Plaquette 5 – Problèmes de synthèse : programmes complets à écrire et à corriger

Algorithmique et Python — Seconde

Corrections détaillées

Boîte à outils

Méthode pour lire un programme. On construit un tableau de suivi, une colonne par variable, une ligne par tour de boucle. C'est long mais infaillible.

Les six erreurs les plus fréquentes.

Erreur	Symptôme
return dans la boucle	la fonction s'arrête au premier tour
mauvaise indentation	instruction exécutée trop souvent, ou pas assez
= au lieu de ==	erreur de syntaxe
borne de *range* oubliée	il manque le dernier terme
accumulateur mal initialisé	résultat décalé, ou toujours nul
mise à jour manquante dans un *while*	boucle infinie

Le piège de l'indentation. En Python, l'indentation **définit** les blocs. Un *return* décalé d'un niveau change complètement le comportement du programme.

Écrire un programme à partir d'un énoncé. Quatre étapes :

1. identifier les **entrées** (les paramètres de la fonction) ; 2. identifier la **sortie** (ce que renvoie *return*) ; 3. choisir la structure : *for* si le nombre de tours est connu, *while* sinon ; 4. traiter les cas limites : $n = 0$, valeur négative, division par zéro.

Tester un programme. On l'essaie sur un cas dont on connaît la réponse à la main, puis sur un cas limite. Un programme non testé n'est pas terminé.

Contrôler un résultat. Trois moyens : une formule mathématique, une seconde méthode de calcul, ou un ordre de grandeur. Par exemple, une boucle qui additionne les entiers de 1 à 100 se contrôle en les groupant par paires de somme 101 ($1 + 100$, $2 + 99$, …) :

$$1 + 2 + \cdots + 100 = 50 \times 101 = 5\,050$$

Correction de l'exercice 1 – Prévoir une sortie

Les variables a et b partent de 0, qui est la valeur de v pour $x = 0$; a retient la plus grande valeur rencontrée, b la plus petite.

x	0	1	2	3	4	5
v	0	−3	−4	−3	0	5
a	0	0	0	0	0	5
b	0	−3	−4	−4	−4	−4

À la fin, $a = 5$ est le maximum et $b = -4$ le minimum des valeurs de $x^2 - 4x$ pour x entier de 0 à 5. La fonction renvoie

$$5 - (-4) = 9$$

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

C'est l'**étendue** des valeurs calculées. L'initialisation à 0 fonctionne ici parce que 0 fait partie des valeurs ; en général, on initialise a et b avec la première valeur calculée.

Réponse. Elle renvoie 9 : l'étendue des valeurs de $x^2 - 4x$ pour x entier de 0 à 5 (maximum 5, minimum -4).

Correction de l'exercice 2 – Corriger une indentation

Le *return* est indenté **dans** la boucle : au premier tour, on ajoute 1 à 0, puis le *return* termine aussitôt la fonction, qui renvoie 1 quelle que soit la valeur de n. Il faut sortir le *return* d'un niveau, pour qu'il ne s'exécute qu'une fois la boucle terminée :

```
def somme(n):
    s = 0
    for k in range(1, n + 1):
        s = s + k
    return s
```

Sur $n = 4$, la fonction renvoie maintenant

$$1 + 2 + 3 + 4 = 10$$

Pour repérer ce genre d'erreur, on teste plusieurs valeurs : avant correction, *somme(4)* et *somme(100)* renvoient toutes deux 1, ce qui est suspect ; après correction, 10 et 5 050. Un *return* dans une boucle n'est pas toujours une erreur : dans une recherche, il permet de s'arrêter dès qu'on a trouvé. En Python, l'indentation n'est pas une question de présentation, elle définit la structure du programme.

Réponse. Le *return* doit être **après** la boucle, à l'indentation de la fonction.

Correction de l'exercice 3 – Corriger un range

range(n) parcourt les entiers de 0 à $n - 1$: la fonction calcule $0 + 1 + \cdots + (n - 1)$, et il manque exactement le dernier terme, n. Pour $n = 5$, elle renvoie $0 + 1 + 2 + 3 + 4 = 10$ au lieu de 15.

On corrige avec *range(1, n + 1)*, ou avec *range(n + 1)*, puisque ajouter 0 ne change rien. Pour $n = 5$:

$$1 + 2 + 3 + 4 + 5 = 15 = \frac{5 \times 6}{2}$$

Contrôle pour $n = 100$: la version corrigée donne $\frac{100 \times 101}{2} = 5050$, la version fautive $\frac{99 \times 100}{2} = 4950$; l'écart vaut bien 100. Pour $n = 0$, les deux versions renvoient 0, la somme d'aucun terme.

Réponse. Il faut *range(1, n + 1)* : la version initiale somme 0 à $n - 1$, il manque donc n.

Correction de l'exercice 4 – Corriger un accumulateur

Le produit est initialisé à 0, et $0 \times k = 0$: à chaque tour, p reste nul. Un accumulateur de produit doit partir de 1, l'élément neutre de la multiplication, comme une somme part de 0.

```
def produit_impairs():
    p = 1
    for k in range(1, 10, 2):
        p = p * k
    return p
```

range(1, 10, 2) donne 1, 3, 5, 7 et 9 : la borne 10 est exclue, et l'on avance de 2 en 2. Le

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

produit vaut

$$1 \times 3 \times 5 \times 7 \times 9 = 945$$

Contrôle : $1 \times 3 = 3$, puis 15, 105 et 945. Pour un maximum, il n'y a pas d'élément neutre : c'est pourquoi on l'initialise avec la première valeur rencontrée.

Réponse. Il faut $p = 1$: l'élément neutre de la multiplication. Le produit vaut 945.

Correction de l'exercice 5 – Écrire : moyenne et maximum d'une fonction

Les 17 valeurs de x s'écrivent $x = 0{,}5\,k$ pour k entier de 0 à 16. Un accumulateur donne la somme, et une variable *maxi* retient la plus grande valeur, initialisée avec $f(0)$.

```
def f(x):
    return -x * x + 8 * x

def etude():
    s = 0
    maxi = f(0)
    for k in range(17):
        v = f(0.5 * k)
        s = s + v
        if v > maxi:
            maxi = v
    return s / 17, maxi
```

Les valeurs sont symétriques autour de $x = 4$; leur somme vaut 170, d'où une moyenne de $\dfrac{170}{17} = 10$. Le maximum est atteint en $x = 4$:

$$f(4) = -16 + 32 = 16$$

On le justifie par le calcul : $16 - f(x) = x^2 - 8x + 16 = (x-4)^2 \geq 0$, donc $f(x) \leq 16$ pour tout x. Pour parcourir des valeurs décimales, on fait varier un entier k et l'on calcule x à partir de k : *range* n'accepte que des entiers.

Réponse. Moyenne 10 et maximum 16 (atteint en $x = 4$).

Correction de l'exercice 6 – Écrire : évolution d'une masse d'algues

Perdre 20 % revient à multiplier par 0,8 ; on ajoute ensuite 3 tonnes. On répète cette opération n fois avec une boucle *for*.

```
def algues(n):
    x = 2
    for i in range(n):
        x = 0.8 * x + 3
    return x
```

Les masses successives sont 4,6, 6,68, 8,344, 9,6752 puis

$$0{,}8 \times 9{,}6752 + 3 = 10{,}74016 \text{ tonnes}$$

La masse augmente, mais de moins en moins : les hausses successives 2,6, 2,08, 1,664… sont chaque fois multipliées par 0,8. La masse d'équilibre x vérifie $x = 0{,}8x + 3$, soit $0{,}2x = 3$ et $x = 15$ tonnes. **Contrôle** : $0{,}8 \times 15 + 3 = 15$. Le programme donne environ 13,6 tonnes après 10 ans et 14,85 tonnes après 20 ans : la masse se rapproche bien de 15 tonnes.

Réponse. 10,74016 tonnes au bout de 5 ans ; la masse d'équilibre est de 15 tonnes.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Correction de l’exercice 7 – Écrire : résolution d’équation par balayage

On part de 2 et l’on avance de 0,01 en 0,01 tant que $x^2 < 7$.

```
x = 2
```

```
pas = 0.01
while x * x < 7:
    x = x + pas
print(x)
```

En 2,64, $x^2 = 6{,}9696 < 7$: la boucle continue. En 2,65, $x^2 = 7{,}0225 \geq 7$: la boucle s’arrête, après 65 pas. On en déduit l’encadrement

$$2{,}64 < \sqrt{7} < 2{,}65$$

et en effet $\sqrt{7} \approx 2{,}6458$. Python affiche en réalité 2.649999999999986 : les décimaux sont stockés en binaire, et chaque ajout de 0,01 introduit une infime erreur. On l’évite en comptant les pas avec un entier k et en calculant $x = 2 + k / 100$. Le balayage est simple, mais lent : la dichotomie atteint la même précision en 7 étapes, car $\dfrac{1}{2^7} \approx 0{,}0078 < 0{,}01$.

Réponse. Le programme affiche ≈ 2,65, encadrant $\sqrt{7} \approx 2{,}6458$.

Correction de l’exercice 8 – Écrire : conversion de durée

Une heure fait 3600 secondes et une minute 60. Le quotient entier `//` donne les heures ; avec le reste, `%`, on recommence pour les minutes.

```
def convertit(t):
```

```
    h = t // 3600
    reste = t % 3600
    m = reste // 60
    s = reste % 60
    return h, m, s
```

Pour $t = 10\,000$: $10\,000 = 3600 \times 2 + 2800$, donc 2 heures et 2800 secondes restantes ; puis $2800 = 60 \times 46 + 40$, donc 46 minutes et 40 secondes.

$$10\,000 \text{ s} = 2 \text{ h } 46 \text{ min } 40 \text{ s}$$

Contrôle : $2 \times 3600 + 46 \times 60 + 40 = 7200 + 2760 + 40 = 10\,000$. Avec `/` au lieu de `//`, on obtiendrait 2,777… heures, et en divisant t au lieu de *reste* par 60, on trouverait 166 minutes, beaucoup trop. Pour 86 400 secondes, la fonction renvoie 24 h 0 min 0 s : une journée.

Réponse. $10\,000$ s = 2 h 46 min 40 s.

Correction de l’exercice 9 – Écrire : tableau de signes

La fonction f calcule l’image, et le test à trois branches *if*, *elif*, *else* classe chaque valeur : *elif* n’est examiné que si la première condition est fausse.

```
def f(x):
```

```
    return x * x - 5 * x + 6

for x in range(7):
    if f(x) > 0:
        print(x, "positif")
    elif f(x) == 0:
        print(x, "nul")
    else:
        print(x, "négatif")
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s’expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Les images de 0 à 6 sont 6, 2, 0, 0, 2, 6, 12 : le programme affiche « nul » pour 2 et 3, « positif » partout ailleurs. On vérifie par le développement :

$$(x-2)(x-3) = x^2 - 3x - 2x + 6 = x^2 - 5x + 6$$

donc $f(x) = 0$ exactement pour $x = 2$ et $x = 3$. Mais entre 2 et 3, les deux facteurs sont de signes contraires : par exemple $f(2{,}5) = -0{,}25$. Aucun entier n'est dans cet intervalle, donc le programme ne voit jamais de valeur négative. Un balayage au pas de 0,1 les ferait apparaître : un pas trop grand peut masquer un changement de signe.

Réponse. f s'annule en $x = 2$ et $x = 3$; mais le balayage entier ne voit pas que f est négative entre 2 et 3.

Correction de l'exercice 10 – Corriger une condition

Avec *or*, la condition est vraie dès que **l'une** des deux inégalités l'est. Or tout nombre vérifie l'une d'elles : si $x < 10$, alors $x \leq 20$, et si $x > 20$, alors $x \geq 10$. La fonction compte donc tous les entiers et renvoie 30.

Il faut que les **deux** inégalités soient vraies en même temps : on écrit *if x >= 10 and x <= 20*, ou plus simplement *if 10 <= x <= 20*. La fonction compte alors les entiers de 10 à 20, au nombre de

$$20 - 10 + 1 = 11$$

Les bornes 10 et 20 sont comptées car les inégalités sont larges ; avec < et >, il en resterait 9. Pour tester une condition d'intervalle, on l'essaie sur un nombre **hors** de l'intervalle, par exemple 5 : il doit donner faux, ce que la version avec *or* ne faisait pas.

Réponse. Il faut *and* et non *or* (ou l'écriture *10 <= x <= 20*) ; la fonction renvoie alors 11.

Correction de l'exercice 11 – Écrire : recherche du plus proche

La distance entre x^2 et 50 est *abs(x * x - 50)*. On garde dans *meilleur* l'entier le plus proche trouvé jusqu'ici, en partant de 0.

```
def plus_proche(v):
    meilleur = 0
    for x in range(0, 11):
        if abs(x * x - v) < abs(meilleur * meilleur - v):
            meilleur = x
    return meilleur
```

Pour $v = 50$, les carrés voisins sont 49 pour $x = 7$ (distance 1) et 64 pour $x = 8$ (distance 14) ; tous les autres sont plus loin. La fonction renvoie

$$7^2 = 49 \quad \text{et} \quad |49 - 50| = 1$$

La fonction renvoie donc 7.

On en déduit que $\sqrt{50}$ est proche de 7 : en effet $\sqrt{50} \approx 7{,}07$. En cas d'égalité de distance, l'inégalité stricte garde le premier entier trouvé. Pour $v = 200$, la fonction renverrait 10, la plus grande valeur testée, qui n'est pas forcément la bonne réponse : il faudrait élargir la boucle.

Réponse. 7 (car $7^2 = 49$ est à distance 1 de 50).

Correction de l'exercice 12 – Écrire : le programme de Syracuse

On ne sait pas d'avance combien d'étapes il faudra : on utilise une boucle *while*. Le test de parité se fait avec le reste *n % 2*, et l'on retient le maximum au fil des étapes.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

```
def vol(n):
    etapes = 0
    maxi = n
    while n != 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
        etapes = etapes + 1
        if n > maxi:
            maxi = n
    return etapes, maxi
```

Pour $n = 7$, on obtient successivement 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, soit

16 étapes, avec un maximum de 52

On utilise *n // 2* plutôt que *n / 2* pour que n reste un entier. Personne n'a jamais démontré que la boucle s'arrête pour tout entier de départ : c'est la conjecture de Syracuse, vérifiée par ordinateur jusqu'à des nombres gigantesques.

Réponse. Pour $n = 7$: 16 étapes et une valeur maximale de 52.

Correction de l'exercice 13 – Corriger une boucle infinie

La ligne *p = 2 * n* calcule le double du compteur au lieu du double de p : p augmente de 2 à chaque tour au lieu de doubler. Le programme ne s'arrête pas vraiment jamais, mais il ne s'arrête que lorsque $2n > 1000$, et affiche 502 au lieu du résultat attendu. On corrige en écrivant *p = 2 * p* :

```
n = 0
p = 1
while p <= 1000:
    p = 2 * p
    n = n + 1
print(n)
```

p prend les valeurs 2, 4, 8, …, 512, puis 1024 : la boucle fait 10 tours et le programme affiche $n = 10$.

$$2^9 = 512 \leq 1000 < 1024 = 2^{10}$$

Donc 10 est bien le plus petit exposant qui convient. De même, il faut $n = 20$ pour dépasser un million, car $2^{20} = 1\,048\,576$: comme $2^{10} \approx 10^3$, on a $2^{20} \approx 10^6$.

Réponse. Il faut *p = 2 * p* ; le résultat est alors $n = 10$, car $2^{10} = 1024 > 1000$.

Correction de l'exercice 14 – Écrire : la plus forte hausse

Pour comparer chaque valeur à la précédente sans les stocker, on garde la valeur de l'heure précédente dans une variable *precedent*.

```
def T(h):
    return -0.1 * h * h + 2.4 * h + 8

def plus_forte_hausse():
    precedent = T(0)
    record = T(1) - T(0)
    heure = 0
    for h in range(1, 24):
        hausse = T(h) - precedent
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

```
        if hausse > record:
            record = hausse
            heure = h - 1
        precedent = T(h)
    return record, heure
```

Le calcul montre que la hausse entre h et $h+1$ vaut

$$T(h+1) - T(h) = -0{,}2h + 2{,}3$$

Elle diminue quand h augmente : la plus forte hausse est la première, 2,3 °C entre 0 h et 1 h ($T(0) = 8$ et $T(1) = 10{,}3$). La fonction renvoie donc 2,3 et 0, aux arrondis des flottants près. Après midi, les « hausses » deviennent négatives : la serre se refroidit.

Réponse. 2,3 °C, entre 0 h et 1 h.

Correction de l’exercice 15 – Comparer deux programmes

A ajoute les carrés 1^2, 2^2, …, n^2 ; B ajoute les entiers de 1 à n, puis élève le total au carré. Un seul exemple suffit à montrer qu’ils diffèrent. Pour $n = 3$:

$$\text{A} : 1 + 4 + 9 = 14 \qquad \text{B} : (1 + 2 + 3)^2 = 36$$

Les deux programmes ne calculent donc pas la même chose. Pour $n = 1$, ils donnent tous deux 1 : avec un seul terme, le carré de la somme est la somme des carrés. Mais dès deux termes, le développement $(a+b)^2 = a^2 + 2ab + b^2$ fait apparaître le terme $2ab$, qui manque dans A. Pour $n = 4$, A donne $1 + 4 + 9 + 16 = 30$ et B donne $10^2 = 100$: l’écart grandit avec n. Curiosité : 100 est aussi $1^3 + 2^3 + 3^3 + 4^3 = 1 + 8 + 27 + 64$.

Réponse. Non : A calcule la somme des carrés (14 pour $n = 3$), B le carré de la somme (36 pour $n = 3$).

Correction de l’exercice 16 – Synthèse : analyser les relevés d’un capteur

a) Un seul parcours donne la somme et les deux extrêmes, initialisés avec le premier relevé ; on divise par le nombre de relevés après la boucle.

```
def indicateurs():
    s = 0
    mini = T(0)
    maxi = T(0)
    for h in range(24):
        v = T(h)
        s = s + v
        if v < mini:
            mini = v
        if v > maxi:
            maxi = v
    return s / 24, mini, maxi
```

b) Un compteur augmenté de 1 quand la température dépasse le seuil :

```
def heures_au_dessus(seuil):
    c = 0
    for h in range(24):
        if T(h) > seuil:
            c = c + 1
    return c
```

c) Les relevés montent de 8 °C à 0 h jusqu’à 22,4 °C à 12 h, puis redescendent jusqu’à 10,3 °C à 23 h. La somme des 24 relevés vaut 422, d’où

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s’expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

$$\text{moyenne} = \frac{422}{24} \approx 17{,}58\ °\text{C}$$

Le minimum est 8 °C (0 h) et le maximum 22,4 °C (12 h). La température dépasse 20 °C de 8 h (20,8 °C) à 16 h (20,8 °C), soit pendant 9 relevés ; à 7 h et à 17 h, elle vaut 19,9 °C.

d) Chaque valeur est **recalculée** par la fonction T au moment où l'on en a besoin, et les indicateurs se mettent à jour au fil de la boucle : il n'est jamais nécessaire de garder les 24 valeurs en mémoire. Une liste deviendrait utile pour des relevés réels, qu'on ne peut pas recalculer ; c'est une notion de première.

Réponse. c) Moyenne ≈ 17,58 °C, minimum 8 °C, maximum 22,4 °C, 9 relevés au-dessus de 20 °C ; d) les valeurs sont recalculées par T.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).