Listes du jour 5

Listes

Il existe quatre types de données de collecte dans Python

- Liste: est une collection commandée et modifiable (modifiable). Permet aux membres e n double. • Tuple: est une collection qui est commandée et immuable ou non modifiable (immuable). Permet aux membres en double. • set: est une collection qui n'est pas ordon née, non indexée et non modifiable, mais nous pouvons ajouter de nouveaux éléments à l' ensemble. Les membres en double ne sont pas autorisés. • Dictionnaire: est une collectio n qui n'est pas ordonnée, modifiable (modifiable) et indexée. Pas de membres en double.

Une liste est la collecte de différents types de données qui sont commandés et modifiables (m utables). Une liste peut être vide ou peut avoir différents éléments de type de données.

Comment créer une liste

Dans Python, nous pouvons créer des listes de deux manières:

- Utilisation de la fonction intégrée de la liste

```
# syntaxe lst = list () vide_list = list () # Ceci est une liste vide, aucun élément dans la liste print (len (vide_list)) # 0
```

- Utilisation des crochets, []

```
# syntaxe lst = [] vide_list = [] # Ceci est une liste vide, aucun élément dans la liste imprime r (Len (vide_list)) # 0
```

Listes avec des valeurs initiales. Nous utilisons *len()* pour trouver la longueur d'une liste.

```
fruits = ['banana', 'orange', 'mango', 'lemon']
# list of fruits
```

```
légumes = ['tomate', 'pomme de terre', 'Cabbage', 'Onion', 'Carrot']
# Liste des légumes Animal_Products = ['Milk', 'Meat', 'Butter', 'Yoghurt']

# Liste des produits animaux web_techs = ['html', 'css', 'js', 'react', 'redux', 'node', 'mongd
b'] # Liste des technologies de technologies Web = ['Finlande', ', estonie', 'Denmark', 'Sw
eden', 'Norway']



# Print the lists and its length  print('Fruits:', fruits)  print('Number of fruits:', len(fruits))
print('Vegetables:', vegetables)  print('Number of vegetables:', len(vegetables))  print('A
nimal products:',animal_products)  print('Number of animal products:', len(animal_prod
ucts))  print('Web technologies:', web_techs) imprimer ('Nombre de technologies Web:',
Len (web_techs)) imprimer ('pays:', pays) imprimer ('Nombre de pays:', Len (pays))
```

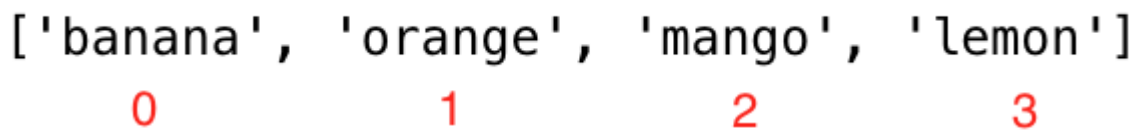
```
Fruits de sortie: ['banane', 'orange', 'mango', 'citron'] nombre de fruits: 4 légumes: ['tomate', 'po
mmes de terre', 'Cabbage', 'oignon', 'carrot'] nombre de légumes: 5 produits animaux: [«lait», ``
viande ". «JS», «react», «redux», «nœud», «Mongdb»] Nombre de technologies Web: 7 pays: [
«Finlande», «Estonie», «Danemark», «Suède», «Norvège»] Nombre de pays: 5
```

- Les listes peuvent avoir des éléments de types de données différents

```
lst = ['asabeneh', 250, true, {'country': 'Finlande', 'City': 'Helsinki'}] # Liste contenant diff
érents types de données
```

Accessing List Items Using Positive Indexing

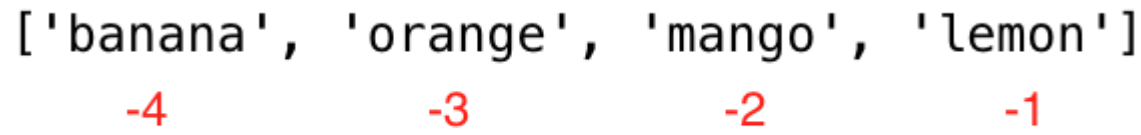
Nous accédons à chaque élément dans une liste en utilisant leur index. Un index de la liste co
mmence à partir de 0. L'image ci-dessous montre clairement où l'index commence




```
Fruits = ['banana', 'orange', 'mango', 'citron'] premier_fruit = fruits [0] # Nous accédons
au premier élément en utilisant son index print (first_fruit) # banana second_fruit = frui
ts [1] imprimer (second_fruit) # Orange Lemon # dernier index last_index = len (fruits)
- 1 last_fruit = fruits [last_index]
```

Accéder aux éléments de la liste à l'aide d'indexation négative

L'indexation négative signifie qu'à partir de la fin, -1 fait référence au dernier élément, -2 fait r
éférence au deuxième dernier élément.




```
fruits = ['banana', 'orange', 'mango', 'citron'] premier_fruit = fruits [-4] la
st_fruit = fruits [-1] second_last = fruits [-2] print (premier_fruit) # bana
na print (last_fruit) # lemon print (second_last) # mango)
```

Déballage des éléments de la liste

```
lst = ['item1', 'item2', 'item3', 'item4', 'item5'] first_item, second_item, third
_item, * rest = lst print (first_item) # item1 print (second_item) # item2 trin
t (tiers_item) # item3 print (rest) # ['item4' ',' item5 ']
```

```
# Premier exemple
Fruits = ['banane', 'orange', 'mango', 'citron', 'lime', 'Apple'] first_fruit, second_fruit, troisième_f
ruit, * rest = fruits print (first_fruit) # banana print (second_fruit) # orange print (troisième_frui
t) # mango imprimer (repos) # [" lime ',' lime ", exemple '] exemple # mango # [repos) # [' 'Con
tan' '. À propos de la liste des premiers, deuxième, troisième, * repos, dixième = [1,2,3,4,5,6,7,
8,9,10] imprimer (premier) # 1 imprimer (deuxième) # 2 imprimer (troisième) # 3 print (rest) #
[4,5,6,7,8,9] «France», «Belgique», «Suède», «Danemark», «Finlande», «Norvège», «Icel et»,
«Estonia»] Gr, Fr, Bg, SW, * Scanddic, es = Pountre Print (Gr) print (fr) print (bg) print (SW)
print (scandic) print (es)
```

Trancher les éléments d'une liste

- Indexation positive: nous pouvons spécifier une plage d'index positifs en spécifiant le d
  ébut, la fin et l'étape, la valeur de retour sera une nouvelle liste. (valeurs par défaut pour
  start = 0, end = len (lst) - 1 (dernier élément), étape = 1)

```
fruits = ['banana', 'orange', 'mango', 'citron'] all_fruits = fruits [0: 4] # Il renvoie tou
s les fruits # Cela donnera également le même résultat que celui ci-dessus

all_fruits = fruits [0:] # Si nous ne définissons pas où s'arrêter, cela prend tout le reste

Orange_and_mango = fruits [1: 3] # Il n'inclut pas le premier index

Orange_mango_lemon = fruits [1:]
```

```
Orange_and_lemon = fruits [:: 2] # Ici, nous avons utilisé un 3ème argument, étape. Cela pre
ndra chaque 2Cnd Item - [«banane», «mangue»]
```

- Indexation négative: nous pouvons spécifier une plage d'index négatifs en spécifiant le d
  ébut, la fin et l'étape, la valeur de retour sera une nouvelle liste.

```
fruits = ['banana', 'orange', 'mango', 'citron'] all_fruits = fruits [-4:] # il renvoie tous les fruits
orange_and_mango = fruits [-3: -1] # il n'inclut pas le dernier index, ['orange', 'mango'] orang
e_mango_ Fruits [-3:] # Cela donnera à partir de -3 à la fin, ['orange', 'mango', 'citron'] reverse
_fruits = fruits [:: - 1] # Une étape négative prendra la liste dans l'ordre inverse, ["citron ',' ma
ngo ',' orange ',' banana '], [citron', 'mango', 'orange', 'banana']
```

Modification des listes

La liste est une collection ordonnée mutable ou modifiable d'articles. Permet de modifier la liste des fruits.

```
Fruits = ['banana', 'orange', 'mango', 'citron'] fruits [0] = 'avocado' imprimer (fruits) # ['avocado'
, 'orange', 'mango', ' 'citron'] last_index = len (fruits) - 1 fruits [last_index] = 'lime' print (fruits)
# ['avocado', 'Apple', 'mango', 'lime']
```

Vérification des éléments dans une liste

Vérification d'un élément s'il est membre d'une liste utilisant l'opérateur *in*. Voir l'exemple ci-dessous.

```
Fruits = ['banana', 'orange', 'mango', 'citron'] DOID_EXIST = 'banana' d
ans les fruits imprimer (DOED_EXIST) # true dacks_exist = 'lime' dans
les fruits imprimés (DOEXIST) # false # false
```

Ajout d'éléments à une liste

Pour ajouter un élément à la fin d'une liste existante, nous utilisons la méthode *append()*.

```
# Syntaxe lst = list () lst.append (item) fruits = ['banana', 'orange', 'mango', 'citron'] fr
uits.append ('Apple') imprimer (fruits ') # [banana', 'orange', 'mango', 'Lemon', 'Appl
e'] fruits.Apprend (lime ') # [Lemon', 'Apple'] fruits.Apprend (lime ') # [Lemon' «ora
nge», «mangue», «citron», «pomme», «chaux»]
```

```
print(fruits)
```

Insertion des articles dans une liste

Nous pouvons utiliser la méthode *insert()* pour insérer un seul élément à un index spécifié da ns une liste. Notez que d'autres éléments sont déplacés vers la droite. Les méthodes *insert()* p rennent deux arguments: index et un élément à insérer.

```
# syntaxe lst = ['item1', 'item2'] lst.ins
ert (index, item)
```

```
Fruits = ['banane', 'orange', 'mango', 'citron'] fruits.insert (2, 'Apple') # insérer Apple entre
orange et mango (fruits) # [`` banane '', `` orange '', `` pomme '', 'mango', 'citron'] fruits.ins
ert (3, 'lime') # [" ',', '] fruits. «chaux», «mangue», «citron»] imprimer (fruits)
```

Supprimer les éléments d'une liste

La méthode supprimée supprime un élément spécifié d'une liste

```
# syntaxe lst = ['item1', 'item2'] lst.re
move (item)
```

```
Fruits = ['banana', 'orange', 'mango', 'citron', 'banana'] fruits.remove ('banana') imprimer (fruits)
# ['orange', 'mango', 'citron', 'banana'] - cette méthode supprime le premier événement de la list
e des fruits.remove ('limon') # lister les fruits.remove ('limon') # # lister fruits.remove ('limon')
# # lister Fruits.Remove (" Lemon ') # Fruit) # lister Fruits.Remove. [«orange», «mangue», «ba
nane»]
```

Suppression d'articles à l'aide de POP

La méthode *pop()* supprime l'index spécifié, (ou le dernier élément si l'index n'est pas spécifié):

```
# syntaxe lst = ['item1', 'item2'] lst.pop ()
# dernier élément lst.pop (index)
```

```
fruits = ['banana', 'orange', 'mango', 'citron'] fruits.pop () print (fruits) # ['banan
a', 'orange', 'mango'] fruits.pop (0) print (fruits) # ['orange', 'mango']
```

Suppression des articles à l'aide de Del

Le mot-clé *del* supprime l'index spécifié et il peut également être utilisé pour supprimer les élé ments dans la plage d'index. Il peut également supprimer complètement la liste

```
# syntaxe lst = ['item1', 'item2'] del lst [index] # un seul élément del lst
# pour supprimer complètement la liste
```

```
Fruits = ['banana', 'orange', 'mango', 'citron', 'kiwi', 'lime'] del fruits [0] print (fruits) # ['or
ange', 'mango', 'citron', 'kiwi', 'lime'] del fruits [1] imprimer (fruits) # ['' orange ',' lemon ','
kiwi 'Lime'] Del Fruits [1: 3] # Ceci supprime les éléments entre les index donnés, il ne s
upprime donc pas l'élément avec l'index 3!  imprimer (fruits) # ['orange', 'lime'] del fruits
imprimer (fruits) # Ceci devrait donner: nameError: nom 'fruits' n'est pas défini
```

Éléments de liste de compensation

La méthode *clear()* vide la liste:

```
# syntaxe lst = ['item1', 'item2'] lst.cle
ar ()


fruits = ['banane', 'orange', 'mango', 'citron'] fruits.clear () print (fruits) #
[]
```

Copie d'une liste

Il est possible de copier une liste en le réaffectant à une nouvelle variable de la manière suiva nte: list2 = list1. Maintenant, List2 est une référence de List1, toutes les modifications que no us apportons dans List2 modifieront également l'original, List1. Mais il y a beaucoup de cas dans lesquels nous n'aimons pas modifier

L'original, nous aimons plutôt avoir une copie différente. L'un des moyens d'éviter le pr oblème ci-dessus est d'utiliser *copy()*.

```
# syntaxe lst = ['item1', 'item2'] lst_co
py = lst.copy ()
```

```
fruits = ['banana', 'orange', 'mango', 'citron'] fruits_copy = fruits.copy () print (fruits_
copy) # ['banana', 'orange', 'mango',

'citron']
```

JOINS LISTES

Il existe plusieurs façons de rejoindre ou de concaténer deux listes ou plus dans Python.

- Plus opérateur (+)

```
# syntaxe list3 = list1 + list2
```

```
Positive_numbers = [1, 2, 3, 4, 5] zéro = [0] négatif_numbers = [-5, -4, -3, -2, -1] entières = n
égatif_numbers + zéro + positif_numbers -1, 0, 1, 2, 3, 4, 5] Fruits = ['banana', 'orange', 'man
go', 'citron'] légumes = ['tomato', 'pomme print (fruits_and_vegetables) # ['banane', 'orange', '
mango', 'citron', 'tomate', 'pommes de terre', 'chou', 'oignon', 'carrot']
```

- JOINS Utilisation de la méthode Extend () La méthode *extend()* permet d'ajouter la liste dans une liste. Voir l'exemple ci-dessous.

```
# syntaxe
list1 = ['item1', 'item2'] list2 = ['item3', 'item4', 'item
5']
list1.extend (list2)
```

```
num1 = [0, 1, 2, 3] num2 = [4, 5, 6] num1.extend (num2) print ('nombres:', num1) # n
ombres: [0, 1, 2, 3, 4, 5, 6] négatif_numbers = [-5, -4, -3, -2, -1] positif_numbers = 3,4
,5]
```

```
zéro = [0]

négatif_numbers.extend (zéro) négatif_numbers.extend (positif_numbers) print ('entiers:', né
gatif_numbers) # entières: [-5, -4, -3,

-2, -1, 0, 1, 2, 3, 4, 5] Fruits = ['banane', 'orange', 'mango', 'citron'] légumes = ['tomato', 'pom
me de terre', `` Cabbage ', `` Onion ',' carot '] fruits. [«banane», «orange», «mangue», «citro
n», «tomate», «pomme de terre», «chou», «oignon», «carotte»]
```

Compter les éléments dans une liste

La méthode *count()* renvoie le nombre de fois qu'un élément apparaît dans une liste:

```
# Syntaxe lst = ['item1', 'item2'] lst.co
wing (item)
```

```
fruits = ['banana', 'orange', 'mango', 'citron'] print (fruits.count ('orange'))
# 1 âges = [22, 19, 24, 25, 26, 24, 25, 24] print (Ages.Count (24)) # 3
```

Trouver l'indice d'un élément

La méthode *index()* renvoie l'index d'un élément dans la liste:

```
# syntaxe lst = ['item1', 'item2'] lst.in
dex (item)
```

```
Fruits = ['banana', 'orange', 'mango', 'citron'] print (fruits.index ('orange')) # 1 Ages = [2
2, 19, 24, 25, 26, 24, 25, 24] print (Ages.index (24)) # 2, la première occurrence
```

Inverser une liste

La méthode *reverse()* inverse l'ordre d'une liste.

```
# syntaxe lst = ['item1', 'item2'] lst.re
verse ()
```

```
Fruits = ['banane', 'orange', 'mango', 'citron'] fruits.reverse () print (fruits) # ['citron'
, 'mango', 'orange', 'banana']
```

```
Âges = [22, 19, 24, 25, 26, 24, 25, 24] Ages.reverse () imprimer (âges)
# [24, 25, 24, 26, 25, 24, 19, 22]
```

Tri des éléments de liste

Pour trier les listes, nous pouvons utiliser la méthode *sort()* ou les fonctions intégrées *sorted()* . La méthode *sort()* réorque les éléments de la liste dans l'ordre croissant et modifie la liste ori ginale. Si un argument de la méthode *sort()* inverse est égal à True, il organisera la liste dans l' ordre descendant.

- TROT (): Cette méthode modifie la liste originale

```
# syntaxe
lst = ['item1', 'item2'] lst.sort () # ascendant lst.sort (revers
= true) # descendant
```

Exemple:

```
Fruits = ['banane', 'orange', 'mango', 'citron'] fruits.sort () print (fruits) # triés dans l'ordr
e alphabétique, ['banana', 'citron', 'mango', 'orange'] fruits.sort ',' mango ',' limon ', lon 'b
anane'] Âges = [22, 19, 24, 25, 26, 24, 25, 24] Âges.Sort () imprimer (âges) # [19, 22, 2
4, 24, 24, 25, 25, 26]




Âges.Sort (revers = true) imprimer (âges) # [26, 25, 25, 24, 24, 24, 22, 1
9]
```

trid (): renvoie la liste commandée sans modifier l'exemple de liste d'origine:

```
Fruits = ['banana', 'orange', 'mango', 'citron'] imprimer (trié (fruits)) # ['banana', 'citron', '
mango', 'orange'] # fruits de commande inverse = ["banana ',' orange ',' mango ',' citron ']
fruits = Trised (fruits, revers = true) print (fruits) # ['Orange', 'mango', 'citron', 'banana']
```

Vous êtes diligent et vous avez déjà réalisé beaucoup. Vous venez de terminer les défis du j our 5 et vous êtes à 5 pas de la grandeur vers la grandeur. Faites maintenant quelques exercice s pour votre cerveau et vos muscles.

Exercices: Jour 5

Exercices: niveau 1

1. Déclarer une liste vide

2. Déclarer une liste avec plus de 5 articles

3. Trouvez la durée de votre liste 4. Obtenez le premier élément, l'élément moyen et l e dernier élément de la liste 5. Déclarez une liste appelée mixtes_data_types, mettez votre (nom, âge, hauteur, marit

al

statut, adresse) 6. Déclarez une variable de liste nommée IT_COMPANIES et attribuez les valeurs initiales Facebook, Google, Microsoft, Apple, IBM, Oracle et Amazon. 7. Imprimez la liste à l'aide de *print()* 8. Imprimez le nombre d'entrepris es dans la liste 9. Imprimez la première, moyenne et dernière société 10. Imprimez la liste après avoir modifié l'une des sociétés 11. Ajoutez une société informatique à IT_COMPANIES 12. Insérez une société informatique au milieu de la liste des s ociétés 13. Changer l'un des noms IT_COMPANIES à Up It (IBM exclu!) 14. «15 . Vérifiez si une certaine entreprise existe dans la liste IT_COMPANIES. 16. Trie r la liste à l'aide de Sort () Méthode 17. Inversez la liste dans l'ordre descendant en utilisant la méthode Reverse () 18. Éliminez les 3 premières sociétés de la liste 19. Éliminez les 3 dernières sociétés de la liste 20. Tranchez la société informatique in termédiaire ou les sociétés de la liste 21. Supprimer la dernière société informatiqu e de la liste des sociétés de la liste des sociétés. Listes:

```
front_end = ['html', 'css', 'js', 'react', 'redux'] back_end = ['node', 'express', 'mon
godb']
```

27. Après avoir rejoint les listes de la question 26. Copiez la liste jointe et affectez-le à une variable full_stack, puis insérez Python et SQL après Redux.

Exercices: niveau 2

1. Ce qui suit est une liste de 10 étudiants à l'âge:

```
Âges = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
```

•Trier la liste et retrouver l'âge min et max • Ajoutez à nouveau l'âge min et l'âge maximu m à la liste • Trouvez l'âge médian (un élément moyen ou deux éléments moyens divisés p ar deux) • Trouver la plage de l'âge moyen (somme de tous les éléments divisés par leur nu méro) • Moyenne), utilisez *abs()* Méthode 1. Trouvez le (IES du milieu dans la liste des p ays 2. Divisez la liste des pays en deux listes égales si elle est même si ce n'est pas un pays de plus pour la première moitié. 3. [«Chine», «Russie», «États-Unis», «Finlande», «Suède », «Norvège», «Danemark»]. Débordez les trois premiers pays et les autres en tant que pay s scandiques.

Félicitations!