Fonctions du jour-11

Fonctions

Jusqu'à présent, nous avons vu de nombreuses fonctions Python intégrées. Dans cette section, nous nous concentrerons sur les fonctions personnalisées. Quelle est une fonction? Avant de c ommencer à faire des fonctions, apprenons ce qu'est une fonction et pourquoi nous en avons b esoin?

Définir une fonction

Une fonction est un bloc réutilisable d'instructions de code ou de programmation conçues pour effectuer une certaine tâche. Pour définir ou déclarer une fonction, Python fournit le mot-clé *def*. Ce qui suit est la syntaxe pour définir une fonction. Le bloc fonction du cod e est exécuté uniquement si la fonction est appelée ou invoquée.

Déclarer et appeler une fonction

Lorsque nous faisons une fonction, nous l'appelons déclarant une fonction. Lorsque nous com mençons à utiliser l'IT, nous l'appelons *calling* ou *invoking* une fonction. La fonction peut êtr e déclarée avec ou sans paramètres.

```
# syntaxe # déclarant une fonction
def function_name (): codes codes
# appelant une fonction function_n
ame ()
```

Fonction sans paramètres

La fonction peut être déclarée sans paramètres.

Exemple:

```
Def generate_full_name (): premier_name = 'asabeneh' last_name = 'ye
tayeh' espace = '' full_name = first_name + Space + last_name print (f
ull_name) Generate_full_name () # appelle un fonction ajout ajout_tw
o): Generate_full_name () # appelle a fonction afaf ad_two num_one =
2 num_two = 3 total = num_one + num_two print (total) add_two_nu
mbers ()
```

Fonction Renvoi d'une valeur - Partie 1

La fonction peut également renvoyer des valeurs, si une fonction n'a pas d'instruction de reto ur, la valeur de la fonction n'est aucune. Laissez-nous réécrire les fonctions ci-dessus à l'aide de retour. À partir de maintenant, nous obtenons une valeur à partir d'une fonction lorsque n ous appelons la fonction et l'imprimons.

```
Def Generate_full_name (): premier_name = 'asabeneh' last_name = 'e
ncoreayeh' espace = " full_name = first_name + Space + last_name ret
urn full_name print (generate_full_name ()) def ajout = 2 num_two = 3
total = num_one + num_two return total print (add_two_numbers ())
```

Fonction avec paramètres

Dans une fonction, nous pouvons transmettre différents types de données (numéro, chaîne, booléen, liste, tuple, dictionnaire ou ensemble) en tant que paramètre

- Paramètre unique: si notre fonction prend un paramètre, nous devons appeler notr e fonction avec un argument

```
# syntaxe
# Déclarant une fonction
Def function_name (paramètre):
codes
codes
# Fonction d'appel
print (function_name (argument))
```

Exemple:

```
Def Greetings (nom):
```

```
Message = name + ', bienvenue à Python pour tout le monde!'      return message print
(salutations ('asabeneh')) def add_ten (num): dix = 10 return num + ten print (add_ten
(90)) def square_number (x): return x * x print (square_number (2))
```

```
DEF Area_Of_Circle (R): PI = 3.14
Zone = Pi * R ** 2 RETOUR ZON
E

print (Area_Of_Circle (10))

def sum_of_numbers (n): total = 0 po
ur i dans la plage (n + 1): total += i i
mprimer (total)


print (sum_of_numbers (10)) # 55 PRINT (SUM_
OF_NUMBERS (100)) # 5050
```

- Deux paramètres: une fonction peut ou non avoir un paramètre ou des paramètres. Une fo nction peut également avoir deux paramètres ou plus. Si notre fonction prend des paramètr es, nous devons l'appeler avec des arguments. Vérifions une fonction avec deux paramètre s:

```
# syntaxe
# Déclarant une fonction
Def function_name (para1, para2):
codes
codes
# Fonction d'appel
print (function_name (arg1, arg2))
```

Exemple:

```
Def Generate_full_name (first_name, last_name): espace = " full_name = first_name + espace
+ last_name return full_name print ('full name:', generate_full_name ('asabeneh', 'yeayeh')))
```

```
def sum_two_numbers (num_one, num_two):
sum = num_one + num_two return sum
```

```
print ('somme de deux nombres:', sum_two_numbers (1, 9))

Def Calculate_age (Current_year, Birth_year): Age = Current_year -
Birth_year Return Age;

print ('Âge:', Calculate_age (2021, 1819)))


def poids_of_object (masse, gravité): poids = str (masse * gravity) + 'n' # La valeur doit être
changée en une chaîne de retour en poids de retour ('poids d'un objet dans les newtons:', poi
ds_of_object (100, 9.81)))
```

Passer des arguments avec clé et valeur

Si nous passons les arguments avec clé et valeur, l'ordre des arguments n'a pas d'importance
.

```
# syntaxe # déclarant une fonction def function_name (para1, para2): codes codes # app
elle function print (function_name (para1 = 'John', para2 = 'doe')) # L'ordre des argume
nts n'a pas d'importance ici
```

Exemple:

```
def print_fullName (FirstName, LastName): Space = " full_name = FirstName + Space +
LastName PRINT (Full_name) Print (print_fullname (FirstName = 'Asabeneh', LastName
= 'Yelayeh')) Def Add_two_nUM num2): total = num1 + num2 print (total) print (add_tw
o_numbers (num2 = 3, num1 = 2)) # L'ordre n'a pas d'importance
```

Fonction renvoyant une valeur - partie 2

Si nous ne renvoyons pas de valeur avec une fonction, notre fonction renvoie *None* par défa ut. Pour renvoyer une valeur avec une fonction, nous utilisons le mot-clé *return* suivi de la v ariable que nous renvoyons. Nous pouvons retourner tout type de types de données à partir d' une fonction.

- Renvoi d'une chaîne: Exemple:

```
def print_name (FirstName): retourne premier nom
print_name ('asabeneh') # asabeneh



def print_full_name (FirstName, LastName): Space = " full_name =
FirstName + Space + LastName return full_name



print_full_name (FirstName = 'Asabeneh', LastName = 'encoreayeh')
```

- Renvoi d'un nombre:

Exemple:

```
def add_two_numbers (num1, num2): total = num1
+ num2 return total print (add_two_numbers (2, 3))



Def Calculate_age (Current_year, Birth_year): Age = current_year -
naissance_year Return Age;

print ('Âge:', Calculate_age (2019, 1819)))
```

- Rendre un booléen: Exemple:

```
def is_even (n): si n% 2 == 0: print ('même') return true # return stops davantage l'exécution de
la fonction, similaire à briser return false print (is_even (10)) # true print (is_even (7)) # false
```

- Renvoi d'une liste: Exemple:

```
def find_even_numbers (n): evens = [] po
ur i dans la plage (n + 1): si i% 2 == 0: ev
ens.append (i) return evens



print (find_even_numbers (10))
```

Fonction avec paramètres par défaut

Parfois, nous passons des valeurs par défaut aux paramètres, lorsque nous invoquons la fon ction. Si nous ne transmettant pas les arguments lors de l'appel de la fonction, leurs valeurs par défaut seront utilisées.

```
# syntaxe # déclarant une fonction def function_na
me (param = valeur): codes codes # appelle functio
n function_name () function_name (arg)
```

Exemple:

```
Def Greetings (Name = 'Peter'): Message = name + ', bienvenue à Python pour tout le
monde!'      Return Message Print (salutations ()) imprimer (salutations ('asabeneh')))
```

```
Def Generate_full_name (first_name = 'asabeneh', last_name = 'yetayeh'): espace = " full_n
ame = first_name + Space + last_name return full_name}
```

```
print (generate_full_name ()) imprimer (generate_full_name ('d
avid', 'smith'))

Def Calculate_age (Birth_year, current_year = 2021): Age = current_year - Bi
rth_year Return Age;

Imprimer ('Age:', Calculate_age (1821)) def poids_of_object (masse, gravité = 9.81): poids
= str (masse * gravité) + 'n' # La valeur doit être modifiée en première chaîne de retour de p
oids imprimé ('poids d'un objet dans la terre:', weight_of_object Imprimer ('poids d'un objet
dans les newtons:', poids_of_object (100, 1,62)) # Gravité à la surface de la lune
```

Nombre arbitraire d'arguments

Si nous ne connaissons pas le nombre d'arguments que nous transmettons à notre fonction, nou s pouvons créer une fonction qui peut prendre un nombre arbitraire d'arguments en ajoutant * a vant le nom du paramètre.

```
# syntaxe # déclarant une fonction def function_name (* args)
: codes codes # appelle function function_name (param1, para
m2, param3, ..)
```

Exemple:

```
def sum_all_nums (* nums): total = 0 pour num in num: total += num # identique à
Total = total + num return total print (sum_all_nums (2, 3, 5)) # 10
```

Nombre par défaut et arbitraire de paramètres dans les fonctions

```
Def Generate_Groups (Team, * Args): Print (Team) pour I
dans Args: Print (i) Print (Generate_Groups ('Team- 1', 'As
abeneh', 'Brook', 'David', 'EYOB')))
```

Fonction comme un paramètre d'une autre fonction

```
#Vous pouvez passer les fonctions comme paramètres def carré_nu
mber (n): retour n * n def do_something (f, x): return f (x) imprimer
(do_something (carré_number, 3)) # 27
```

Vous avez réalisé beaucoup jusqu'à présent. Continue! Vous venez de terminer les défis d u jour 11 et vous êtes à 11 étapes de la grandeur à la grandeur. Faites maintenant quelques e xercices pour votre cerveau et vos muscles.

Témoignage

Il est maintenant temps d'exprimer vos réflexions sur l'auteur et le 30 jours. Vous pouvez lais ser votre témoignage sur ce lien

Exercices: Jour 11

Exercices: niveau 1

1. Déclarer une fonction *add_two_numbers*. Il prend deux paramètres et il renvoie une so mme.

2. La zone d'un cercle est calculée comme suit: zone = π x r x r. Écrivez une fonction qui c alcule *area_of_circle*. 3. Écrivez une fonction appelée add_all_nums qui prend un nombr e arbitraire d'arguments et résume tous les arguments. Vérifiez si tous les éléments de la lis te sont des types de nombres. Sinon, donnez un commentaire raisonnable. 4. La températu re en ° C peut être convertie en ° F en utilisant cette formule: ° F = (° C X 9/5) + 32. Écriv ez une fonction qui convertit ° C en ° F, *convert_celsius_to-fahrenheit*. 5. Écrivez une f onction appelée Check-Season, il faut un paramètre de mois et renvoie la saison: automne, w inter, printemps ou été. 6. Écrivez une fonction appelée calcul_slope qui renvoie la pent e d'une équation linéaire 7. L'équation quadratique est calculée comme suit: $ax^2 + bx + c = 0$. Écrivez une fonction qui calcule le jeu de solutions d'une équation quadratique, *solve_quadratic_eqn*. 8. Déclarer une fonction nommée print_list. Il prend une liste en t ant que paramètre et il imprime chaque élément de la liste. 9. Déclarer une fonction nomm ée Reverse_List. Il prend un tableau en tant que paramètre et il renvoie l'inverse du tableau (utilisez des boucles).

```
print (reverse_list ([1, 2, 3, 4, 5])) # [5, 4, 3, 2, 1] impri
mer (reverse_list1 (["a", "b", "c"]))

# ["C", "B", "A"]
```

10. Déclarer une fonction nommée CAPITalize_list_items. Il prend une liste en tant que paramètre et il renvoie une liste majuscule des éléments

11. Déclarer une fonction nommée add_item. Il faut une liste et un paramètre d'élément. I l renvoie une liste avec l'élément ajouté à la fin.

```
Food_staff = ['Potato', 'Tomato', 'Mango', 'Milk'] imprimer (add_item (aliments_staff, 'viande'))
# ['Potato', 'Tomato', 'mango', 'lait', 'viande'] nombres = [2, 3, 7, 9] print (add_item (nombre, 5))
[2, 3, 7, 5, 5, 5]
```

12. Déclarer une fonction nommée Remove_Item. Il faut une liste et un para mètre d'élément. Il renvoie une liste avec l'élément supprimé.

```
Food_staff = ['Potato', 'Tomato', 'Mango', 'Milk'] imprimer (retire_item (aliment
s_staff, 'mango')) # ['Potato',
«Tomate», «lait»];  Nombres = [2, 3, 7, 9] Imprimer (supprimer_it
em (nombres, 3)) # [2, 7, 9]
```

13. Déclarer une fonction nommée SUM_OF_NUMBERS. Il prend un paramètre de nombr e et il ajoute tous les nombres de cette plage.

```
print (sum_of_numbers (5)) # 15 PRINT (SUM_O
F_NUMBERS (10)) # 55 PRINT (SUM_OF_NUM
BERS (100)) # 5050
```

14. Déclarer une fonction nommée SUM_OF_ODDS. Il prend un paramètre de nombre e t ajoute tous les nombres impairs de cette plage.

15. Déclarer une fonction nommée SUM_OF_EVEN. Il prend un paramètre de nombre e t il ajoute tous les nombres pair dans cette plage.

Exercices: niveau 2

1. Déclarez une fonction nommée evens_and_odds. Il prend un entier positif comme p aramètre et il compte le nombre d'evens et les cotes dans le nombre.

```
 print (evens_and_odds (100))
 # Le nombre de cotes est de 50.
 # Le nombre d'Evens est de 51.
```

1. Appelez votre fonction factorielle, il prend un numéro entier en tant que paramètre et il renvoie un factoriel du numéro

2. Appelez votre fonction *is_empty*, il prend un paramètre et il vérifie s'il est vide ou non

3. Écrivez différentes fonctions qui prennent des listes. Ils doivent calculer_mean, ca lculer_median, calculer_mode, calculer_range, calculer_variance, calculer_std (écart -type).

Exercices: niveau 3

1. Écrivez une fonction appelée IS_PRIME, qui vérifie si un nombre est premier.

2. Écrivez une fonction qui vérifie si tous les éléments sont uniques dans la liste.

3. Écrivez une fonction qui vérifie si tous les éléments de la liste sont du même type de do nnées. 4. Écrivez une fonction qui vérifie si la variable fournie est une variable Python va lide 5. Accédez au dossier de données et accédez au fichier Pays-data.py. • Créez une fon ction appelée la plus_pose au monde. Il doit renvoyer 10 ou 20 langues les plus parlées du monde dans l'ordre descendant • créer une fonction appelée la plus_populaled_countries. I l devrait retourner 10 ou 20 pays les plus peuplés dans l'ordre descendant.

Félicitations!