Boucles du jour-10

Boucles

La vie est pleine de routines. En programmation, nous effectuons également beaucoup de tâc hes répétitives. Afin de gérer les langages de programmation de tâches répétitifs, utilisez des boucles. Le langage de programmation Python fournit également les types de deux boucles s uivants:

1. Pendant la boucle
2. Pour Loop

Pendant la boucle

Nous utilisons le mot réservé *while* pour faire une boucle de temps. Il est utilisé pour exécut er un bloc d'instructions à plusieurs reprises jusqu'à ce qu'une condition donnée soit satisfait e. Lorsque la condition devient fausse, les lignes de code après la fin de la boucle se poursui vront.

```
# Syntaxe pendant la conditi
on: le code va ici
```

Exemple:

```
count = 0 tandis que le nombre <
5: print (count) count = count +
1 #prints de 0 à 4
```

Dans ce qui précède, la boucle, la condition devient fausse lorsque le nombre est de 5. C'est à ce moment que la boucle s'arrête. Si nous sommes intéressés à exécuter le bloc de code une fois que la condition n'est plus vraie, nous pouvons utiliser *else*.

```
  # syntax
while condition:
    code goes here
else:
    code goes here
```

Exemple:

```
Count = 0 tandis que le nombre
< 5: print (count) count = count
+ 1 else: print (count)
```

La condition de boucle ci-dessus sera fausse lorsque le nombre est 5 et la boucle s'arrêt e, et l'exécution démarre l'instruction ELSE. En conséquence, 5 sera imprimé.

Break and Contin - Partie 1

- Break: Nous utilisons la pause lorsque nous aimons sortir de la boucle ou arrêter la boucle.

```
# Syntaxe pendant la condition: le co
de va ici si un autre_condition:



casser
```

Exemple:

```
Count = 0 tandis que le nombre
< 5: print (count) compter = cou
nt + 1 if count == 3: pause
```

La boucle ci-dessus est uniquement imprimée 0, 1, 2, mais quand elle atteint 3, elle s'arrête.

- Continuez: avec l'instruction Continuer, nous pouvons ignorer l'itération actuelle et cont inuer avec la suivante:

```
# Syntaxe pendant la condition: le co
de va ici si un autre_condition:



continuer
```

Exemple:

```
Count = 0 tandis que le nombre < 5: If
Count == 3: Count = Count + 1 Contin
uer Print (Count) Count = Count + 1
```

La boucle ci-dessus imprime uniquement 0, 1, 2 et 4 (sauter 3).

Pour boucle

Un mot-clé *for* est utilisé pour faire une boucle pour une boucle, similaire avec d'autres langag es de programmation, mais avec certaines différences de syntaxe. La boucle est utilisée pour it ération sur une séquence (c'est soit une liste, un tuple, un dictionnaire, un ensemble ou une cha îne).

- Pour boucle avec liste

```
# Syntaxe pour itérateur en L
ST:
Le code va ici
```

Exemple:

```
Numéros = [0, 1, 2, 3, 4, 5] Pour le nombre en numéros: # Le numéro est un nom temporaire pour se référer aux éléments de la liste, valide uniquement à l'intérieur de cette boucle imprim er (numéro) # Les numéros seront imprimés en ligne par ligne, de 0 à 5
```

- Pour boucle avec chaîne

```
# syntaxe
Pour Iterator dans String: le code va
ici
```

Exemple:

```
Langue = 'Python' pour la lettre en l
angue: imprimer (lettre)



pour I à portée (Len (langue)):
imprimer (langue [i])
```

- Pour boucle avec tuple

syntaxe

```
for iterator in tpl:
    code goes here
```

Exemple:

```
Nombres = (0, 1, 2, 3, 4, 5) pour le nombre
en numéros: imprimer (numéro)
```

- Pour une boucle avec un dictionnaire en boucle à travers un dictionnaire vous donne la cl é du dictionnaire.

```
# Syntaxe pour Iterator dans D
CT: le code va ici
```

Exemple:

```
personne = {'first_name': 'asabeneh',

'last_name': 'encoreayeh', 'Âge': 250,
```

'Country': 'Finlande', 'is_marred': true, 'compétences': ['javascript', 'react', 'node', 'mongodb', ' python'], 'adresse': {street ':' Street ',' zipcode ':' 02210 '}} pour la clé: la clé: la clé) pour la c lé, la valeur, la valeur en personne. Nous obtenons les touches et les valeurs imprimées • bo ucles dans Set # Syntaxe

Pour Iterator dans ST: Code v
a ici

Exemple:

IT_COMPANIES = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
pour la société dans IT_COMPANIES: print (Company)

Break and Contin - Partie 2

Rappel court: *Break*: Nous utilisons Break lorsque nous aimons arrêter notre boucle av ant sa fin.

Syntaxe pour itérateur en séquence:

Le code va ici si la condition
: pause

Exemple:

Nombres = (0,1,2,3,4,5) pour le no
mbre en nombres: imprimer (numér
o) si numéro == 3: pause

Dans l'exemple ci-dessus, la boucle s'arrête lorsqu'elle atteint 3.

Continuez: nous utilisons Continuer lorsque nous aimons ignorer certaines des étapes de l'itér ation de la boucle.

```
# Syntaxe pour itérateur en séquence: l
e code va ici si condition: Continue
```

Exemple:

```
Numéros = (0,1,2,3,4,5) pour le nombre en nombres: imprimer (numéro) si numéro == 3: Cont inuez Print ('Le numéro suivant devrait être', le numéro + 1) si le numéro! = 5 Else Print ("Loo p's End") # pour les conditions courtes de la main nécessite les deux si et bien les statuts impri ment ('Extérieur de la boucle'))
```

Dans l'exemple ci-dessus, si le nombre est égal à 3, l'étape *after* La condition (mais à l'intérieur de la boucle) est ignorée et l'exécution de la boucle continue s'il reste des itérations.

La fonction de plage

La fonction *range()* est utilisée sur la liste des nombres. Le *range(start, end, step)* prend tr ois paramètres: Démarrage, fin et incrément. Par défaut, il part à partir de 0 et l'incrément est 1. La séquence de plage nécessite au moins 1 argument (fin). Création de séquences à l'aide d e la plage

```
LIST = (plage (11)) imprimer (LST) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] ST = set (Range (1, 11)) # 2 Arguments indiquent le démarrage et la fin de la séquence, pas sur défaut 1 print (ST) # {1, 3 , 4, 5, 6, 7, 8, 9, 10} LST = list (plage (0,11,2)) print (lst) # [0, 2, 4, 6, 8, 10] st = set (plage (0,1 1,2)) print (st) # {0, 2, 4, 6, 8, 10} # syntaxe pour iterator in plage (start, end, step):
```

Exemple:

```
pour le nombre dans la gamme (11): imprimer (numéro) # imprime 0 à 10, sans co
mpter 11
```

Imbriqué pour la boucle

Nous pouvons écrire des boucles dans une boucle.

```
# Syntaxe pour x en y: p
our t dans x: imprimer (
t)
```

Exemple:

```
personne = {'premier_name': 'asabeneh', 'last_name': 'yetayeh', 'age': 250, 'country': 'Fi
nland', 'is_marred': true, 'compétences': 'javascript' 'Zipcode': '02210'}} pour la clé en
personne: si clé == 'compétences': pour la compétence en personne ['compétences']: i
mprimer (compétence)
```

Pour ailleurs

Si nous voulons exécuter un message à la fin de la boucle, nous utilisons d'autre.

```
# Syntaxe pour iterator dans la gamme (démarrage, fin, étape)
: faire autre chose: imprimer («la boucle terminée»)
```

Exemple:

```
pour le nombre dans la gamme (11): imprimer (numéro) # imprime 0 à 10, sans c
ompter 11
else: imprimer («la boucle s'arrête à», numéro)
```

Passer

Dans Python, lorsque l'instruction est requise (après le demi-colon), mais nous n'aimons pas exécuter de code là-bas, nous pouvons écrire le mot `pass` pour éviter les erreurs. Nous pouv ons également l'utiliser comme espace réservé, pour les futures déclarations.

Exemple:

```
pour le nombre dans la plage (6): pa
sser
```

Vous avez établi une grande étape, vous êtes imparable. Continue! Vous venez de terminer l es défis du jour 10 et vous êtes à 10 pas de la grandeur à la grandeur. Faites maintenant quelqu es exercices pour votre cerveau et vos muscles.

Exercices: Jour 10

Exercices: niveau 1

1. Itérer 0 à 10 en utilisant pour la boucle, faites de même en utilisant la boucle.

2. Iléter 10 à 0 en utilisant pour la boucle, faites de même en utilisant la boucle pendant.

3. Écrivez une boucle qui fait sept appels à imprimer (), nous obtenons donc sur la sor tie le triangle suivant:

```
#
##
###
####
#####
######
#######
```

4. Utilisez des boucles imbriquées pour créer ce qui suit:

```
# # # # # # # # # # # # #
# # # # # # # # # # # # #
# # # # # # # # # # # # #
# # # # # # # # # # # # #
# # # # # # # # # # # # #
#
```

5. Imprimez le modèle suivant:

```
0 x 0 = 0 1 x 1 = 1 2
x 2 = 4 3 x 3 = 9 4 x
4 = 16 5 x 5 = 25 6 x
6 = 36 7 x 7 = 49 8 x
8 = 64 9 x 9 = 81 10
x 10 =} 64 9 X 9 = 8
1 10 x 10 =
```

6. itérer dans la liste, [«python», «numpy», «pandas», «django», «flask»] en utilisant une b oucle pour une boucle et imprimez les éléments. 7. Utiliser pour la boucle pour itérer de 0 à 100 et imprimer uniquement les nombres 8. Utiliser pour la boucle pour itérer de 0 à 100 et imprimer uniquement les nombres impairs

Exercices: niveau 2

1. Utiliser pour la boucle pour itérer de 0 à 100 et imprimer la somme de tous les nombres.

La somme de tous les nombres est 5050.

1. Utilisez pour la boucle pour itérer de 0 à 100 et imprimez la somme de tous les even s et la somme de toutes les chances.

La somme de tous les evens est de 2550. Et la somme de toutes les chances est de 2500.

Exercices: niveau 3

1. Accédez au dossier de données et utilisez le fichier Pays.py. Boucle à travers les pays et extraire tous les pays contenant le mot *land*. 2. Il s'agit d'une liste de fruits, [«banane», «or ange», «mangue», «citron»] inversent l'ordre à l'aide de la boucle. 3. Accédez au dossier d e données et utilisez le fichier Pays_data.py.

---

je. Quel est le nombre total de langues dans les données II. T rouvez les dix langues les plus parlées des données III. Trouve z les 10 pays les plus peuplés du monde

Félicitations!