2nde / Python : algorithmique et programmation

benprof particulier.fr

48 exercices en 6 parties, du plus simple au plus difficile : ★☆☆ facile · ★★☆ moyen · ★★★ difficile. Le corrigé détaillé est sur **benprofparticulier.fr**.

1. Variables, types et affectation

À retenir : = range une valeur : `x = x + 1` augmente x de 1. == compare et vaut True ou False. / donne un float (9/3 vaut 3.0), // le quotient, % le reste, ** la puissance. `input` renvoie un str : on écrit `int(input(…))`.

Ex. 1 ★☆☆ **Automatismes : types et affectations** — Réponds sans ordinateur.

a) Donne le type de chacune de ces valeurs : `-4` ; `4.0` ; `"4"` ; `False`.

b) Que vaut `15 / 5` ? Quel est son type ?

c) La variable n vaut 7. Que vaut n après l'instruction `n = n + 1` ?

d) La variable p vaut 3. Que vaut `p == 4` ? Que vaut p ensuite ?

e) Ben écrit `prix = 3,75` pour ranger 3,75 dans prix. Corrige son instruction.

f) Vrai ou faux ? Après `a = 10` puis `a = 4`, la variable a contient 10 et 4. Justifie.

Ex. 2 ★☆☆ **Automatismes : les opérations** — Réponds sans calculatrice.

a) Qu'affichent `print(23 // 4)` et `print(23 % 4)` ? Écris l'égalité de la division euclidienne qui le justifie.

b) Noé pense que `print(10 / 4)` affiche 2, comme `print(10 // 4)`. A-t-il raison ? Justifie.

c) Que valent `2 + 5 * 3` et `(2 + 5) * 3` en Python ?

d) Tom écrit `3 ^ 4` pour calculer 3^4. A-t-il raison ? Justifie et donne la bonne instruction.

e) Avec // et %, écris le calcul qui donne le nombre de semaines complètes dans 45 jours, puis celui qui donne le nombre de jours restants. Donne les deux résultats.

Ex. 3 ★★☆ **Places de cinéma** — Un groupe achète des places de cinéma. Voici le script de la caisse :

```
prix = 9
places = 4
total = prix * places
prix = prix - 2
total = total + places
print(total)
print(prix)
```

a) Complète la trace d'exécution :

Ligne	prix	places	total
1	9	–	–
2	?	?	–
3	?	?	?
4	?	?	?
5	?	?	?

b) Qu'affiche ce script ?

c) Lina pense que la 4e ligne modifie aussi total, puisque `total = prix * places`. A-t-elle raison ? Justifie.

d) On échange la 3e et la 4e ligne. Qu'affiche alors le script ?

Ex. 4 ★★☆ **Températures inversées** — Ben a saisi à l'envers les températures du jour : il faisait 19 °C le matin et 12 °C le soir, après un orage. Pour échanger les deux valeurs, il écrit :

```
matin = 12
soir = 19
matin = soir
soir = matin
print(matin)
print(soir)
```

a) Complète la trace des lignes 3 et 4 :

Étape	matin	soir
Ligne 2	12	19
Ligne 3	?	?
Ligne 4	?	?

b) Qu'affiche ce script ? Les deux valeurs sont-elles échangées ? Justifie.

c) Ben ajoute à la fin de ce script `ecart = matin - soir`. Que vaut ecart ? Quel écart de température attendait-il ?

Ex. 4 **(suite)**

(d) Complète ce programme pour qu'il échange vraiment les deux valeurs :

```
matin = 12
soir = 19
temp = ...
matin = ...
soir = ...
print(matin)
print(soir)
```

Ex. 5 ★★☆ **Les points d'un jeu** — Dans un jeu, Inès marque des points en deux manches. Elle veut afficher son total avec ce script :

```
a = input("Points manche 1 : ")
b = input("Points manche 2 : ")
print(a + b)
```

(a) Inès tape 4 puis 6. Qu'affiche le script ?

(b) Quel est le type de la variable a ? Explique le résultat de la question précédente.

(c) Corrige le script pour qu'il affiche le total des points.

(d) Dans un autre programme, Inès veut ranger dans taille sa taille en mètres, tapée au clavier. Écris l'instruction. Que se passe-t-il si elle tape 1,62 ?

Ex. 6 ★★★ **Le chrono du trail** — Sur un trail, le chronomètre de Ben donne la durée de course en secondes. Ben veut l'afficher en heures, minutes et secondes. Voici son programme à compléter :

```
duree = int(input("Durée en secondes : "))
heures = ...
reste = ...
minutes = ...
secondes = ...
print(heures, minutes, secondes)
```

(a) Sans ordinateur, convertis 8 000 s en heures, minutes et secondes. Écris les deux divisions euclidiennes que tu utilises.

(b) Complète les quatre lignes à trous en utilisant uniquement `//` et `%`. Qu'affiche le programme si l'on tape 5000 ?

(c) Léo recopie ton programme mais remplace chaque `//` par `/`. Qu'affiche son programme si l'on tape 8000 ? Explique son erreur.

(d) Léo a aussi oublié `int(…)` à la ligne 1. Que se passe-t-il à la ligne 2 ? Justifie à l'aide du type de `duree`.

(e) Ben ajoute une pénalité de 30 s : il écrit `duree = duree + 30` juste après la ligne 1. Un camarade dit : « Cette équation n'a pas de solution. » Réponds-lui, puis donne l'affichage obtenu si l'on tape 8000.

Ex. 7 ★★★ **Soldes : l'ordre compte** — Une veste coûte 80 €. Le magasin propose une remise de 25 % et un bon de 10 €. La caisse peut appliquer les deux réductions dans un ordre ou dans l'autre :

```
# Programme A
prix = 80
prix = prix * 0.75
prix = prix - 10
print(prix)

# Programme B
prix = 80
prix = prix - 10
prix = prix * 0.75
print(prix)
```

(a) Dresse la trace du programme A (une ligne par instruction) et donne ce qu'il affiche.

(b) Dresse la trace du programme B (une ligne par instruction) et donne ce qu'il affiche.

(c) Le vendeur affirme : « Pour n'importe quel prix p, les deux programmes donnent le même résultat, puisqu'ils contiennent les mêmes instructions. » Vrai ou faux ? Justifie par un calcul.

(d) Écris un programme qui demande le prix initial (il peut être décimal), calcule `prix_a` et `prix_b`, puis affiche ces deux prix et le booléen `prix_a == prix_b`.

(e) À la fin des soldes, le prix initial baisse deux fois de 25 %. Léa écrit `80 * 0.75 ^ 2`. Explique son erreur, corrige-la et donne la valeur obtenue.

Ex. 8 ★★★ **La rotation au volley** — Au volley, Lina, Tom et Sami occupent trois postes. À chaque rotation, le joueur du poste 2 passe au poste 1, celui du poste 3 passe au poste 2 et celui du poste 1 passe au poste 3. Zoé programme une rotation :

```
poste1 = "Lina"
poste2 = "Tom"
poste3 = "Sami"
poste1 = poste2
poste2 = poste3
poste3 = poste1
print(poste1, poste2, poste3)
```

(a) Complète la trace des lignes 4 à 6, puis donne ce qu'affiche le script. Explique pourquoi la rotation de Zoé est fausse.

Ligne	poste1	poste2	poste3
3	Lina	Tom	Sami
4	?	?	?
5	?	?	?
6	?	?	?

(b) Corrige les lignes 4 à 6 en utilisant une seule variable supplémentaire `aux`. Donne l'affichage obtenu.

(c) Au bout de combien de rotations, au minimum, chaque joueur retrouve-t-il son poste de départ ? Déduis-en, avec `%`, qui occupe le poste 1 après 10 rotations.

Ex. 8 **(suite)**

(d) Dans le programme corrigé, on insère `capitaine = poste1` juste après la ligne 3. Que contient `capitaine` après la rotation ? Justifie.

(e) Après la rotation corrigée, qu'affiche `print(poste3 == "Lina")` ? Quel est le type de cette valeur ? Pourquoi `print(poste3 = "Lina")` ne convient-il pas ?

2. *Instruction conditionnelle*

À retenir : Ligne du test finie par « : », bloc décalé de 4 espaces ; égalité : ==. Avec `elif`, seul le premier test vrai s'exécute : le cas le plus restrictif d'abord. « À partir de 16 » : $n \geqslant 16$, soit `n >= 16`.

Ex. 9 ★☆☆ **Valeur d'une condition** — On a exécuté `a = 9` puis `b = -2`. Donne la valeur (`True` ou `False`) de chaque condition.

(a) `a > 9`

(b) `a >= 9`

(c) `a > 5 and b > 0`

(d) `a > 5 or b > 0`

(e) `not (b < 0)`

(f) `a % 2 == 0`

(g) `-5 <= b <= 0`

Ex. 10 ★☆☆ **Écrire un test sans erreur** — Réponds à chaque question en une ligne.

(a) La ligne `if x = 5:` est-elle correcte ? Corrige-la si besoin.

(b) La ligne `if vitesse >= 90` est-elle correcte ? Corrige-la si besoin.

(c) Sous `if a > 0:`, la ligne `print("positif")` est écrite sans décalage. Que se passe-t-il ?

(d) Vrai ou faux ? Un `if` doit toujours être suivi d'un `else`. Justifie.

(e) Écris la ligne de test « si l'entier `n` est pair ».

(f) Écris le test « $15 \leqslant n \leqslant 40$ » de deux façons.

Ex. 11 ★★☆ **Tarif de cinéma** — Un cinéma affiche : « moins de 14 ans : 5 € ; de 14 à 23 ans : 7 € ; à partir de 24 ans : 10 € ». Ben écrit cette fonction.

```
def tarif(age):
    if age < 14:
        return 5
    elif age < 24:
        return 7
    else:
        return 10
```

(a) Donne `tarif(9)`, `tarif(19)` et `tarif(40)`. Combien paient deux jeunes de 9 et 19 ans avec deux adultes de 40 ans ?

Ex. 11 **(suite)**

(b) Donne `tarif(14)` et `tarif(24)`. La fonction respecte-t-elle l'affiche aux frontières ?

(c) Ben remplace `age < 24` par `age <= 24`. Pour quel âge le résultat change-t-il ? Conclus.

(d) Le cinéma ajoute « gratuit pour les moins de 3 ans ». Où places-tu ce test ? Écris la fonction complète.

Ex. 12 ★★☆ **Badge de marche : trouver l'erreur** — Une appli donne un badge selon le nombre de pas de la journée : « or » à partir de 10 000, « argent » à partir de 7 000, « bronze » à partir de 4 000, « aucun » sinon. Lina écrit :

```
def badge(pas):
    if pas >= 4000:
        return "bronze"
    elif pas >= 7000:
        return "argent"
    elif pas >= 10000:
        return "or"
    else:
        return "aucun"
```

(a) Que renvoie `badge(12500)` ? Quel badge devrait-on obtenir ? Justifie.

(b) Explique l'erreur de Lina.

(c) Écris la fonction corrigée.

(d) Avec ta fonction, donne `badge(3999)`, `badge(4000)`, `badge(7000)` et `badge(10000)`.

Ex. 13 ★★☆ **Jeu de deux dés** — On lance deux dés. Si les deux faces sont égales (double), on rejoue ; sinon, si la somme est paire, on gagne 2 points ; sinon, on perd. Complète ce programme.

```
de1 = 3
de2 = 5
if ...:
    print("rejoue")
elif ...:
    print("gagne 2 points")
else:
    print("perd")
```

(a) Complète les deux tests du programme.

(b) Qu'affiche le programme pour `de1 = 3` et `de2 = 5` ?

(c) Qu'affiche le programme pour `de1 = 4` et `de2 = 4` ? La somme 8 est paire : explique le rôle du `elif` pour ces valeurs.

(d) Donne deux faces qui font afficher « perd ». Justifie avec %.

Ex. 14 ★★★ **Années bissextiles : corriger et réordonner** — Une année est bissextile si elle est divisible par 4 mais pas par 100, ou si elle est divisible par 400. Tom écrit cette fonction.

```
def bissextile(annee):
    if annee % 4 == 0:
        return True
    elif annee % 100 == 0:
        return False
    elif annee % 400 == 0:
        return True
    else:
        return False
```

(a) Que renvoient `bissextile(2032)`, `bissextile(2025)` et `bissextile(1800)` ? Quel résultat est faux ?

(b) Justifie que les tests des deux `elif` ne sont jamais vrais quand Python les examine.

(c) Réécris la fonction avec `if` / `elif` / `else` en plaçant le cas le plus restrictif en premier.

(d) Écris une autre version qui ne contient qu'une seule instruction `return` et qui utilise `and`, `or` et `not`.

(e) Donne les valeurs renvoyées par ta fonction pour 2000, 2100, 2024 et 2026. Explique pourquoi ces quatre années suffisent à tester chaque cas.

Ex. 15 ★★★ **Serre : and ou or ?** — Dans une serre, la température `t` (en °C) est correcte si elle est comprise entre 18 et 25, bornes comprises. Sinon, une alerte doit se déclencher. Lou écrit cette fonction.

```
def etat(t):
    if t < 18 and t > 25:
        return "alerte"
    else:
        return "normal"
```

(a) Que renvoient `etat(12)`, `etat(21)` et `etat(31)` ? Compare avec les résultats attendus.

(b) Pour quelles valeurs de `t` la condition `t < 18 and t > 25` est-elle vraie ? Justifie, puis corrige la ligne du test.

(c) Écris le test de l'alerte d'une autre façon, en utilisant `not`. Vérifie-le pour $t = 12$ et $t = 25$.

(d) Écris un script sans `else` qui commence par `t = 25` et affiche "chauffage" si $t < 18$, "ventilation" si $t > 25$, rien sinon. Qu'affiche-t-il ?

(e) Mia remplace dans ton script `<` par `<=` et `>` par `>=`. Pour quelles valeurs affectées à `t` les deux scripts affichent-ils des résultats différents ? Justifie.

Ex. 16 ★★★ **Jeu des cartes numérotées** — Dans un jeu, on tire une carte numérotée de 1 à 50. Si son numéro `n` est un multiple de 6, on gagne 10 points. Sinon, si `n` est pair, on gagne 3 points. Sinon, si `n` est un multiple de 3, on gagne 2 points. Sinon, on gagne 0 point.

(a) Écris une fonction `points(n)` qui renvoie le nombre de points gagnés, avec `if` / `elif` / `else`.

(b) Calcule `points(42)`, `points(34)`, `points(27)` et `points(35)`.

Ex. 16 (suite)

(c) Léa remplace chaque `elif` de ta fonction par `if`. Que renvoie sa fonction pour 42 ? Pour quels numéros de 1 à 50 diffère-t-elle de la tienne ? Justifie.

(d) Vrai ou faux ? Justifie. « Si `n` est impair, `points(n)` vaut 0. »

(e) Écris une fonction `manche(a, b)` qui appelle `points` et renvoie "A gagne", "B gagne" ou "égalité". Que renvoie `manche(30, 44)` ?

3. *Boucle bornée for*

À retenir : `range(a, b)` donne a, a + 1, …, b − 1 : $b - a$ tours, borne de droite exclue (`range(2, 5)` → 2, 3, 4) ; `range(n)` part de 0. Somme et compteur à 0, produit à 1, AVANT la boucle. Un `print` indenté s'exécute à chaque tour.

Ex. 17 ★☆☆ **Lire un range** — Automatismes. Réponds sans écrire le programme en entier.

(a) Écris les valeurs prises par k dans `for k in range(5):`.

(b) Combien de tours fait la boucle `for k in range(3, 8):` ?

(c) Écris les valeurs prises par k dans `for k in range(2, 15, 4):`.

(d) Vrai ou faux ? Avec `for k in range(10, 0, -3):`, k prend la valeur 0. Justifie.

(e) Lina veut que k aille de 1 à 100. Elle écrit `for k in range(1, 100):`. A-t-elle raison ? Justifie.

(f) Avec `for k in range(0, 20, 5):`, quelle est la dernière valeur de k ? Combien de tours fait la boucle ?

Ex. 18 ★☆☆ **Choisir le bon range** — Automatismes. Pour chaque cas, écris la ligne `for k in range(…):` qui convient.

(a) k prend les valeurs 0, 1, 2, …, 11.

(b) k prend les valeurs 5, 6, 7, …, 40. Combien de tours fait alors la boucle ?

(c) k prend les nombres pairs de 2 à 20 inclus.

(d) k prend les valeurs 30, 25, 20, …, 5.

(e) La boucle commence avec k égal à 3, avance de 1 en 1 et fait 7 tours. Quelle est la dernière valeur de k ?

Ex. 19 ★★☆ **Trace d'une boucle qui utilise k** — On considère le programme suivant.

```
a = 2
for k in range(1, 5):
    a = 2 * a - k
print(a)
```

(a) Quelles valeurs prend k ? Combien de tours fait la boucle ?

Ex. 19 **(suite)**

(b) Recopie et complète la trace d'exécution :

Tour	k	a
Avant	—	2
1	?	?
2	?	?
3	?	?
4	?	?

(c) Qu'affiche le programme ?

(d) Ben décale la ligne `print(a)` de 4 espaces vers la droite, sous le `for`. Qu'affiche alors le programme ?

Ex. 20 ★★☆ **Le programme de pompes de Noé** — Noé s'entraîne pendant 20 jours : le jour k, il fait $3 \times k$ pompes (3 le jour 1, 6 le jour 2, etc.). Pour calculer le nombre total de pompes, il écrit ce programme.

```
for k in range(1, 20):
    total = 0
    total = total + 3 * k
print(total)
```

(a) Qu'affiche le programme de Noé ? Explique pourquoi.

(b) Repère les deux erreurs de Noé et dis ce que chacune provoque.

(c) Écris le programme corrigé.

(d) Qu'affiche le programme corrigé ? Combien de pompes Noé fait-il en tout ?

Ex. 21 ★★☆ **Population d'une ville sur 10 ans** — Une ville compte 8 000 habitants en 2026, et sa population augmente de 4 % par an. Ben veut connaître la population en 2036 et compter les années, de 2027 à 2036, où elle dépasse 10 000 habitants.

```
pop = 8000
c = ...
for annee in range(..., ...):
    pop = ...
    if pop > 10000:
        c = ...
print(round(pop))
print(c)
```

(a) Par quel nombre la population est-elle multipliée chaque année ? Justifie.

(b) Complète tous les « ... » du programme, sachant que annee doit prendre les valeurs 1 à 10 (une valeur par année, de 2027 à 2036).

(c) Qu'affiche le programme complété ? Explique la valeur de c.

(d) Ben modifie une seule ligne pour compter les années où la population dépasse 9 000 habitants. Quelle ligne change-t-il, et que vaut alors c ?

Ex. 22 ★★★ **Les tickets gagnants de la tombola** — Pour la tombola du lycée, les tickets sont numérotés de 1 à 200. Un ticket gagne si son numéro est un multiple de 3 mais pas un multiple de 5. Lina veut compter les tickets gagnants avec Python.

(a) Traduis « k est un multiple de 3 mais pas de 5 » par une condition Python qui utilise `%`, `==`, `!=` et `and`.

(b) Écris un programme qui utilise une boucle `for` et un compteur pour afficher le nombre de tickets gagnants.

(c) Lina écrit `for k in range(1, 200):`. Elle dit : « Mon programme affiche le même nombre, donc il est juste. » A-t-elle raison ? Justifie.

(d) Sans exécuter de programme, retrouve par un calcul le nombre de tickets gagnants. Pour cela, compte les multiples de 3, puis les multiples de 15, entre 1 et 200.

(e) On remplace la boucle par `for k in range(3, 201, 3):`. Quelle condition reste-t-il à tester dans le `if` ? Combien de tours fait cette boucle ?

Ex. 23 ★★★ **La carte de cinéma à tarif dégressif** — Avec la carte d'un cinéma, les places n° 1 à 5 coûtent 8 €, les places n° 6 à 10 coûtent 6 € et les suivantes 4 €. Ben écrit une fonction qui renvoie le prix total de n places.

```
def prix_total(n):
    S = ...
    for k in range(...):
        if k <= 5:
            S = S + 8
        elif ...:
            S = S + 6
        else:
            S = S + 4
    return S
```

(a) Sans programme, calcule le prix total de 12 places.

(b) Complète la fonction de Ben et écris-la en entier.

(c) Dans la fonction de Ben complétée, Léo remplace seulement `if k <= 5:` par `if k < 5:`. Que renvoie alors `prix_total(12)` ? Explique l'écart.

(d) Dans la fonction de Ben complétée, Léo remplace seulement la ligne du `for` par `for k in range(n):`. Que renvoie `prix_total(12)` ? Explique.

(e) Avec un budget de 100 €, combien de places peut-on acheter au maximum ? Justifie par un calcul.

Ex. 24 ★★★ **L'élevage de lapins** — Un éleveur modélise le nombre de couples de lapins mois après mois. Chaque mois, le nouveau nombre de couples est la somme des nombres des deux mois précédents. Ben écrit ce programme.

```
a = 1
b = 1
for k in range(6):
    c = a + b
    a = b
    b = c
print(b)
```

(a) Combien de tours fait la boucle ? Quelle est la dernière valeur prise par k ?

(b) Complète la trace d'exécution, puis donne ce qu'affiche le programme.

Tour	k	c	a	b
Avant	—	—	1	1
1	0	?	?	?
2	1	?	?	?
3	2	?	?	?
4	3	?	?	?
5	4	?	?	?
6	5	?	?	?

(c) Modifie le programme de Ben pour qu'il affiche aussi la somme des valeurs de b obtenues à la fin des 6 tours. Écris-le et donne cette somme.

(d) Un élève supprime c et écrit dans la boucle `a = b` puis `b = a + b`. Qu'affiche son programme ? Explique pourquoi le résultat change.

(e) Dans le programme de Ben, par quel nombre faut-il remplacer 6 dans `range(6)` pour que le programme affiche 144 ? Justifie.

4. *Boucle non bornée while*

À retenir : Le `while` contient la condition pour CONTINUER, contraire de l'arrêt : arrêt dès que $v < 50$ → `while v >= 50:`. Le bloc modifie la variable testée ; le compteur part de 0 et augmente de 1. Jamais `!=` avec des flottants.

Ex. 25 ★☆☆ **Automatismes : écrire la bonne condition** — Réponds en une ou deux lignes et justifie.

(a) On veut s'arrêter dès que la somme `s` dépasse 500. Écris la ligne `while`.

(b) Une température `t` baisse. On veut s'arrêter dès que $t < 20$. Écris la ligne `while`.

(c) Pour s'arrêter dès que le `score` atteint 100, un élève écrit `while score <= 100:`. A-t-il raison ? Justifie.

(d) Pour trouver au bout de combien d'années une épargne double, choisis-tu une boucle `for` ou `while` ? Justifie.

Ex. 25 (suite)

(e) Juste avant `while n > 0:`, la variable `n` vaut 0. Combien de fois le bloc est-il exécuté ?

(f) Après `x = 0`, on écrit `while x != 1: x = x + 0.1`. Le programme s'arrête-t-il ? Justifie.

Ex. 26 ★☆☆ **Automatismes : nombre de tours et valeur à la sortie** — Chaque boucle est écrite sur une seule ligne. Réponds sans ordinateur.

(a) Après `a = 3`, on exécute `while a < 50: a = a * 2`. Quelle est la valeur de `a` à la sortie ?

(b) Après `n = 100`, on exécute `while n >= 30: n = n - 25`. Combien de tours fait la boucle ?

(c) Après `m = 13`, on exécute `while m % 4 != 0: m = m + 1`. Quelle est la valeur de `m` à la sortie ?

(d) Après `c = 0`, on exécute `while c != 7: c = c + 2`. Que se passe-t-il ?

(e) Après `t = 3`, on exécute `while t < 10: print(t)`. Que se passe-t-il ?

(f) Après `p = 1`, on exécute `while p <= 1000: p = p * 10`. Combien de tours fait la boucle ? Et si on remplace `<=` par `<` ?

Ex. 27 ★★☆ **Épargne : trace d'un seuil** — Ben place 2 500 € sur un livret qui rapporte 4 % par an. Il cherche au bout de combien d'années son capital atteint 3 000 €.

```
capital = 2500
annees = 0
while capital < 3000:
    capital = capital * 1.04
    annees = annees + 1
print(annees)
```

(a) Explique pourquoi la ligne 4 multiplie `capital` par 1.04.

(b) Recopie et complète la trace (montants arrondis au centime ; la colonne « Test » donne la valeur de `capital < 3000` après l'étape).

Étape	capital	annees	Test
Début	2500	0	Vrai
Tour 1	2600	1	Vrai
Tour 2	?	?	?
Tour 3	?	?	?
Tour 4	?	?	?
Tour 5	?	?	?

(c) Qu'affiche le programme ? Que vaut alors `capital`, arrondi au centime ?

(d) Ben place finalement 2 800 € : la ligne 1 devient `capital = 2800`. Qu'affiche le programme ? Vérifie par deux calculs.

(e) Avec 2 500 € au départ, pour savoir quand le capital dépasse strictement 3 000 €, quelle ligne `while` faut-il écrire ? L'affichage change-t-il ?

Ex. 28 ★★☆ **Voiture d'occasion : programme à compléter** — Une voiture achetée 18 000 € perd 15 % de sa valeur chaque année. On cherche au bout de combien d'années elle vaut moins de 6 000 €.

```
valeur = 18000
annees = ...
while ...:
    valeur = ...
    annees = ...
print(annees)
```

(a) Par quel nombre faut-il multiplier la valeur chaque année ? Justifie.

(b) Écris en français la condition d'arrêt, puis la ligne `while` qui convient.

(c) Recopie et complète le programme.

(d) Qu'affiche le programme ? Vérifie par deux calculs.

(e) Un élève écrit `annees = 1` à la ligne 2. Qu'affiche son programme ? Explique son erreur.

Ex. 29 ★★★ **Jeu de dés : attendre le 6** — On lance un dé équilibré jusqu'à obtenir un 6. Un élève écrit cette fonction, qui doit renvoyer le nombre de lancers effectués.

```
from random import randint

def attente_six():
    de = randint(1, 6)
    lancers = 1
    while de != 6:
        lancers = lancers + 1
    return lancers
```

(a) Le premier lancer donne 4. Que se passe-t-il ensuite ? Explique.

(b) Corrige la fonction. Explique pourquoi `lancers` vaut 1 avant la boucle.

(c) Ici, le test `de != 6` convient. Pourtant, avec `x = 0`, la boucle `while x != 1: x = x + 0.1` ne s'arrête jamais. Explique la différence.

(d) Écris un programme qui appelle la fonction corrigée 10 000 fois et affiche la moyenne des nombres de lancers. Que constates-tu en le relançant ?

Ex. 30 ★★★ **Tasse de thé : corriger un programme de seuil** — Une tasse de thé sort à 85 °C. On admet que sa température baisse de 6 % chaque minute. Lina cherche au bout de combien de minutes le thé passe sous 40 °C. Elle écrit ce programme.

```
temp = 85
minutes = 1
while temp < 40:
    minutes = minutes + 1
print(minutes)
```

(a) Qu'affiche ce programme ? Explique en suivant son exécution.

(b) Lina inverse la condition de la ligne 3, sans rien changer d'autre. Quelle ligne écrit-elle ? Que se passe-t-il alors ? Justifie.

Ex. 30 (suite)

(c) Écris le programme corrigé complet. Par quel nombre faut-il multiplier `temp` à chaque tour ? Justifie.

(d) Qu'afficherait le programme corrigé si l'on gardait la ligne 2 de Lina ? Explique l'écart.

(e) Qu'affiche le programme corrigé ? Justifie ce résultat par deux calculs de température.

Ex. 31 ★★★ **Course à pied : un objectif jamais atteint** — Ben prépare une course. La première semaine, il court 4 km, puis il ajoute 3 km chaque semaine. Il veut savoir au bout de combien de semaines il court 20 km. Il écrit ce programme.

```
distance = 4
semaines = 0
while distance != 20:
    distance = distance + 3
    semaines = semaines + 1
print(semaines, distance)
```

(a) Complète la trace d'exécution, puis explique pourquoi ce programme ne s'arrête jamais.

Tour	distance	semaines	Condition
Début	4	0	True
1	?	?	?
2	?	?	?
3	?	?	?
4	?	?	?
5	?	?	?
6	?	?	?

(b) Corrige la ligne 3 pour que la boucle s'arrête dès que la distance atteint ou dépasse 20 km. Qu'affiche alors le programme ?

(c) Lina affirme : « Pour "atteindre 19 km" ou "dépasser 19 km", le programme affiche le même nombre de semaines. » Vrai ou faux ? Justifie avec les deux conditions du while.

(d) Écris une fonction `nb_semaines(objectif)` qui renvoie le nombre de semaines d'ajouts nécessaires pour que la distance atteigne au moins `objectif` km. Que renvoie `nb_semaines(42)` ? Justifie.

(e) Ben veut maintenant sa distance après 8 semaines d'ajouts. Faut-il une boucle while ou une boucle for ? Écris le programme et donne son affichage.

Ex. 32 ★★★ **Deux villes : quand A dépasse-t-elle B ?** — La ville A compte 12 000 habitants et gagne 3 % par an. La ville B compte 18 000 habitants et en perd 2 % par an. On modélise leurs populations après n années par ces fonctions.

```
def ville_a(n):
    return 12000 * 1.03 ** n

def ville_b(n):
    return 18000 * 0.98 ** n
```

(a) Calcule à la main les valeurs renvoyées par `ville_a(1)` et `ville_b(1)`. Explique les nombres 1.03 et 0.98.

(b) Écris une fonction `depassement()`, sans paramètre, qui appelle `ville_a` et `ville_b` et renvoie le premier nombre d'années où A est plus peuplée que B.

(c) Quelle valeur renvoie `depassement()` ? Justifie par deux calculs de populations (arrondies à l'unité).

(d) Un élève propose de tester l'égalité des deux populations avec `!=` dans la condition du `while`. Explique pourquoi c'est une erreur grave.

(e) Un autre élève termine sa fonction par `print(...)` au lieu de `return ...`, puis exécute `x = depassement()` et `print(x)`. Qu'affiche son programme ? Explique.

5. *Fonctions*

À retenir : `def nom(paramètres):` puis un bloc indenté terminé par `return`, qui renvoie la valeur et arrête la fonction. On donne les arguments dans l'ordre des paramètres. Avec print à la place de return, la fonction renvoie None. $f(x) = 2x^2$ s'écrit `return 2 * x ** 2`.

Ex. 33 ★☆☆ **Lire un appel de fonction** — Automatismes. On a défini les deux fonctions ci-dessous. Réponds sans ordinateur.

```
def g(x):
    return 5 - 2 * x

def h(a, b):
    return a * 10 + b
```

(a) Que renvoie l'appel `g(4)` ?

(b) Que renvoie l'appel `g(-1)` ?

(c) Que renvoie l'appel `h(3, 7)` ?

(d) Que renvoie `h(7, 3)` ? Compare avec la réponse précédente.

(e) Que renvoie `g(h(0, 2))` ?

(f) Quelle est la valeur de `y` après l'instruction `y = g(1) + h(1, 1)` ?

(g) Pourquoi l'appel `h(5)` provoque-t-il une erreur ?

Ex. 34 ★★☆ **Fonction mathématique et fonction Python** — Automatismes. Dans chaque fonction Python, le paramètre s'appelle x.

(a) Écris la ligne `return` qui traduit $f(x) = 4x^2 - 5$.

(b) Écris la ligne `return` qui traduit $g(x) = \frac{x+1}{2}$.

(c) Pour $h(x) = 3x - 1$, un élève écrit `return 3x - 1`. Corrige son erreur.

(d) Quelle fonction mathématique la ligne `return (x - 2) ** 2` traduit-elle ?

(e) Calcule de tête $f(3)$ pour vérifier la fonction f écrite plus haut. Que doit renvoyer `f(3)` ?

(f) Pour $k(x) = \frac{6}{x+1}$, un élève écrit `return 6 / x + 1`. Que renvoie son `k(2)` ? Et le bon `k(2)` ?

Ex. 35 ★★☆ **Tarif de cinéma : print ou return** — Un cinéma fait payer 8 € par place, plus 5 € de frais de réservation. Ben écrit deux versions d'une fonction qui calcule le prix pour n places.

```
def tarif_a(n):
    return 8 * n + 5
    print("Bon film")

def tarif_b(n):
    print(8 * n + 5)
```

(a) Ben exécute ces lignes seules, sans rien ajouter. Qu'affiche l'écran ? Justifie.

(b) Qu'affiche `print(tarif_a(3))` ? La ligne `print("Bon film")` est-elle exécutée ? Justifie.

(c) Qu'affiche l'instruction `tarif_b(3)` seule ? Et l'instruction `print(tarif_b(3))` ?

(d) Explique pourquoi `total = tarif_b(3) + 2` provoque une erreur, puis corrige tarif_b.

Ex. 36 ★★☆ **Admis ou non : compléter un programme** — Un élève est admis si sa note est supérieure ou égale à 10. La fonction est_admis renvoie un booléen et la fonction resultat renvoie un texte. Complète le programme.

```
def est_admis(note):
    return ...

def resultat(note):
    if ...:
        return "admis"
    ...
        return "non admis"
```

(a) Complète la ligne 2 : `est_admis(note)` doit renvoyer True si la note est supérieure ou égale à 10, et False sinon.

(b) Complète les lignes 5 et 7. La fonction resultat doit appeler est_admis.

(c) Que renvoient `est_admis(9.5)` et `resultat(7.5 + 2.5)` ?

(d) Ben écrit le corps de est_admis ainsi : `if note >= 10:` puis `return True`, puis hors du if `return False`, sans else. A-t-il raison ? Justifie.

Ex. 37 ★★☆ **Vitesse moyenne d'un coureur** — La vitesse moyenne d'un coureur (en km/h) est égale à la distance parcourue (en km) divisée par la durée de la course (en h). On veut une fonction qui marche pour toutes les courses.

(a) Écris une fonction `vitesse(distance, duree)` qui renvoie la vitesse moyenne. Teste-la sur un cas que tu calcules de tête.

(b) Un trail de 15 km est couru en 1,25 h. Que renvoie `vitesse(15, 1.25)` ?

(c) Un élève tape `vitesse(1.25, 15)`. Que renvoie cet appel (arrondi au centième) ? Pourquoi ce résultat n'a-t-il aucun sens ?

(d) Écris une fonction `temps_en_heures(h, m)` qui convertit h heures et m minutes en heures. Que renvoie `vitesse(25, temps_en_heures(2, 30))` ?

Ex. 38 ★★★ **Location de kayaks** — Une base de loisirs loue des kayaks. Le prix par personne est de 12 € pour 2 heures ou moins, puis de 5 € par heure au-delà de 2 heures. Pour un groupe de 4 personnes ou plus, le prix total baisse de 10 €. Le nombre d'heures est un entier.

(a) Écris une fonction `prix_kayak(heures)` qui renvoie le prix pour une personne. Utilise un if et deux return. Vérifie à la main `prix_kayak(2)` et `prix_kayak(5)`.

(b) Écris une fonction `prix_groupe(heures, personnes)` qui appelle `prix_kayak` et renvoie le prix total du groupe. Calcule `prix_groupe(3, 4)`.

(c) Pour 4 personnes pendant 3 heures, Léo tape `prix_groupe(4, 3)`. Quelle valeur obtient-il ? Explique son erreur.

(d) Ben tape `print(prix_groupe(3))`. Que se passe-t-il ? Justifie.

Ex. 39 ★★★ **Températures : corriger le programme de Noé** — Pour convertir une température de degrés Celsius (°C) en degrés Fahrenheit (°F), on calcule $F = 1{,}8 \times C + 32$. Noé écrit le programme ci-dessous. La fonction ecart doit renvoyer la différence, en °F, entre deux températures données en °C.

```
def fahrenheit(c):
    f = 1.8 * (c + 32)
    print(f)

def ecart(c1, c2):
    return fahrenheit(c2) - fahrenheit(c1)
```

(a) Noé exécute `fahrenheit(0)`. L'écran affiche 57.6, alors que 0 °C correspond à 32 °F. Explique l'erreur de la ligne 2, puis corrige-la.

(b) Une fois la ligne 2 corrigée, Noé exécute `print(fahrenheit(25))`. Qu'affiche l'écran ? Justifie chaque ligne affichée.

(c) Noé exécute ensuite `print(ecart(10, 30))` et obtient une erreur. Explique pourquoi, puis corrige la fonction fahrenheit.

Ex. 39 (suite)

(d) Avec ta correction, que renvoie `ecart(10, 30)` ? Vérifie le résultat par un calcul à la main.

(e) Ben ajoute la ligne `print("Conversion faite")` à la fin du corps de la fonction corrigée. Il dit que `y = fahrenheit(-40)` affichera Conversion faite. Vrai ou faux ? Justifie.

Ex. 40 ★★★ **Années bissextiles** — Une année est bissextile si elle est divisible par 400, ou si elle est divisible par 4 sans l'être par 100. Complète le programme ci-dessous.

```
def divisible(n, d):
    return ...

def est_bissextile(annee):
    if divisible(annee, 400):
        return True
    elif divisible(annee, 100):
        return ...
    else:
        return ...
```

(a) Complète la ligne 2 : `divisible(n, d)` doit renvoyer True si d divise n, et False sinon. Que renvoie `divisible(2026, 4)` ?

(b) Complète les lignes 8 et 10. Que renvoient `est_bissextile(1900)`, `est_bissextile(2000)` et `est_bissextile(2028)` ? Justifie.

(c) Écris une fonction `nb_bissextiles(debut, fin)` qui utilise un compteur et renvoie le nombre d'années bissextiles de debut à fin, fin comprise. Calcule `nb_bissextiles(1901, 2000)`.

(d) Léa écrit `range(debut, fin)` dans sa boucle. Que renvoie alors `nb_bissextiles(1901, 2000)` ? Explique l'écart.

(e) Ben copie les trois fonctions correctes dans un fichier et l'exécute sans rien ajouter. L'écran reste vide et il en conclut que ses fonctions sont fausses. A-t-il raison ? Justifie.

6. *Simulation avec random*

À retenir : `randint(1, 6)` inclut 6, alors que `range(1, 6)` s'arrête à 5 ; `random() < p` est vrai avec la probabilité p. Fréquence $= \frac{c}{n}$ (c compteur) : elle fluctue, mais pour n grand elle approche la probabilité, ex. $\frac{1}{6} \approx 0{,}167$.

Ex. 41 ★☆☆ **Lire les instructions du module random** — Automatismes. Réponds sans exécuter de programme.

(a) Écris la ligne à placer au début d'un programme pour pouvoir utiliser `randint` et `random`.

(b) Donne toutes les valeurs que peut renvoyer `randint(3, 7)`.

(c) Donne toutes les valeurs prises par `k` dans la boucle `for k in range(3, 7):`.

(d) Vrai ou faux : « `random()` peut renvoyer 1. » Justifie.

Ex. 41 **(suite)**

(e) Donne le type de la valeur renvoyée par `random()`, puis celui de la valeur renvoyée par `randint(3, 7)`.

(f) Avec quelle probabilité la condition `random() >= 0.35` est-elle vraie ?

Ex. 42 ★☆☆ **Fréquence, moyenne et fluctuation** — Automatismes. Lina simule 300 lancers d'un dé équilibré : le compteur des six vaut `c = 46`.

(a) Calcule la fréquence des six obtenue par Lina, arrondie au millième.

(b) Calcule l'écart entre la probabilité d'obtenir 6 et cette fréquence, arrondi au millième.

(c) Vrai ou faux : « Avec ce résultat, Lina prouve que son dé est truqué. » Justifie.

(d) Lina relance sa simulation de 300 lancers. Vrai ou faux : « Elle obtiendra encore 46 six. » Justifie.

(e) Dans un jeu en ligne, une fonction `points()` est appelée 400 fois et la somme des résultats vaut `s = 1090`. Calcule la moyenne de cette série.

Ex. 43 ★★☆ **Lancers francs au basket** — Au basket, Yanis réussit un lancer franc avec la probabilité 0,72. Ben commence un programme qui simule un lancer franc (1 s'il est réussi, 0 sinon), puis la fréquence de réussite sur `n` lancers.

```
from random import random

def lancer_franc():
    if random() < ...:
        return 1
    return 0

def frequence(n):
    c = ...
    for i in range(...):
        c = c + lancer_franc()
    return ...
```

(a) Complète la condition de la fonction `lancer_franc` pour que le lancer soit réussi avec la probabilité 0,72.

(b) Complète la fonction `frequence` pour qu'elle renvoie la fréquence des réussites sur `n` lancers.

(c) Un élève termine la fonction par `return c`. Que renvoie alors `frequence(500)` ? Est-ce une fréquence ?

(d) Yanis obtient 0.55 avec `frequence(20)`, puis 0.7183 avec `frequence(10000)`. Quelle valeur estime le mieux la probabilité 0,72 ? Justifie.

Ex. 44 ★★☆ **La somme de deux dés** — Dans un jeu de société, on lance deux dés et on gagne un bonus si la somme vaut 7. Noé veut estimer la probabilité du bonus et écrit ce programme.

```
from random import randint

def somme_des():
    return randint(2, 12)

def frequence_sept(n):
    c = 0
    for i in range(n):
        if somme_des() == 7:
            c = c + 1
    return c / n
```

(a) Avec ce programme, quelle est la probabilité que `somme_des()` renvoie 7 ?

(b) Avec deux vrais dés, combien des 36 couples (dé 1 ; dé 2) donnent une somme de 7 ? Déduis-en la probabilité du bonus.

(c) Explique l'erreur de Noé et corrige la fonction `somme_des`.

(d) Pour n grand, de quelle valeur `frequence_sept(n)` est-elle proche avant correction ? Et après correction ?

Ex. 45 ★★☆ **Le stand de la kermesse** — À la kermesse, on lance un dé : on gagne 2 points si on obtient un nombre pair, sinon on perd 3 points. Jade simule une partie ainsi.

```
from random import randint

def partie():
    d = randint(1, 6)
    if d % 2 == 0:
        print(2)
    else:
        print(-3)
```

(a) Calcule la probabilité de gagner 2 points lors d'une partie.

(b) Que vaut `5 % 2` ? Explique pourquoi `d % 2 == 0` teste si le dé donne un nombre pair.

(c) Dans une boucle, Jade écrit `s = s + partie()` et Python affiche une erreur. Explique pourquoi, puis corrige la fonction `partie`.

(d) Écris une fonction `moyenne(n)` qui renvoie la moyenne des points obtenus sur `n` parties.

(e) Sur 6 parties où chaque face sortirait une fois, calcule le total des points. De quelle valeur `moyenne(n)` est-elle proche pour n grand ? Explique si le jeu est favorable au joueur.

Ex. 46 ★★★ **Sortir de prison au jeu de plateau** — Dans un jeu de plateau, le pion de Malik sort de prison s'il obtient 5 ou 6 avec un dé équilibré. Malik veut estimer la probabilité de sortir en un lancer. Il écrit ce programme.

```
from random import randint

def lancer():
    return randint(1, 6)

def frequence_sortie(n):
    c = 0
    for i in range(1, n):
        if lancer() == 5 or lancer() == 6:
            c = c + 1
    return c
```

(a) Calcule la probabilité de sortir de prison en un lancer. Donne la valeur exacte, puis une valeur arrondie au millième.

(b) Combien de tours fait la boucle lors de l'appel `frequence_sortie(500)` ? Explique pourquoi la valeur renvoyée n'est pas une fréquence, puis corrige ces deux défauts.

(c) Explique pourquoi la condition `lancer() == 5 or lancer() == 6` ne simule pas un seul lancer. Écris le programme entièrement corrigé.

(d) Dans la condition de Malik, on note a le résultat du 1er appel à `lancer()` et b celui du 2e. Parmi les 36 couples $(a\,;b)$, combien font compter un tour ? Déduis-en la probabilité qu'un tour soit compté.

(e) Une fois le programme corrigé, `frequence_sortie(500)` renvoie 0.342. Malik en conclut que son dé est truqué. A-t-il raison ? Justifie.

Ex. 47 ★★★ **Les sachets de graines de tomate** — Ben sème des graines de tomate par sachets de 5. Chaque graine germe avec la probabilité 0,8, indépendamment des autres. Il veut estimer par simulation le nombre moyen de graines qui germent dans un sachet. Il utilise la fonction random du module random.

(a) On rappelle que `random()` renvoie un flottant de $[0\,;1[$. Explique pourquoi le test `random() < 0.8` est vrai avec la probabilité 0,8.

(b) Écris la ligne d'import, puis une fonction `germe()` qui renvoie 1 si une graine germe et 0 sinon.

(c) Écris une fonction `semis()` qui appelle `germe()` pour les 5 graines d'un sachet et renvoie le nombre de graines qui ont germé.

(d) Écris une fonction `moyenne(n)` qui simule n sachets et renvoie la moyenne du nombre de graines germées par sachet.

(e) Ben exécute deux fois `moyenne(10000)` et obtient 3.9952, puis 4.0031. Explique pourquoi ces valeurs diffèrent et justifie la valeur dont elles s'approchent.

Ex. 48 ★★★ **En attendant le six aux petits chevaux** — Aux petits chevaux, Inès doit obtenir un 6 pour sortir un cheval. La fonction attente renvoie le nombre de lancers d'un dé équilibré qu'il faut pour obtenir le premier 6.

```
from random import randint

def attente():
    nb = 1
    while randint(1, 6) != 6:
        nb = nb + 1
    return nb
```

(a) Les tirages successifs sont 2, 5, puis 6. Complète la trace et donne la valeur renvoyée.

Étape	tirage	tirage != 6	nb
Début	aucun	aucun	1
Test 1	2	?	?
Test 2	5	?	?
Test 3	6	?	?

(b) Léo recopie le programme en écrivant `import random` en 1re ligne et `randint(1, 7)`, « car 7 est exclu comme dans range ». Explique ses deux erreurs.

(c) Inès écrit cette fonction pour calculer le nombre moyen de lancers sur n attentes. Explique son erreur, puis corrige la fonction.

```
def moyenne_attente(n):
    s = 0
    for i in range(n):
        s = attente()
    return s / n
```

(d) Ben obtient 5.97, puis 6.04 avec `moyenne_attente(10000)`. Explique cet écart et relie ces résultats à la probabilité $\frac{1}{6}$ d'obtenir un 6.

(e) Vrai ou faux ? « La condition du while peut rester vraie, donc attente() est une boucle infinie. » Justifie, puis donne une modification qui la rendrait réellement infinie.