Écrire et utiliser des fonctions

Module 6 • Paramètres et résultats • 60 minutes • Nom : _____

Votre mission : écrire une fonction que l'on peut réutiliser avec plusieurs valeurs. Vous apprendrez à définir, appeler et tester une fonction, puis à utiliser son résultat. Le pseudo-code reste notre point de départ ; le mémo et les aides sont page 3.

Comprendre un appel 12 min

Une fonction reçoit ici une valeur appelée paramètre et retourne un résultat. Définir la fonction prépare les instructions ; l'appeler les exécute avec la valeur fournie.

```
Fonction apres_retour(stock : entier) : entier
    Retourner stock + 2
Fin Fonction

depart ← 3
arrivee ← apres_retour(depart)
Afficher arrivee
Afficher depart
```

Dans apres_retour(depart), la valeur 3 est transmise au paramètre stock. Le résultat retourné est mémorisé dans arrivee. Ici, le calcul ne réaffecte pas depart.

Expression ou variable	Valeur attendue
apres_retour(0)	
apres_retour(5)	
arrivee après le programme	
depart après le programme	

Quelles sont les deux valeurs affichées, dans l'ordre ? _____ puis _____.
Quel est le nom du paramètre ? _____. Le type du résultat annoncé est _____.

Compléter une fonction 10 min

Après une distribution d'un câble, il reste un câble de moins. Complétez cette fonction, puis l'appel. On utilise seulement des stocks entiers supérieurs ou égaux à 1.

```
Fonction apres_pret(stock : entier) : entier
    Retourner __________
Fin Fonction

restant ← apres_pret(6)
Afficher __________
```

Résultat de apres_pret(6) : _____ ; de apres_pret(1) : _____.
La fonction affiche-t-elle elle-même quelque chose ? oui / non. Expliquez.

.....

.....

Retourner fournit une valeur à l'appelant. Afficher montre une valeur à l'écran. Ces deux actions ont des rôles différents.

Traduire déboguer et créer

Traduire en Python 10 min

Fonction devient def ; Retourner devient return. Le nom et le paramètre sont suivis de parenthèses et de deux-points. Le corps est décalé de quatre espaces. Traduisez apres_pret et le petit programme qui affiche son résultat pour 6.

.....

.....

.....

.....

.....

.....

Déboguer un résultat manquant 10 min

On veut mémoriser 5 dans restant puis afficher 5 une seule fois. Observez ce code avant de le lancer.

```
def apres_pret(stock):
    print(stock - 1)

restant = apres_pret(6)
print(restant)
```

1. Ce code affiche 5 puis None. Pourquoi restant ne reçoit-il pas 5 ?
2. Corrigez une seule ligne, en conservant l'appel et le dernier print.

.....

.....

.....

Repère : en Python, une fonction qui termine sans return renvoie None. Ce mot représente ici l'absence de résultat utile pour notre calcul.

Créer une fonction avec une condition 15 min

Le service vise un stock d'au moins 5 casques. Écrivez une fonction besoin(stock) qui retourne le nombre de casques manquants : si stock est inférieur à 5, retourner 5 - stock ; sinon retourner 0. Les stocks sont entiers positifs ou nuls. Écrivez le pseudo-code et Python au dos ou dans votre éditeur. La fonction doit retourner le résultat ; le programme principal l'affiche.

stock fourni	besoin(stock) prévu	Résultat vérifié
2		
5		
7		

Bilan individuel 3 min

Pour définir une fonction Python : _____ ; pour fournir son résultat : _____.
Pour obtenir le résultat de besoin avec le stock 2, j'écris _____.
Je distingue retourner et afficher : oui / avec aide / pas encore.

Mémo et aides de la fiche 6

Définir appeler et récupérer

```
Pseudo-code
Fonction apres_retour(stock : entier) : entier
    Retourner stock + 2
Fin Fonction
resultat ← apres_retour(3)
Afficher resultat

Python
def apres_retour(stock):
    return stock + 2

resultat = apres_retour(3)
print(resultat)
```

stock est le paramètre dans la définition ; 3 est l'argument fourni lors de cet appel. resultat reçoit 5. La fonction est définie avant son appel dans nos programmes. Le nom du paramètre n'a pas à être celui de la variable utilisée pour appeler la fonction.

Dans nos exemples sur des entiers, le paramètre est un nom local à la fonction. Chaque appel reçoit sa propre valeur ; le nom stock de la fonction ne remplace pas automatiquement une variable du programme principal.

Points de vigilance

return termine l'appel en cours et transmet son résultat. print affiche. Le type indiqué en pseudo-code décrit la valeur retournée ; nos premières définitions Python n'ajoutent pas d'annotation de type. Conservez les parenthèses lors de l'appel : apres_pret(6).

Aides graduées

Activité	Aide 1	Aide 2
Comprendre	Remplacez le paramètre par la valeur reçue.	Pour 3, calculez 3 + 2 ; depart reste 3.
Compléter	Un câble sort du stock.	Retournez stock - 1 et affichez restant.
Traduire	Reprenez le modèle ci-dessus.	Début : def apres_pret(stock): ; puis return.
Déboguer	Afficher ne fournit pas le résultat du calcul à restant.	Remplacez le print dans la fonction par return.
Créer	Prévoyez les deux cas.	Si stock < 5 : retourner 5 - stock ; sinon 0.

Aide 3 Pseudo-code complet de la création

```
Fonction besoin(stock : entier) : entier
    Si stock < 5 Alors
        Retourner 5 - stock
    Sinon
        Retourner 0
    Fin Si
Fin Fonction
```

Défi « Vers le CCF » : page 4. Le pseudo-code complet se consulte séparément en page 5, après un premier essai.

Vers le CCF assembler deux fonctions

Défi facultatif • Environ 15 minutes • À chercher sur papier puis à tester en Python

On reprend la réserve de la fiche 5 : 2 câbles le jour 1, 3 le jour 2, puis un câble de plus chaque jour. Cette fois, deux fonctions préparent le calcul. lot(jour) retourne la quantité ajoutée ce jour-là ; contenu(n) retourne le cumul des n premiers jours, avec contenu(0) = 0.

Compléter les fonctions

```
Fonction lot(jour : entier) : entier
    Retourner __________
Fin Fonction

Fonction contenu(n : entier) : entier
    total ← 0
    Pour jour de 1 à ______ Faire
        total ← total + __________
    Fin Pour
    Retourner __________
Fin Fonction
```

1. Complétez puis prévoyez lot(1), lot(4), contenu(2) et contenu(4).
2. Traduisez ces deux fonctions en Python et testez-les séparément avant de poursuivre. Ici, n est un entier positif ou nul et jour un entier supérieur ou égal à 1.

Compléter le programme principal

On cherche le premier jour où le contenu dépasse strictement 14.

```
jours ← 0
Tant Que __________ Faire
    jours ← __________
Fin Tant Que
Afficher jours
Afficher __________
```

3. Traduisez ce programme après les deux définitions Python. Vérifiez le nombre de jours et le contenu final.
4. Pourquoi faut-il appeler lot(jour), et non lot(n), dans la boucle de contenu ?

.....

.....

Test	Résultat prévu	Résultat obtenu
lot(1)		
lot(4)		
contenu(0)		
contenu(2)		
contenu(4)		
Premier jour au-delà de 14		

Aides graduées en page 5. Aucun calcul de formule de somme n’est demandé : on additionne les lots un par un.

Aides du défi de la fiche 6

Chercher avant de consulter la solution

Aide 1 : pour obtenir un lot, ajoutez 1 au numéro du jour. Pour un contenu, additionnez les lots des jours 1, 2, …, n.

Aide 2 : total commence à 0 avant la boucle. Utilisez lot(jour) dans le cumul, et Retourner après la boucle. Le programme principal reteste contenu(jours) à chaque passage.

Aide 3 Pseudo-code complet

```
Fonction lot(jour : entier) : entier
    Retourner jour + 1
Fin Fonction

Fonction contenu(n : entier) : entier
    total ← 0
    Pour jour de 1 à n Faire
        total ← total + lot(jour)
    Fin Pour
    Retourner total
Fin Fonction

jours ← 0
Tant Que contenu(jours) <= 14 Faire
    jours ← jours + 1
Fin Tant Que
Afficher jours
Afficher contenu(jours)
```

Passer à Python

Pour inclure le jour n, utilisez range(1, n + 1). Pour n = 0, range(1, 1) ne donne aucune valeur : la boucle ne passe pas et total garde la valeur 0. Définissez lot puis contenu avant le programme principal.

Rappel de calcul et de seuil

Les lots des quatre premiers jours sont 2, 3, 4, 5. Le cumul est 2 puis 2 + 3 puis 2 + 3 + 4 puis 2 + 3 + 4 + 5. Dépasser strictement 14 impose de continuer lorsque le contenu vaut exactement 14.

Vérifier sa compréhension

Refermez cette page, puis expliquez :

- ce que reçoit et retourne chaque fonction ;
- pourquoi total est initialisé dans contenu ;
- pourquoi return total se place après la boucle ;
- où s'effectuent les affichages.

.....

.....

.....

.....