NSI · PREMIÈRE / COURS

Fonctions, spécification et tests

Astrolab Cours · astrolabcours.fr

01 Le problème

Un programme de 200 lignes recopie trois fois le même calcul de moyenne. Il y a une erreur dedans : il faut la corriger à trois endroits, et on en oublie un. Comment écrire un morceau de code une fois et s'en servir partout ?

01 L'idée : la machine à café

Une machine à café prend des entrées (des pièces, un choix de boisson) et rend une sortie (un gobelet). Tu n'as pas besoin de savoir comment elle marche à l'intérieur, seulement ce qu'elle attend et ce qu'elle donne. Une **fonction**, c'est cette machine.

01 Les briques de base

```
age = 16                    # affectation
if age >= 18:               # condition
    print("majeur")
else:
    print("mineur")
for i in range(3):          # boucle bornée : 0, 1, 2
    print(i)
n = 100
while n > 1:                # boucle non bornée
    n = n // 2
```

`for` répète un nombre connu de fois ; `while` répète tant qu'une condition est vraie.

Définir une fonction

```
def moyenne(a, b):
    """Renvoie la moyenne des nombres a et b."""
    return (a + b) / 2
```

`a` et `b` sont les **paramètres** ; `return` renvoie le résultat et **termine** la fonction.

On l'appelle avec des **arguments** : `m = moyenne(12, 15)` donne 13.5.

return n'est pas print

© Astrolab Cours · astrolabcours.fr/nsi/premiere/bases-de-la-programmation/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

`return` donne le résultat au programme, `print` l'affiche seulement à l'écran. Une fonction sans `return` renvoie `None`.

Spécifier et tester

La **docstring** dit ce que fait la fonction. Une **précondition** dit ce qu'elle exige de ses entrées ; on peut la vérifier avec `assert` :

```
def racine_entiere(n):
    """Renvoie le plus grand entier r tel que r * r <= n. n est un entier positif
    assert n >= 0, "n doit être positif"
    r = 0
    while (r + 1) * (r + 1) <= n:
        r = r + 1
    return r

assert racine_entiere(10) == 3
assert racine_entiere(0) == 0
assert racine_entiere(16) == 4
```

Un bon jeu de tests contient des cas courants **et** des cas limites (0, valeur exacte, valeur négative).

Exemple guidé

Écrire une fonction qui calcule la moyenne de trois notes, puis l'utiliser pour trois élèves.

```
def moyenne3(a, b, c):
    """Renvoie la moyenne des nombres a, b et c."""
    return (a + b + c) / 3

assert moyenne3(10, 12, 14) == 12

m1 = moyenne3(12, 15, 9)    # 12.0
m2 = moyenne3(8, 11, 14)    # 11.0
m3 = moyenne3(17, 13, 15)   # 15.0
```

1. **Définir** : `a`, `b`, `c` sont les paramètres, des noms provisoires pour les valeurs qu'on donnera.
2. **Renvoyer** : `return` rend le résultat au programme, qui peut le stocker ou le réutiliser. Un `print` l'afficherait seulement.
3. **Tester** : l'assertion vérifie un cas dont on connaît la réponse.
4. **Appeler** : chaque appel remplace les paramètres par les vraies notes.

Retour au problème

Le calcul n'est écrit qu'une fois. Si on y trouve une erreur, par exemple une division par 2 au lieu de 3, on la corrige dans la fonction et les trois appels sont réparés d'un coup.

© Astrolab Cours · astrolabcours.fr/nsi/premiere/bases-de-la-programmation/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

Les pièges

- Confondre `=` (affectation) et `==` (comparaison).
- Afficher au lieu de renvoyer.
- Une boucle `while` dont la condition ne devient jamais fausse : boucle infinie.

© Astrolab Cours · astrolabcours.fr/nsi/premiere/bases-de-la-programmation/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits