Jour 14 Fonctions d'ordre supérieur

Fonctions d'ordre supérieur

Dans Python, les fonctions sont traitées comme des citoyens de première classe, vous per mettant d'effectuer les opérations suivantes sur les fonctions:

- Une fonction peut prendre une ou plusieurs fonctions comme para mètres • Une fonction peut être renvoyée à la suite d'une autre fonct ion • Une fonction peut être modifiée

- Une fonction peut être attribuée à une variable

Dans cette section, nous couvrirons:

1.

Fonctionner comme paramètre

DEF SUM_NUMBERS (NUMS): # Fonction normale Somme de retour (NUMS) # Une fonc tion triste abusant de la fonction de somme intégrée: <

def supérieur_order_function (f, LST): # Fonctionne comme une somme de paramètres = f (L ST) Retour Retour Sommation

Résultat = supérieur_order_function (SUM_NUMBERS, [1, 2, 3, 4, 5]) Imprimer (Résultat) # 15

Fonctionne comme une valeur de retour

```
def square(x):          # a square function
    return x ** 2

def cube(x):            # a cube function
    return x ** 3

def absolute(x):        # an absolute value function
    if x >= 0:
        return x
    else:
        return -(x)
```

```
def supérieur_order_function (type): # Une fonction d'ordre supérieur renvoyant une fonc
tion
    Si type == 'carré':
    carré de retour
    elif type == 'cube':
    cube de retour
    elif type == 'absolue':
    Retour Absolute

Résultat = supérieur_order_function ('carré') print (result (3)) #
9 Result = supérieur_order_function ('cube') print (result (3)) #
27 result = supérieur_order_function ('absolu'))



imprimer (résultat (-3)) # 3
```

Vous pouvez voir à partir de l'exemple ci-dessus que la fonction d'ordre supérieur renvo
ie différentes fonctions en fonction du paramètre passé

Fermetures de python

Python permet à une fonction imbriquée d'accéder à la portée extérieure de la fonction d'enclo
sage. Ceci est connu comme une fermeture. Voyons comment les fermetures fonctionnent à P
ython. Dans Python, la fermeture est créée en nichant une fonction dans une autre fonction en
capsulante, puis en renvoyant la fonction intérieure. Voir l'exemple ci-dessous.

Exemple:

```
def add_ten (): dix = 10 def add (num
): return num + dix return add




Close_Result = add_ten () print (closure_result (
5)) # 15 imprimer (clôture_result (10)) # 20
```

Décorateurs de python

Un décorateur est un motif de conception dans Python qui permet à un utilisateur d'ajouter de
nouvelles fonctionnalités à un objet existant sans modifier sa structure. Les décorateurs sont
généralement appelés avant la définition d'une fonction que vous souhaitez décorer.

Créer des décorateurs

Pour créer une fonction de décorateur, nous avons besoin d'une fonction extérieure avec une fonction de wrapper intérieure.

Exemple:

```
# Fonction normale Def Greeting (): Retour 'Bienvenue s
ur Python' def uppercase_decorator (fonction): def wrapp
er (): func = function () make_uppercase = func.upper ()
return Make_uppercase return wrapper wrapper wrapper




g = uppercase_decorator (salutation)                    print (g ())
# Bienvenue à Python

## Implémentons l'exemple ci-dessus avec un décorateur

" 'Cette fonction de décorateur est une fonction d'ordre supérieur qui prend une fo
nction en tant que paramètre' " '
Def upperase_decorator (fonction): def wrapper (): func = fu
nction () make_uppercase = func.upper () return make_uppe
rase return wrapper @upperase_decorator def grelting (): ret
urn 'bienvenue à python' print (salut ()) # bienvenue à pytho
n ()
```

Appliquer plusieurs décorateurs à une seule fonction

```
" 'Ces fonctions de décorateur sont des fonctions d'ordre supérieur qui prennent des f
onctions comme des paramètres' " # First décorateur def uppercase_decorator (foncti
on): def wrapper (): func = function () make_uppercase = func.upper () return make_
uppercase return wrapper # deuxième décorateur def split_string_decorator (fonction
): func wrapper (func = fonction ()
```

```
        splitted_string = func.split()
        return splitted_string

    return wrapper

@split_string_decorator
@uppercase_decorator     # order with decorators is important
in this case - .upper() function does not work with lists
def greeting():
    return 'Welcome to Python'
print(greeting())   # WELCOME TO PYTHON
```

Accepter les paramètres dans les fonctions de décorateur

La plupart du temps, nous avons besoin de nos fonctions pour prendre des paramètres, no us pourrions donc devoir définir un décorateur qui accepte les paramètres.

```
Def Decorator_With_Parameters (fonction): def wrapper_acceping_parameters (para1, p
ara2, para3): function (para1, para2, para3) print ("je vis dans {}".




@decorator_with_parameters def print_full_name (first_name, last_name, count
ry): print ("je suis {} {}. J'adore enseigner.". Format (first_name, last_name, co
untry)))




print_full_name ("asabeneh", "encoreayeh", 'Finlande')
```

Fonctions d'ordre supérieur intégrés

Certaines des fonctions d'ordre supérieur intégré que nous couvrons dans cette partie sont *map()*, *filter* et *reduce*. La fonction lambda peut être adoptée en tant que paramètre et le meil leur cas d'utilisation des fonctions lambda est dans des fonctions telles que la carte, le filtre et la réduction.

Python - fonction de carte

La fonction map () est une fonction intégrée qui prend une fonction et itérable comm e paramètres.

```
# Carte de syntaxe (fonction, itérable)
```

Exemple: 1

```
Nombres = [1, 2, 3, 4, 5] # iTable Def Square (x):
```

```
retour x ** 2
nombres_squared = map (carré, nombres) print (list (nombres_squared)) # [1
, 4, 9, 16, 25]
# Permet de l'appliquer avec une carte Lambda Numbers_Squared = (lambda
x: x ** 2, nombres) imprimer (list (nombres_squared)) # [1, 4, 9, 16, 25]
```

Exemple: 2

```
nombres_str = ['1', '2', '3', '4', '5'] # iTable nombres_int = map (int, nombres_st
r) print (list (nombres_int)) # [1, 2, 3, 4, 5]
```

Exemple: 3

```
Noms = ['Asabeneh', 'lidiya', 'ermias', 'Abraham'] # iTable def change_to_upper (nom): r
etour name.upper () names_upper_Cased = map (change_to_upper, noms) imprimer (list
(names_upper_cased)) # ['asabeneh', 'lidIya', ', ''. 'Abraham'] # Letons-le avec une fonctio
n de fonction lambda names_upper_lased = map (name lambda: name.upper (), noms) pr
int (list (names_upper_cled)) # ['asabeneh', 'lidiya', 'ermias', 'abraham']
```

Ce que la carte fait réellement est d'itréger une liste. Par exemple, il modifie les noms en supér ieur et renvoie une nouvelle liste.

Python - Fonction de filtre

La fonction Filter () appelle la fonction spécifiée qui renvoie booléen pour chaque élément de l' Itable spécifié (liste). Il filtre les éléments qui satisfont aux critères de filtrage.

```
# Filtre de syntaxe (fonction, itérable)
```

Exemple: 1

```
# Permet de filtrer uniquement les nombres nubers = [1,
2, 3, 4, 5] # itérable
```

```
def is_even (num): si num% 2
== 0: return true return false



Même_numbers = filtre (is_even, nombres) print (list (mêm
e_numbers)) # [2, 4]
```

Exemple: 2

```
nombres = [1, 2, 3, 4, 5] # itérable

def is_odd (num): si num% 2!
= 0: return vrai return false



odd_numbers = filtre (is_odd, nombres) print (list (odd_numbe
rs)) # [1, 3, 5]
```

```
# Filtre Noms Long Names = ['Asabeneh', 'lidiya', 'ermias', 'Abraham'] # itérable d
ef is_name_long (name): if Len (nom) > 7: return return false false_names = filter
(iS_name_long, noms) imprimer (list (list)
```

Python - Réduire la fonction

La fonction *reduce()* est définie dans le module Functools et nous devons l'importer à partir de ce module. Comme la carte et le filtre, il faut deux paramètres, une fonction et un itérable. Cep endant, il ne renvoie pas un autre itérable, mais il renvoie un seul

valeur. Exemple: 1

```
nombres_str = ['1', '2', '3', '4', '5'] # itérable def add_two_nums (x, y): return int
(x) + int (y)



Total = Réduire (add_two_nums, nombres_str) imprimer (total)
# 15
```

Exercices: Jour 14

```
pays = [«Estonie», «Finlande», «Suède», «Danemark»,
«Norvège», «Islande»]
```

```
Noms = ['Asabeneh', 'lidiya', 'ermias', 'Abraham'] nombres = [1, 2, 3, 4, 5, 6, 7,
8, 9, 10]
```

Exercices: niveau 1

1. Expliquez la différence entre la carte, le filtre et la réduction. 2. Expliquez la différen ce entre la fonction d'ordre supérieur, la fermeture et le décorateur 3. Définissez une fon ction d'appel avant la carte, filtrer ou réduire, voir des exemples. 4. Utiliser pour Loop p our imprimer chaque pays dans la liste des pays. 5. Utiliser pour imprimer chaque nom dans la liste des noms. 6. Utiliser pour imprimer chaque numéro dans la liste des nombr es.

Exercices: niveau 2

1. Utilisez la carte pour créer une nouvelle liste en modifiant chaque pays en majuscules d ans la liste des pays 2. Utilisez la carte pour créer une nouvelle liste en modifiant chaque n uméro en sa place dans la liste des numéros 3. Utilisez la carte pour modifier chaque nom en majuscules dans la liste des noms 4. Utilisez le filtre pour filtrer les pays contenant `` L and ". 5. Utilisez un filtre pour filtrer les pays ayant exactement six caractères. 6. Utiliser le filtre pour filtrer les pays contenant six lettres et plus dans la liste des pays. 7. Utilisez l e filtre pour filtrer les pays à commencer par un «E» 8. Chaîne de deux ou plusieurs itérate urs de liste (par exemple, arr 10. Utilisez Réduire pour résumer tous les nombres de la list e des nombres. 11. Utiliser Réduire pour concaténer tous les pays et pour produire cette p hrase: l'Estonie, la Finlande, la Suède, le Danemark, la Norvège et l'Islande sont en Europ e du Nord

Pays 12. Déclarer une fonction appelée catégorize_countries qui renvoie une liste de p ays avec un modèle commun (vous pouvez trouver la liste des pays dans ce référentiel co mme

Pays.js (par exemple, «terre», «ia», «île», «Stan»)).

13. Créer une fonction renvoyant un dictionnaire, où les clés représentent des lettres de d épart des pays et des valeurs sont le nombre de noms de pays à commencer par cette lettr e.

14. Déclarer une fonction GET_FIRST_TEN_COUNTRES - Il renvoie une liste des dix pr emiers pays de la liste des pays.js dans le dossier de données.

15. Déclarer une fonction GET_LAST_TEN_COUNTRES qui renvoie les dix derniers pay s de la liste des pays.

Exercices: niveau 3

1. Utilisez les pays_data.py (https://github.com/asabeneh/30-days-of- python / blob / m aître / data / pays-data.py) et suivez les tâches ci-dessous:

   - Trier les pays par nom, par capital, par la population o trier les dix langues les plus parlées par emplacement.

   - Trier les dix pays les plus peuplés.

Félicitations!