 25

Créer un dictionnaire de chaînes de caractères en répétant chaque mot trois fois

Exemple de résultat
['apple', 'banana', 'cherry']
{'apple' : 'appleappleapple', 'banana' : 'bananabanabanana', 'cherry' : 'cherrycherry-cherry'}.

Code python :

```python
words = ['apple', 'banana', 'cherry']
repeated_words = {word: word * 3 for word in words}
print(words)
print(repeated_words)
```

q684.py

Crée un dictionnaire mots_répétés, dont les clés sont des mots d'une liste et les valeurs sont ces mots répétés trois fois. Voici comment fonctionne le code :
words = ['apple', 'banana', 'cherry'] : Cette ligne initialise une liste nommée words avec trois valeurs de chaîne. repeated_words = {word : word * 3 for word in words} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire repeated_words. Elle parcourt chaque mot de la liste words et crée une paire clé-valeur pour chaque mot. for word in words : Cette partie du code parcourt chaque mot de la liste des mots. 'repeated_words' = {word : word * 3 for word in words} : Pour chaque mot, le code prend le mot lui-même comme clé et utilise la multiplication de chaîne (mot * 3) pour répéter le mot trois fois, créant ainsi la valeur. print(words) : Cette ligne affiche la liste originale des mots sur la console. print(mots_répétés) : Cette ligne affiche le dictionnaire mots_répétés, qui contient les mots comme clés et les mots répétés trois fois comme valeurs.

Question 26

Créer un dictionnaire de chaînes avec les mots contenant "a" et leur longueur
Exemple de résultat
['apple', 'banana', 'cherry', 'date']
{'apple' : 5, "banane" : 6, "date" : 4}

Code python :

```python
words = ['apple', 'banana', 'cherry', 'date']
a_word_lengths = {word: len(word) for word in words if 'a' in word}
print(words)
print(a_word_lengths)
```

q685.py

Crée un dictionnaire a_word_lengths, dont les clés sont les mots d'une liste contenant la lettre "a" et les valeurs sont les longueurs de ces mots. Voici comment fonctionne le code :
words = ['apple', 'banana', 'cherry', 'date'] : Cette ligne initialise une liste nommée words avec quatre valeurs de chaîne. a_word_lengths = {word : len(word) for word

in words if 'a' in word} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire a_word_lengths. Elle parcourt chaque mot de la liste des mots, vérifie si la lettre "a" est présente dans le mot et crée une paire clé-valeur pour les mots qui contiennent "a". pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. 'a_word_lengths' = {word : len(word) for word in words if 'a' in word} : Pour chaque mot, il prend le mot lui-même comme clé et utilise la fonction len(word) pour calculer la longueur du mot, créant ainsi la valeur. if 'a' in word : Cette condition garantit que seuls les mots contenant la lettre 'a' sont inclus dans le dictionnaire. print(words) : Cette ligne affiche la liste originale des mots sur la console. print(a_word_lengths) : Cette ligne affiche le dictionnaire a_word_lengths, qui contient les mots (avec 'a') comme clés et leurs longueurs respectives comme valeurs.

Question 27

Créer un dictionnaire des nombres et de leurs carrés, mais uniquement pour les nombres supérieurs à 5

Exemple de résultat

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

{6 : 36, 7 : 49, 8 : 64, 9 : 81, 10 : 100}

Code python :

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
squares_greater_than_5 = {x: x ** 2 for x in numbers if x > 5}
print(numbers)
print(squares_greater_than_5)
```

q686.py

Crée un dictionnaire nommé squares_greater_than_5, où les clés sont des nombres d'une liste qui sont supérieurs à 5, et les valeurs sont les carrés de ces nombres. Voici comment fonctionne le code :

nombres = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] : Cette ligne initialise une liste nommée numbers avec dix valeurs entières. carrés_plus_que_5 = {x : x ** 2 for x in numbers if x > 5} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire squares_greater_than_5. Elle parcourt chaque nombre de la liste des nombres et vérifie s'il est supérieur à 5. Si la condition est remplie, elle crée une paire clé-valeur dans le dictionnaire. pour x dans nombres : Cette partie du code parcourt chaque nombre de la liste des nombres. carrés_plus_que_5 = {x : x ** 2 for x in numbers if x > 5} : Pour chaque nombre, le code prend le nombre lui-même comme clé et calcule le carré du nombre à l'aide de l'expression x ** 2, créant ainsi la valeur. if x > 5 : Cette condition garantit que seuls les nombres supérieurs à 5 sont inclus dans le dictionnaire. print(numbers) : Cette ligne imprime la liste originale des nombres sur la console. print(carrés_plus_grands_que_5) : Cette ligne affiche le dictionnaire squares_greater_than_5, qui contient les nombres supérieurs à 5 comme clés et leurs carrés correspondants comme valeurs.

Question 28

Créer un dictionnaire de caractères et de leurs valeurs ASCII à partir d'une chaîne de caractères, à l'exclusion des caractères non alphabétiques.

Exemple de sortie

Bonjour123

{'H' : 72, 'e' : 101, 'l' : 108, 'o' : 111}

Code python :

```python
text = "Hello123"
ascii_values = {char: ord(char) for char in text if char.isalpha()}
print(text)
print(ascii_values)
```

q687.py

Crée un dictionnaire nommé ascii_values pour faire correspondre les caractères alphabétiques de la variable texte à leurs valeurs ASCII. Voici comment fonctionne le code :

text = "Hello123" : Cette ligne initialise la variable texte avec la chaîne "Hello123". ascii

values = {char : ord(char) for char in text if char.isalpha()} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire ascii_values. Elle parcourt chaque caractère (char) de la chaîne de texte et vérifie si le caractère est alphabétique à l'aide de la méthode char.isalpha(). Si le caractère est alphabétique, il crée une paire clé-valeur dans le dictionnaire. pour char dans text : Cette partie du code parcourt chaque caractère de la chaîne de texte. ascii_values = {char : ord(char) for char in text if char.isalpha()} : Pour chaque caractère alphabétique, il prend le caractère lui-même comme clé et récupère sa valeur ASCII à l'aide de la fonction ord(char), créant ainsi la valeur. if char.isalpha() : Cette condition garantit que seuls les caractères alphabétiques sont inclus dans le dictionnaire. print(text) : Cette ligne imprime la chaîne de texte originale sur la console. print(ascii_values) : Cette ligne affiche le dictionnaire ascii_values, qui contient des caractères alphabétiques comme clés et leurs valeurs ASCII correspondantes comme valeurs.

Question 29

Créer un dictionnaire de mots et de leurs formes en majuscules, en excluant les mots sans majuscules.

Exemple de résultat

['apple', 'Banana', 'cherry', 'Date']

{'Banane' : 'BANANE', 'Date' : 'DATE'}

Code python :

```python
words = ['apple', 'Banana', 'cherry', 'Date']
uppercase_words = {word: word.upper() for word in words if
    ↪  any(char.isupper() for char in word)}
print(words)
print(uppercase_words)
```

q688.py

Crée un dictionnaire nommé uppercase_words qui associe les mots comportant au moins un caractère en majuscule à leur version en majuscule. Voici comment fonctionne le code :

words = ['apple', 'Banana', 'cherry', 'Date'] : Cette ligne initialise la liste des mots avec une collection de mots, dont certains contiennent des caractères majuscules. uppercase_words = {word : word.upper() for word in words if any(char.isupper() for char in word)} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire uppercase_words. Elle parcourt chaque mot de la liste des mots et vérifie s'il contient au moins un caractère majuscule (char.isupper()). Si le mot contient au moins une majuscule, il crée une paire clé-valeur dans le dictionnaire. pour mot dans mots : Cette partie du code parcourt chaque mot de la liste des mots. if any(char.isupper() for char in word) : Cette condition vérifie s'il existe au moins un caractère majuscule dans le mot actuel. La fonction any vérifie si l'un des caractères du mot est en majuscule. word : word.upper() for word in words if any(char.isupper() for char in word) : Si la condition est remplie, une paire clé-valeur est créée dans le dictionnaire. La clé est le mot original, et la valeur est la version en majuscules de ce mot obtenue à l'aide de word.upper(). print(words) : Cette ligne affiche la liste des mots originaux sur la console. print(mots_en_majuscules) : Cette ligne affiche le dictionnaire uppercase_words, qui contient des mots comportant au moins un caractère majuscule comme clés et leurs versions majuscules comme valeurs.

Question 30

Créer un dictionnaire de chaînes de mots contenant plus de 4 lettres

Exemple de résultat

Python est un langage de programmation puissant et polyvalent.

{'Python' : 6, 'puissant' : 8, 'polyvalent' : 9, "programmation" : 11, "langage" : 8}

Code python :

```python
sentence = "Python is a powerful and versatile programming language"
long_words = {word: len(word) for word in sentence.split() if len(word)
    ↪  > 4}
print(sentence)
print(long_words)
```

q689.py

Le code Python présente une phrase et crée un dictionnaire appelé mots_longs. Ce dictionnaire associe les mots d'une longueur supérieure à 4 à leurs longueurs respectives. Voici comment fonctionne le code :

phrase = "Python est un langage de programmation puissant et polyvalent" : Cette ligne initialise la variable phrase avec une chaîne contenant des mots. long_words = {word : len(word) for word in sentence.split() if len(word) > 4} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire long_words. Elle divise la phrase en mots à l'aide de sentence.split(), puis parcourt chaque mot de la liste des mots. for word in sentence.split() : Cette partie du code parcourt chaque mot de la phrase. if len(word) > 4 : cette condition vérifie si la longueur du mot actuel est supérieure à 4 caractères. mot : len(mot) : Si la condition est remplie, une paire clé-valeur est créée dans le dictionnaire. La clé est le mot lui-même, et la valeur est la longueur du mot obtenue à l'aide de len(mot). print(phrase) : Cette ligne affiche la phrase originale sur la console. print(mots_longs) : Cette ligne affiche le dictionnaire mots_longs, qui contient des mots de longueur supérieure à 4 comme clés et leurs longueurs respectives comme valeurs.

Question 31

Créer un dictionnaire des mots et de leur première lettre en majuscule
Exemple de résultat
['apple', 'banana', 'cherry']
{'apple' : 'Apple', 'banana' : 'Banana', 'cherry' : 'Cherry'}

Code python :

```python
words = ['apple', 'banana', 'cherry']
capitalized_first_letter = {word: word[0].upper() + word[1:] for word
↪  in words}
print(words)
print(capitalized_first_letter)
```

q690.py

La liste des mots et crée un dictionnaire appelé capitalized_first_letter. Ce dictionnaire associe chaque mot à une version du mot dont la première lettre est en majuscule. Voici comment fonctionne le code :
words = ['apple', 'banana', 'cherry'] : Cette ligne initialise la variable words avec une liste de mots. première_lettre_capitalisée = {mot : mot[0].upper() + mot[1 :] pour mot dans mots} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire capitalized_first_letter. Elle parcourt chaque mot de la liste des mots. {mot : mot[0].upper() + mot[1 :]} : Cette partie du code crée des paires clé-valeur dans le dictionnaire. La clé est le mot original et la valeur est une version modifiée du mot dont la première lettre est en majuscule. Pour ce faire, on utilise word[0].upper() pour mettre la première lettre en majuscule, puis on la concatène avec le reste du mot obtenu à l'aide de word[1 :]. print(words) : Cette ligne imprime la liste originale de mots, words, sur la console. print(première_lettre_capitalisée) : Cette ligne affiche le dictionnaire capitalized_first_letter, qui contient les mots originaux comme clés et leurs versions avec la première lettre en majuscule comme valeurs.

Question 32

Créer un dictionnaire de chaînes de caractères dont les voyelles ont été supprimées
Exemple de résultat
['apple', 'banana', 'cherry']
{'apple' : 'ppl', 'banana' : 'bnn', 'cherry' : 'chrry'}

Code python :

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

q691.py

La liste des mots et crée un dictionnaire nommé without_vowels. Ce dictionnaire associe chaque mot à une version du mot dans laquelle toutes les voyelles (majuscules et minuscules) ont été supprimées. Voici comment fonctionne le code :
words = ['apple', 'banana', 'cherry'] : Cette ligne initialise la variable words avec une liste de mots. without_vowels = {word : ''.join([char for char in word if char.lower() not in 'aeiou']) for word in words} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire sans_voyelles. Elle parcourt chaque mot de la liste des mots. {mot : ''.join([char for char in word if char.lower() not in 'aeiou'])} : Cette partie du code crée des paires clé-valeur dans le dictionnaire. La clé est le mot original et la valeur est une version modifiée du mot. La modification est effectuée en parcourant chaque caractère char dans le mot et en vérifiant si char.lower() (la version minuscule du caractère) n'est pas dans la chaîne 'aeiou', qui représente les voyelles. Si char n'est pas une voyelle, il est inclus dans la version modifiée du mot. La fonction join est utilisée pour concaténer les caractères en une seule chaîne. print(words) : Cette ligne imprime la liste originale des mots, words, sur la console. print(sans_voyelles) : Cette ligne affiche le dictionnaire without_vowels, qui contient les mots originaux comme clés et leurs versions sans voyelles comme valeurs.

Question 33

Créer un dictionnaire des nombres avec leurs signes inversés
Exemple de résultat
[5, -10, 15, -20, 25]
{5 : -5, -10 : 10, 15 : -15, -20 : 20, 25 : -25}

Code python :

```python
numbers = [5, -10, 15, -20, 25]
reversed_signs = {x: -x for x in numbers}
print(numbers)
print(reversed_signs)
```

q692.py

La liste des nombres et crée un dictionnaire appelé signes_inversés. Ce dictionnaire associe chaque nombre à sa négation (en changeant son signe de positif à négatif ou vice versa). Voici comment fonctionne le code :

nombres = [5, -10, 15, -20, 25] : Cette ligne initialise la variable numbers avec une liste d'entiers, comprenant à la fois des nombres positifs et négatifs. signes_inversés = {x : -x for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire signes_inversés. Elle parcourt chaque nombre x dans la liste des nombres. {x : -x} : Cette partie du code crée des paires clé-valeur dans le dictionnaire. La clé est le nombre original x, et la valeur est la négation de ce nombre, qui est calculée comme -x. print(nombres) : Cette ligne imprime la liste originale des nombres, numbers, sur la console. print(signes_inversés) : Cette ligne affiche le dictionnaire signes_inversés, qui contient les nombres originaux comme clés et leurs négations comme valeurs.

Question 34

Créer un dictionnaire de chaînes de caractères avec des mots inversés

Exemple de résultat

Python est amusant

{0 : 'fun', 1 : 'is', 2 : 'Python'}

Code python :

```python
sentence = "Python is fun"
reversed_words = {i: word for i, word in
↪  enumerate(sentence.split()[::-1])}
print(sentence)
print(reversed_words)
```

q693.py

Le code Python traite une phrase et crée un dictionnaire appelé mots_inversés. Ce dictionnaire associe l'index (position) de chaque mot dans l'ordre inverse au mot lui-même. Voici comment fonctionne le code :

phrase = "Python is fun" : Cette ligne initialise la variable phrase avec une chaîne contenant plusieurs mots. reversed_words = {i : word for i, word in enumerate(sentence.split()[: :-1])} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire mots_inversés. Voici ce qui se passe : sentence.split() : Cette partie du code divise la phrase en mots individuels et renvoie une liste de mots. [: :-1] : Cette partie inverse l'ordre des mots dans la liste. Elle inverse donc l'ordre des mots dans la phrase. enumerate(...) : Cette fonction est utilisée pour parcourir la liste inversée des mots

et renvoie à la fois l'index (i) et le mot lui-même. {i : mot pour i, mot dans ...} : Cette partie du code crée des paires clé-valeur dans le dictionnaire. Les clés (i) sont les indices des mots, et les valeurs (mot) sont les mots individuels de la liste inversée. print(phrase) : Cette ligne imprime la phrase originale, sentence, sur la console. print(mots_inversés) : Cette ligne affiche le dictionnaire mots_inversés, qui contient les indices des mots comme clés (dans l'ordre inverse) et les mots correspondants comme valeurs.

Question 35

Créer un dictionnaire des nombres avec leurs diviseurs

Exemple de résultat

[1, 2, 3, 4, 5]

{1 : [1], 2 : [1, 2], 3 : [1, 3], 4 : [1, 2, 4], 5 : [1, 5]}

Code python :

```python
numbers = [1, 2, 3, 4, 5]
divisors = {x: [i for i in range(1, x + 1) if x % i == 0] for x in
    numbers}
print(numbers)
print(divisors)
```

q694.py

Crée un dictionnaire nommé diviseurs qui associe chaque nombre de la liste des nombres à une liste de ses diviseurs. Voici comment fonctionne le code :

nombres = [1, 2, 3, 4, 5] : Cette ligne initialise la liste des nombres avec une séquence d'entiers. diviseurs = {x : [i for i in range(1, x + 1) if x % i == 0] for x in numbers} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire des diviseurs. Voici ce qui se passe : for x in numbers : Cette partie du code itère sur chaque nombre x dans la liste des nombres. {x : ...} : Cette partie du code crée des paires clé-valeur dans le dictionnaire. La clé est le nombre x et la valeur est le résultat de la compréhension de la liste intérieure. [i for i in range(1, x + 1) if x % i == 0] : Il s'agit de la compréhension de liste qui génère une liste de diviseurs pour chaque nombre x. Elle parcourt les nombres de 1 à x (inclus) et vérifie si x est divisible par i. Si c'est le cas, i est inclus dans la liste des diviseurs. print(nombres) : Cette ligne imprime la liste des nombres sur la console. print(diviseurs) : Cette ligne affiche le dictionnaire des diviseurs, qui contient des nombres comme clés et des listes de diviseurs comme valeurs.

Question 36

Créer un dictionnaire de caractères et leur nombre, à l'exclusion des caractères d'espacement, à partir d'une chaîne de caractères.

Exemple de sortie

bonjour le monde

{'o' : 2, 'h' : 1, 'r' : 1, 'w' : 1, 'e' : 1, 'l' : 3, 'd' : 1}

Code python :

```python
text = "hello world"
char_counts = {char: text.count(char) for char in set(text) if not
    ↪  char.isspace()}
print(text)
print(char_counts)
```

q695.py

Il compte les occurrences de caractères non spatiaux dans le texte donné et crée un dictionnaire qui associe chaque caractère à son nombre. Voici comment fonctionne le code :
text = "hello world" : Cette ligne initialise la variable texte avec le texte d'entrée. char_counts = {char : text.count(char) for char in set(text) if not char.isspace()} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire char_counts. Voici ce qui se passe : {char : ...} : Cette partie du code crée des paires clé-valeur dans le dictionnaire. La clé est un caractère char, et la valeur est le résultat de l'expression text.count(char), qui compte les occurrences de char dans le texte. for char in set(text) : Cette partie itère sur chaque caractère unique du texte. L'expression set(text) est utilisée pour obtenir un ensemble de caractères uniques, en éliminant les doublons. if not char.isspace() : Cette partie vérifie si le caractère char n'est pas un caractère d'espacement. Elle permet de s'assurer que seuls les caractères non spatiaux sont comptés. print(text) : Cette ligne imprime le texte original sur la console. print(char_counts) : Cette ligne affiche le dictionnaire char_counts, qui contient des caractères non spatiaux comme clés et leur nombre comme valeurs.

Question 37

Correspondance entre les lettres minuscules et leurs valeurs ASCII
Exemple de sortie
{'a' : 97, 'b' : 98, 'c' : 99, 'd' : 100, 'e' : 101, 'f' : 102, "g" : 103, "h" : 104, 'i' : 105, 'j' : 106, 'k' : 107, "l" : 108, "m" : 109, "n" : 110, "o" : 111, "p" : 112, 'q' : 113, 'r' : 114, 's' : 115, "t" : 116, "u" : 117, 'v' : 118, 'w' : 119, 'x' : 120, 'y' : 121, 'z' : 122}

Code python :

```python
ascii_mapping = {char: ord(char) for char in
    ↪  'abcdefghijklmnopqrstuvwxyz'}
print(ascii_mapping)
```

q696.py

Crée un dictionnaire appelé ascii_mapping qui associe chaque lettre minuscule de l'alphabet anglais à la valeur ASCII correspondante. Voici ce que fait le code :
ascii_mapping = {char : ord(char) for char in 'abcdefghijklmnopqrstuvwxyz'} : Cette ligne crée le dictionnaire ascii_mapping en utilisant une compréhension de dictionnaire. Voici comment cela fonctionne : {char : ord(char) for char in 'abcdefghijklm-

nopqrstuvwxyz'} est la compréhension du dictionnaire. Il parcourt chaque caractère (char) de la chaîne 'abcdefghijklmnopqrstuvwxyz'. Pour chaque caractère, il crée une paire clé-valeur dans le dictionnaire. La clé (char) est le caractère lui-même, et la valeur (ord(char)) est la valeur ASCII de ce caractère obtenue à l'aide de la fonction ord.

Question 38

Nombres et leur factorielle de 1 à 10
Exemple de résultat
{1 : 1, 2 : 2, 3 : 6, 4 : 24, 5 : 120, 6 : 720, 7 : 5040, 8 : 40320, 9 : 362880, 10 : 3628800}

Code python :

```python
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

factorials = {x: factorial(x) for x in range(1, 11)}
print(factorials)
```

q697.py

Le code définit une fonction Python factorial(n) pour calculer la factorielle d'un nombre n à l'aide de la récursivité, puis utilise cette fonction pour créer un dictionnaire appelé factorials. Ce dictionnaire associe les nombres de 1 à 10 à leurs valeurs factorielles respectives. Voici une explication pas à pas du code :
def factorial(n) : Cette ligne définit une fonction nommée factorial qui prend un entier n en entrée et calcule la factorielle de n en utilisant la récursivité. Le cas de base est défini comme suit : if n == 0 :. Lorsque n est 0, la fonction renvoie 1 car 0 ! (lire "factorielle zéro") est défini comme 1. Dans le cas récursif, la fonction calcule la factorielle de n comme n * factorielle(n - 1). Elle s'appelle récursivement avec une valeur plus petite jusqu'à ce qu'elle atteigne le cas de base. factorials = {x : factorial(x) for x in range(1, 11)} : Cette ligne crée le dictionnaire factoriel en utilisant une compréhension de dictionnaire. Voici comment cela fonctionne : x : factorial(x) for x in range(1, 11) est la compréhension du dictionnaire. Elle parcourt la plage de nombres de 1 à 10 (inclus). Pour chaque nombre x, il crée une paire clé-valeur dans le dictionnaire. La clé (x) est le nombre lui-même, et la valeur (factorielle(x)) est calculée en appelant la fonction factorielle avec ce nombre en entrée. print(factorials) : Cette ligne affiche le dictionnaire factoriel sur la console.

Question 39

Les nombres et leur représentation binaire de 1 à 10
Exemple de sortie
{1 : '1', 2 : '10', 3 : '11', 4 : '100', 5 : '101', 6 : '110', 7 : '111', 8 : '1000', 9 : '1001', 10 : '1010'}

Code python :

```python
binary_rep = {x: bin(x)[2:] for x in range(1, 11)}
print(binary_rep)
```

q698.py

Crée un dictionnaire appelé binary_rep en utilisant une compréhension de dictionnaire en Python. Ce dictionnaire associe les nombres de 1 à 10 à leurs représentations binaires respectives sous forme de chaînes de caractères. Voici une explication pas à pas du code :
binary_rep = {x : bin(x)[2 :] for x in range(1, 11)} : Cette ligne crée le dictionnaire binary_rep à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : {x : bin(x)[2 :] for x in range(1, 11)} est la compréhension du dictionnaire. Elle parcourt la plage de nombres de 1 à 10 (inclus). Pour chaque nombre x, il crée une paire clé-valeur dans le dictionnaire. La clé (x) est le nombre lui-même, et la valeur (bin(x)[2 :]) est calculée en convertissant x en sa représentation binaire à l'aide de la fonction bin, puis en supprimant les deux premiers caractères (qui sont '0b') de la chaîne binaire pour obtenir la représentation binaire sous la forme d'une chaîne. print(binary_rep) : Cette ligne affiche le dictionnaire binary_rep sur la console.

Question 40

Paires d'éléments distincts et leur différence absolue à partir de deux listes
Exemple de résultat
[3, 6, 9]
[5, 10, 15]
{(3, 5) : 2, (3, 10) : 7, (3, 15) : 12, (6, 5) : 1, (6, 10) : 4, (6, 15) : 9, (9, 5) : 4, (9, 10) : 1, (9, 15) : 6}

Code python :

```python
list1 = [3, 6, 9]
list2 = [5, 10, 15]
abs_diff_dict = {(x, y): abs(x - y) for x in list1 for y in list2}
print(list1)
print(list2)
print(abs_diff_dict)
```

q699.py

Crée un dictionnaire nommé abs_diff_dict. Ce dictionnaire associe des paires d'éléments de list1 et list2 à leurs différences absolues. Voici comment fonctionne le code : list1 = [3, 6, 9] et list2 = [5, 10, 15] : Ces lignes initialisent deux listes, list1 et list2, avec des valeurs entières. abs_diff_dict = {(x, y) : abs(x - y) for x in list1 for y in list2} : Cette ligne utilise une compréhension de dictionnaire pour créer le dictionnaire abs_diff_dict. Voici ce qui se passe : (x, y) est utilisé comme un tuple pour représenter des paires d'éléments, où x est un élément de la liste 1 et y est un élément de la liste 2. abs(x - y) calcule la différence absolue entre x et y pour chaque

paire. La compréhension du dictionnaire passe en revue toutes les paires possibles d'éléments de list1 et list2 et calcule la différence absolue pour chaque paire, en utilisant le tuple (x, y) comme clé et la différence absolue comme valeur. print(list1) : Cette ligne affiche le contenu de la liste 1 sur la console. print(list2) : Cette ligne affiche le contenu de la liste 2 sur la console. print(abs_diff_dict) : Cette ligne affiche le dictionnaire abs_diff_dict, qui contient des paires d'éléments comme clés et leurs différences absolues comme valeurs.

Question 41

Paires d'éléments distincts et leur position de caractère additionnés à partir de deux listes

Exemple de résultat

['abc', 'def', 'ghi']

['jkl', 'mno', 'pqr']

{('abc', 'jkl') : 615, ('abc', 'mno') : 624, ('abc', 'pqr') : 633, ('def', 'jkl') : 624, ('def', 'mno') : 633, ("def", "pqr") : 642, ('ghi', 'jkl') : 633, ('ghi', 'mno') : 642, ('ghi', 'pqr') : 651}

Code python :

```python
list1 = ["abc", "def", "ghi"]
list2 = ["jkl", "mno", "pqr"]
char_pos_sum_dict = {(x, y): sum(ord(char) for char in x) +
    ↪  sum(ord(char) for char in y) for x in list1 for y in list2}
print(list1)
print(list2)
print(char_pos_sum_dict)
```

q700.py

Crée un dictionnaire appelé char_pos_sum_dict à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe des paires de chaînes de caractères provenant de deux listes, list1 et list2, à la somme des valeurs ordinales (valeurs ASCII) des caractères de chaque paire. Voici une explication pas à pas du code :

list1 = ["abc", "def", "ghi"] : Cette ligne définit une liste appelée list1 contenant trois chaînes de caractères. list2 = ["jkl", "mno", "pqr"] : Cette ligne définit une autre liste appelée list2 contenant trois chaînes de caractères. char_pos_sum_dict = {(x, y) : sum(ord(char) for char in x) + sum(ord(char) for char in y) for x in list1 for y in list2} : Cette ligne crée le dictionnaire char_pos_sum_dict à l'aide d'une compréhension de dictionnaire imbriqué. Voici comment cela fonctionne : {(x, y) : sum(ord(char) for char in x) + sum(ord(char) for char in y) for x in list1 for y in list2} est la compréhension du dictionnaire. Il itère sur des paires de chaînes (x, y) où x provient de list1 et y de list2. Pour chaque paire de chaînes, il crée une paire clé-valeur dans le dictionnaire. La clé x, y est un tuple contenant les deux chaînes. La valeur est calculée en additionnant les valeurs ordinales (valeurs ASCII) de tous les caractères de la première chaîne x et de tous les caractères de la deuxième chaîne y. print(list1) : Cette ligne imprime le contenu de la liste 1 sur la console. print(list2) :

Cette ligne affiche le contenu de la liste 2 sur la console. print(char_pos_sum_dict) : Cette ligne affiche le dictionnaire char_pos_sum_dict sur la console.

Question 42

Mots distincts et leur longueur, à l'exclusion des mots de longueur impaire, dans une phrase

Exemple de sortie

Bonjour, comment allez-vous ?

{'Hello,' : 6, 'you ?' : 4}

Code python :

```python
sentence = "Hello, how are you?"
distinct_word_length_no_odd = {word: len(word) for word in
    ↪  set(sentence.split()) if len(word) % 2 == 0}
print(sentence)
print(distinct_word_length_no_odd)
```

q701.py

Le code traite une phrase et crée un dictionnaire nommé distinct_word_length_no_odd à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les mots distincts de la phrase à leurs longueurs respectives, mais uniquement pour les mots de longueur paire. Voici une explication pas à pas du code :

sentence = "Hello, how are you ?" : Cette ligne définit une variable de type chaîne appelée sentence contenant la phrase "Hello, how are you ?" distinct_word_length_no_odd = {word : len(word) for word in set(sentence.split()) if len(word) % 2 == 0} : Cette ligne crée le dictionnaire distinct_word_length_no_odd en utilisant une compréhension de dictionnaire. Voici comment cela fonctionne : {word : len(word) for word in set(sentence.split()) if len(word) % 2 == 0} est la compréhension du dictionnaire. Il effectue les étapes suivantes : sentence.split() divise la phrase en une liste de mots. Dans ce cas, elle inclura des mots comme "Hello," (avec une virgule) et "you ?" (avec un point d'interrogation). (avec un point d'interrogation). set(sentence.split()) crée un ensemble de mots distincts en supprimant les doublons et la ponctuation. Il contiendra donc des mots comme "Hello," (sans la virgule) et "you" (sans le point d'interrogation). for word in set(sentence.split()) itère sur chaque mot distinct de l'ensemble. len(word) % 2 == 0 vérifie si la longueur du mot est paire. Si la longueur d'un mot est paire, il crée une paire clé-valeur dans le dictionnaire. La clé (mot) est le mot lui-même, et la valeur (len(mot)) est la longueur du mot. print(phrase) : Cette ligne imprime la phrase originale, qui est "Hello, how are you ?", sur la console. print(distinct_word_length_no_odd) : Cette ligne affiche le dictionnaire distinct_mot_longueur_no_odd sur la console.

Question 43

Mots distincts et leur longueur, à l'exclusion des mots de longueur paire, dans une

phrase
Exemple de sortie
Bonjour, comment allez-vous ?
{'how' : 3, 'are' : 3}

Code python :

```python
sentence = "Hello, how are you?"
distinct_word_length_no_even = {word: len(word) for word in
    ↪  set(sentence.split()) if len(word) % 2 != 0}
print(sentence)
print(distinct_word_length_no_even)
```

q702.py

Le code traite une phrase et crée un dictionnaire nommé distinct_word_length_no_even à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les mots distincts de la phrase à leurs longueurs respectives, mais uniquement pour les mots de longueur impaire. Voici une explication pas à pas du code :
sentence = "Hello, how are you ?" : Cette ligne définit une variable de type chaîne appelée sentence contenant la phrase "Hello, how are you ?" distinct_word_length_no_even = {word : len(word) for word in set(sentence.split()) if len(word) % 2 != 0} : Cette ligne crée le dictionnaire distinct_mot_longueur_même à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : {word : len(word) for word in set(sentence.split()) if len(word) % 2 != 0} est la compréhension du dictionnaire. Il effectue les étapes suivantes : sentence.split() divise la phrase en une liste de mots. Dans ce cas, elle inclura des mots comme "Hello," (avec une virgule) et "you ?" (avec un point d'interrogation). (avec un point d'interrogation). set(sentence.split()) crée un ensemble de mots distincts en supprimant les doublons et la ponctuation. Il contiendra donc des mots comme "Hello," (sans la virgule) et "you" (sans le point d'interrogation). for word in set(sentence.split()) itère sur chaque mot distinct de l'ensemble. len(word) % 2 != 0 vérifie si la longueur du mot est impaire. Si la longueur d'un mot est impaire, cela crée une paire clé-valeur dans le dictionnaire. La clé (mot) est le mot lui-même, et la valeur (len(mot)) est la longueur du mot. print(phrase) : Cette ligne imprime la phrase originale, qui est "Hello, how are you ?", sur la console. print(distinct_word_length_no_even) : Cette ligne affiche le dictionnaire distinct_mot_longueur_non_pair sur la console.

Question 44

Caractères et nombre d'occurrences, hors ponctuation, dans une phrase
Exemple de sortie
Bonjour, comment allez-vous ?
{'w' : 1, 'r' : 1, 'u' : 1, 'o' : 3, 'e' : 2, 'H' : 1, 'l' : 2, 'h' : 1, ' ' : 3, 'y' : 1, 'a' : 1}

Code python :

```python
import string

sentence = "Hello, how are you?"
char_occurrence_no_punct = {char: sentence.count(char) for char in
    ↪  set(sentence) if char not in string.punctuation}
print(sentence)
print(char_occurrence_no_punct)
```

q703.py

Le code traite une phrase et crée un dictionnaire nommé char_occurrence_no_punct à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les caractères présents dans la phrase à leur nombre respectif, à l'exclusion des caractères de ponctuation. Voici une explication pas à pas du code :

import string : Cette ligne importe le module string, qui fournit une chaîne contenant tous les caractères de ponctuation ASCII. sentence = "Hello, how are you ?" : Cette ligne définit une variable de type chaîne de caractères appelée sentence contenant la phrase "Hello, how are you ?" char_occurrence_no_punct = {char : sentence.count(char) for char in set(sentence) if char not in string.punctuation} : Cette ligne crée le dictionnaire char_occurrence_no_punct à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : {char : sentence.count(char) for char in set(sentence) if char not in string.punctuation} est la compréhension du dictionnaire. Il exécute les étapes suivantes : set(sentence) crée un ensemble de caractères distincts à partir de la phrase. for char in set(sentence) itère sur chaque caractère distinct de l'ensemble. char not in string.punctuation vérifie si le caractère n'est pas dans la chaîne string.punctuation, qui contient tous les caractères de ponctuation. Si un caractère n'est pas un caractère de ponctuation, il crée une paire clé-valeur dans le dictionnaire. La clé (char) est le caractère lui-même, et la valeur (sentence.count(char)) est le nombre de fois où ce caractère apparaît dans la phrase. print(phrase) : Cette ligne imprime la phrase originale, qui est "Hello, how are you ?", sur la console. print(char_occurrence_no_punct) : Cette ligne affiche le dictionnaire char_occurrence_no_punct sur la console.

Question 45

Mots distincts et leur longueur, à l'exclusion de ceux dont la longueur n'est pas de Fibonacci, dans une phrase

Exemple de sortie

Bonjour, comment allez-vous ?

{'are' : 3, 'how' : 3}

Code python :

```python
def is_fibonacci_length(n):
    fib = [0, 1]
    while fib[-1] <= n:
        if fib[-1] == n:
            return True
        fib.append(fib[-1] + fib[-2])
    return False

sentence = "Hello, how are you?"
distinct_word_length_fibonacci = {word: len(word) for word in
    ↪ set(sentence.split()) if is_fibonacci_length(len(word))}
print(sentence)
print(distinct_word_length_fibonacci)
```

q704.py

Le code définit une fonction is_fibonacci_length(n) pour vérifier si un nombre donné est un nombre de Fibonacci (en vérifiant si sa longueur fait partie de la séquence de Fibonacci). Il utilise ensuite cette fonction pour créer un dictionnaire nommé distinct_word_length_fibonacci à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les mots distincts de la phrase à leurs longueurs respectives, mais uniquement pour les mots dont les longueurs sont des nombres de Fibonacci. Voici une explication pas à pas du code :

def is_fibonacci_length(n) : Cette ligne définit une fonction nommée is_fibonacci_length qui prend un entier n en entrée et vérifie si n est un nombre de Fibonacci en comparant sa longueur à la séquence de Fibonacci. Dans la fonction, une liste fib est initialisée avec les deux premiers nombres de Fibonacci (0 et 1). Une boucle while vérifie si le dernier nombre de la liste fib (fib[-1]) est inférieur ou égal à n. Elle continue à calculer les nombres de Fibonacci jusqu'à ce que le dernier soit inférieur ou égal à n. Si fib[-1] est égal à n, la fonction renvoie True car n est un nombre de Fibonacci en raison de sa longueur. Si la boucle se termine sans trouver de nombre de Fibonacci de la longueur donnée, la fonction renvoie False. sentence = "Hello, how are you ?" : Cette ligne définit une variable de type chaîne de caractères appelée sentence contenant la phrase "Hello, how are you ?" distinct_word_length_fibonacci = word : len(word) for word in set(sentence.split()) if is_fibonacci_length(len(word)) : Cette ligne crée le dictionnaire distinct_mot_longueur_fibonacci à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : {word : len(word) for word in set(sentence.split()) if is_fibonacci_length(len(word)} est la compréhension du dictionnaire. Il effectue les étapes suivantes : sentence.split() divise la phrase en une liste de mots. set(sentence.split()) crée un ensemble de mots distincts en supprimant les doublons. for word in set(sentence.split()) itère sur chaque mot distinct de l'ensemble. is_fibonacci_length(len(word)) vérifie si la longueur du mot est un nombre de Fibonacci à l'aide de la fonction is_fibonacci_length. Si la longueur d'un mot est un nombre de Fibonacci, une paire clé-valeur est créée dans le dictionnaire. La clé (mot) est le mot lui-même, et la valeur (len(mot)) est la longueur du mot. print(phrase) : Cette ligne imprime la phrase originale, qui est "Hello, how are you ?", sur la console. print(longueur_de_mot_distinct_fibonacci) : Cette ligne affiche le dictionnaire distinct_mot_longueur_fibonacci sur la console.

Question 46

Correspondance entre les mots et la fréquence à laquelle ils contiennent la lettre "e".
Exemple de résultat
Bonjour, comment allez-vous ?
{'Hello,' : 1, 'comment' : 0, 'are' : 1, 'you ? 0}

Code python :

```python
sentence = "Hello, how are you?"
word_contains_e = {word: word.count('e') for word in sentence.split()}
print(sentence)
print(word_contains_e)
```

q705.py

Le code traite une phrase et crée un dictionnaire nommé word_contains_e à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les mots de la phrase au nombre de lettres "e" dans chaque mot. Voici une explication pas à pas du code :
sentence = "Hello, how are you ?" : Cette ligne définit une variable de type chaîne appelée sentence contenant la phrase "Hello, how are you ?" word_contains_e = {word : word.count('e') for word in sentence.split()} : Cette ligne crée le dictionnaire word_contains_e à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : {word : word.count('e') for word in sentence.split()} est la compréhension du dictionnaire. Elle effectue les étapes suivantes : sentence.split() divise la phrase en une liste de mots. Dans ce cas, elle inclura des mots comme "Hello," (avec une virgule) et "you ?" (avec un point d'interrogation). (avec un point d'interrogation). for word in sentence.split() parcourt chaque mot de la liste de mots. word.count('e') compte les occurrences de la lettre "e" dans le mot. Il crée une paire clé-valeur dans le dictionnaire, où la clé (mot) est le mot lui-même, et la valeur (mot.count('e')) est le nombre d'occurrences de la lettre 'e' dans le mot. print(phrase) : Cette ligne imprime la phrase originale, qui est "Hello, how are you ?", sur la console. print(mot_contient_e) : Cette ligne affiche le dictionnaire word_contains_e sur la console.

Question 47

Correspondance entre les caractères et leur position dans l'alphabet
Exemple de sortie
{'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5, 'f' : 6, 'g' : 7, 'h' : 8, 'i' : 9, 'j' : 10, 'k' : 11, "l" : 12, "m" : 13, "n" : 14, "o" : 15, "p" : 16, 'q' : 17, 'r' : 18, 's' : 19, "t" : 20, "u" : 21, "v" : 22, "w" : 23, 'x' : 24, 'y' : 25, 'z' : 26}

Code python :

```python
import string

char_alphabet_position = {char: string.ascii_lowercase.index(char) + 1
    for char in string.ascii_lowercase}
print(char_alphabet_position)
```

q706.py

Crée un dictionnaire nommé char_alphabet_position à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les lettres minuscules de l'alphabet anglais à leurs positions respectives dans l'alphabet, en commençant par 1. Voici une explication du code étape par étape :

import string : Cette ligne importe le module string, qui fournit une constante string.ascii_lowercase contenant toutes les lettres minuscules de l'alphabet anglais. char_alphabet_position = char : string.ascii_lowercase.index(char) + 1 for char in string.ascii_lowercase : Cette ligne crée le dictionnaire char_alphabet_position à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : {char : string.ascii_lowercase.index(char) + 1 for char in string.ascii_lowercase} est la compréhension du dictionnaire. Il effectue les étapes suivantes : string.ascii_lowercase est la chaîne contenant toutes les lettres minuscules de l'alphabet anglais : "abcdefghijklmnopqrstuvwxyz". for char in string.ascii_lowercase itère sur chaque lettre minuscule de l'alphabet. string.ascii_lowercase.index(char) trouve l'indice (position) du caractère char dans la chaîne de caractères string.ascii_lowercase. Cet indice est basé sur zéro. + 1 est ajouté à l'index pour le faire passer d'un index basé sur zéro à une position basée sur un, car les positions dans l'alphabet commencent à 1. Cela crée une paire clé-valeur dans le dictionnaire, où la clé (char) est la lettre minuscule, et la valeur (string.ascii_lowercase.index(char) + 1) est sa position dans l'alphabet. print(char_alphabet_position) : Cette ligne affiche le dictionnaire char_alphabet_position sur la console.

Question 48

Mise en correspondance des mots avec leur statut de palindrome dans une phrase
Exemple de sortie
madame voit la voiture de course
{'madam' : True, 'sees' : True, 'the' : False, 'racecar' : True}

Code python :

```python
sentence = "madam sees the racecar"
word_is_palindrome = {word: word == word[::-1] for word in
    sentence.split()}
print(sentence)
print(word_is_palindrome)
```

q707.py

Le code traite une phrase et crée un dictionnaire nommé word_is_palindrome à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les mots de

la phrase à une valeur booléenne indiquant si chaque mot est un palindrome. Voici une explication pas à pas du code :
sentence = "madame voit la voiture de course" : Cette ligne définit une variable de type chaîne appelée phrase contenant la phrase "madame voit la voiture de course". word_is_palindrome = {word : word == word[: :-1] for word in sentence.split()} : Cette ligne crée le dictionnaire word_is_palindrome en utilisant la compréhension du dictionnaire. Voici comment cela fonctionne : {word : word == word[: :-1] for word in sentence.split()} est la compréhension du dictionnaire. Elle effectue les étapes suivantes : sentence.split() divise la phrase en une liste de mots. Dans ce cas, la phrase contient des mots comme "madam", "sees", "the" et "racecar". for word in sentence.split() itère sur chaque mot de la liste de mots. mot == mot[: :-1] vérifie si le mot est un palindrome. Pour ce faire, il compare le mot lui-même (word) à son inverse (word[: :-1]). Si un mot est un palindrome, il crée une paire clé-valeur dans le dictionnaire. La clé (mot) est le mot lui-même, et la valeur (mot == mot[: :-1]) est True, indiquant qu'il s'agit d'un palindrome. Si un mot n'est pas un palindrome, la valeur est False. print(phrase) : Cette ligne imprime la phrase originale, qui est "madame voit la voiture de course", sur la console. print(mot_est_palindrome) : Cette ligne affiche le dictionnaire word_is_palindrome sur la console.

Question 49

Correspondance entre les nombres et leurs représentations binaires et hexadécimales
Exemple de sortie
{1 : ('1', '1'), 2 : ('10', '2'), 3 : ('11', '3'), 4 : ('100', '4'), 5 : ('101', '5'), 6 : ('110', '6'), 7 : ('111', '7'), 8 : ('1000', '8'), 9 : ('1001', '9'), 10 : ('1010', 'a')}

Code python :

```python
number_binary_hex = {num: (bin(num)[2:], hex(num)[2:]) for num in range(1, 11)}
print(number_binary_hex)
```

q708.py

Crée un dictionnaire nommé number_binary_hex en utilisant une compréhension de dictionnaire en Python. Ce dictionnaire associe les nombres de 1 à 10 à leurs représentations binaires et hexadécimales. Voici une explication pas à pas du code :
number_binary_hex = {num : (bin(num)[2 :], hex(num)[2 :]) for num in range(1, 11)} : Cette ligne crée le dictionnaire number_binary_hex à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : {num : (bin(num)[2 :], hex(num)[2 :]) for num in range(1, 11)} est la compréhension du dictionnaire. Il effectue les étapes suivantes : for num in range(1, 11) itère sur les nombres de 1 à 10 (inclus). (bin(num)[2 :], hex(num)[2 :]) est un tuple contenant deux éléments : bin(num)[2 :] convertit le nombre num en sa représentation binaire à l'aide de la fonction bin, puis coupe les deux premiers caractères (qui représentent "0b" au début de la chaîne binaire). hex(num)[2 :] convertit le nombre num en sa représentation hexadécimale à l'aide de la fonction hex, puis coupe les deux premiers caractères (qui représentent "0x" au début de la chaîne hexadécimale). Il crée une paire clé-valeur

dans le dictionnaire, où la clé (num) est le nombre lui-même, et la valeur est le tuple contenant les représentations binaire et hexadécimale. print(nombre_binaire_hex) : Cette ligne imprime le dictionnaire number_binary_hex sur la console.

Question 50

Mettre en correspondance les mots et leur inverse dans une phrase

Exemple de sortie

Bonjour, comment allez-vous ?

{'Hello,' : ',olleH', 'how' : 'woh', 'are' : 'era', 'you ?' : ' ?uoy'}

Code python :

```python
sentence = "Hello, how are you?"
word_reverse_mapping = {word: word[::-1] for word in sentence.split()}
print(sentence)
print(word_reverse_mapping)
```

q709.py

Le code traite une phrase et crée un dictionnaire nommé word_reverse_mapping à l'aide d'une compréhension de dictionnaire en Python. Ce dictionnaire associe les mots de la phrase à leurs versions inversées respectives. Voici une explication pas à pas du code :

sentence = "Hello, how are you ?" : Cette ligne définit une variable de chaîne appelée sentence contenant la phrase "Hello, how are you ?" word_reverse_mapping = {word : word[: :-1] for word in sentence.split()} : Cette ligne crée le dictionnaire word_reverse_mapping à l'aide d'une compréhension de dictionnaire. Voici comment cela fonctionne : word : word[: :-1] for word in sentence.split() est la compréhension du dictionnaire. Elle effectue les étapes suivantes : sentence.split() divise la phrase en une liste de mots. Dans ce cas, elle inclura des mots comme "Hello," (avec une virgule) et "you ?" (avec un point d'interrogation). (avec un point d'interrogation). for word in sentence.split() parcourt chaque mot de la liste de mots. mot[: :-1] inverse l'ordre des caractères dans le mot. Pour ce faire, on utilise la notation de découpage de Python, où [: :-1] est utilisé pour inverser une chaîne de caractères. Cela crée une paire clé-valeur dans le dictionnaire, où la clé (word) est le mot lui-même, et la valeur (word[: :-1]) est le mot inversé. print(phrase) : Cette ligne imprime la phrase originale, qui est "Hello, how are you ?", sur la console. print(word_reverse_mapping) : Cette ligne affiche le dictionnaire word_reverse_mapping sur la console.