

Plaquette 3 – Écrire une fonction : calculs techniques et tests

Algorithmique et Python — Première STI2D

Corrections détaillées

Boîte à outils

Écrire une fonction.

```
def nom(arguments):
    ...
    return valeur
```

- Les **arguments** sont les données d'entrée, la valeur du *return* est le résultat.
- *return* **arrête** la fonction : ce qui suit n'est pas exécuté.
- *print* **affiche** mais ne renvoie rien : une fonction qui se termine par *print* renvoie *None* et ne peut pas être réutilisée dans un calcul.

Tester dans une fonction.

```
if condition:
    ...
elif autre_condition:
    ...
else:
    ...
```

Le *elif* n'est examiné que si le *if* est faux : l'ordre des conditions compte.

Formules techniques utilisées dans cette plaquette.

- Puissance électrique (en W, avec *U* en V et *I* en A) :

$$P = U \times I$$

- Rendement : $\eta = \frac{P_{\text{utile}}}{P_{\text{absorbée}}}$, toujours compris entre 0 et 1.

- Débit : $Q = \frac{V}{t}$.

- Taux d'évolution : $t = \frac{V_{\text{finale}} - V_{\text{initiale}}}{V_{\text{initiale}}}$.

- Tolérance : la cote *x* est conforme si $|x - c| \leq \varepsilon$, où *c* est la cote visée.

- Deux résistances en parallèle : $\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$.

Arrondir : *round(x, 2)* donne la valeur arrondie à deux décimales.

Correction de l'exercice 1 - Une fonction de conversion

On traduit directement la formule $F = \frac{9}{5}C + 32$: la fonction reçoit la température *c* en degrés Celsius et **renvoie** sa conversion.

```
def fahrenheit(c):
    return 9 / 5 * c + 32
```

Pour $C = 20$: $\frac{9}{5} \times 20 = 36$, puis $36 + 32 = 68$ °F. Pour $C = 37$: $\frac{9}{5} \times 37 = 66,6$, puis $66,6 + 32 = 98,6$ °F.

$$F(20) = 68 \quad F(37) = 98,6$$

Contrôle de bon sens : 0 °C donne 32 °F, valeur connue du point de congélation de l'eau.

Réponse. *fahrenheit(20)* = 68 et *fahrenheit(37)* = 98,6.

Correction de l'exercice 2 - Puissance électrique

La puissance électrique est le produit de la tension par l'intensité :

$$P = U \times I$$

```
def puissance(u, i):
    return u * i
```

Avec $U = 230$ V et $I = 1,5$ A, *puissance(230, 1.5)* renvoie $230 \times 1,5 = 345$ W. Les unités sont cohérentes : des volts multipliés par des ampères donnent des watts. C'est l'ordre de grandeur d'un gros outil portatif. La fonction **renvoie** le résultat avec *return*, ce qui permet de le réutiliser dans un autre calcul.

Réponse. *puissance(230, 1.5)* renvoie 345 W.

Correction de l'exercice 3 - Contrôle d'une cote

Une cote est conforme lorsque son écart à 25 mm, en plus ou en moins, ne dépasse pas 0,4 mm. La valeur absolue traite les deux sens de la même façon :

$$|x - 25| \leq 0,4 \Leftrightarrow 24,6 \leq x \leq 25,4$$

```
def conforme(x):
    return abs(x - 25) <= 0.4
```

Pour $x = 25,3$: l'écart vaut $0,3 \leq 0,4$, la fonction renvoie *True*, la pièce est conforme. Pour $x = 24,5$: l'écart vaut $0,5 > 0,4$, elle renvoie *False*, la pièce est refusée.

Réponse. 25,3 est conforme, 24,5 ne l'est pas (intervalle [24,6 ; 25,4]).

Correction de l'exercice 4 - Trois classes de rendement

Les conditions sont testées **dans l'ordre**, et la fonction s'arrête au premier *return* atteint.

```
def classe(r):
    if r >= 90:
        return "A"
    elif r >= 75:
        return "B"
    else:
        return "C"
```

Pour $r = 92$, le premier test est vrai : "A". Pour $r = 80$, $80 \geq 90$ est faux, on passe au *elif*, et $80 \geq 75$ est vrai : "B". Pour $r = 60$, les deux tests sont faux : "C".

L'ordre compte : en testant d'abord $r \geq 75$, le rendement 92 serait classé "B". Les cas limites suivent les inégalités larges : 90 donne "A" et 75 donne "B".

Réponse. $92 \rightarrow$ "A", $80 \rightarrow$ "B", $60 \rightarrow$ "C".

Correction de l'exercice 5 - return ou print ?

print **affiche** une valeur à l'écran, mais ne la **renvoie** pas. Une fonction sans *return* renvoie *None*, c'est-à-dire « rien » : *double(5)* affiche 10 mais vaut *None*, et le calcul $2 * \text{None}$ n'a pas de sens, d'où l'erreur.

Correction : on remplace l’affichage par un renvoi.

```
def double(x):
    return 2 * x
```

Désormais $\text{double}(5)$ vaut 10 et $2 * \text{double}(5)$ vaut 20. Une fonction destinée à être réutilisée dans un calcul se termine par *return* ; *print* sert seulement à afficher, par exemple un tableau de valeurs.

Réponse. Avec *print*, la fonction renvoie *None* : il faut écrire *return 2 * x*.

Correction de l’exercice 6 - Un tarif par paliers

Le second tarif ne s’applique qu’aux pièces **au-delà** de 100 : on sépare les deux tranches.

```
def cout(n):
    if n <= 100:
        return 12 * n
    else:
        return 12 * 100 + 10 * (n - 100)
```

Pour $n = 60$, on reste dans la première tranche : $12 \times 60 = 720$ €. Pour $n = 150$, les 100 premières pièces coûtent 1 200 € et les 50 suivantes $10 \times 50 = 500$ € :

$$\text{cout}(150) = 1\,200 + 500 = 1\,700 \text{ €}$$

Appliquer 10 € à toutes les pièces donnerait 1 500 €, ce qui est faux. Le prix moyen réel vaut $\frac{1\,700}{150} \approx 11,33$ € la pièce.

Réponse. $\text{cout}(60) = 720$ € · $\text{cout}(150) = 1\,700$ €.

Correction de l’exercice 7 - Une fonction qui contient une boucle

On accumule les entiers de 1 à n dans une variable s initialisée à 0 **avant** la boucle, et on la renvoie **après** la boucle. Pour que n soit inclus, on écrit *range(1, n + 1)*.

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

Pour $n = 10$, la fonction calcule $1 + 2 + \dots + 10$. **Contrôle** en regroupant les termes deux à deux depuis les extrémités :

$$(1 + 10) + (2 + 9) + (3 + 8) + (4 + 7) + (5 + 6) = 5 \times 11 = 55$$

Avec *range(1, n)*, on oublierait 10 et on trouverait 45.

Réponse. *somme(10)* renvoie 55.

Correction de l’exercice 8 - Calculer un terme de suite

Pour passer de u_0 à u_n , il faut appliquer n fois la relation de récurrence : la boucle fait n tours, en partant de $u = 5$.

```
def terme(n):
    u = 5
    for i in range(n):
        u = 0.8 * u + 3
    return u
```

Pour $n = 3$: $u_1 = 0,8 \times 5 + 3 = 7$, $u_2 = 0,8 \times 7 + 3 = 8,6$, puis

$$u_3 = 0,8 \times 8,6 + 3 = 9,88 \text{ m}^3$$

Le volume augmente de moins en moins vite. La valeur 15 est **stable** : $0,8 \times 15 + 3 = 15$, si bien qu'une cuve contenant 15 m^3 n'évoluerait plus ; en calculant d'autres termes avec la fonction, on constate que le volume s'en rapproche.

Réponse. $u_3 = 9,88 \text{ m}^3$.

Correction de l'exercice 9 - Taux d'évolution

Le taux d'évolution est la variation divisée par la valeur de départ :

$$t = \frac{V_{\text{finale}} - V_{\text{initiale}}}{V_{\text{initiale}}}$$

```
def taux(vi, vf):
    return (vf - vi) / vi * 100
```

La production gagne $1\,400 - 1\,250 = 150$ pièces, et $\frac{150}{1\,250} = 0,12$: la fonction renvoie 12, soit une hausse de 12 %.

Les parenthèses sont indispensables : sans elles, Python calculerait d'abord vi / vi .

Vérification par le coefficient multiplicateur : $1\,250 \times 1,12 = 1\,400$.

Réponse. `taux(1250, 1400)` renvoie 12, soit une hausse de 12 %.

Correction de l'exercice 10 - Prix hors taxes

Ajouter 20 % de TVA revient à multiplier par le coefficient $1 + \frac{20}{100} = 1,2$. L'opération inverse consiste à **diviser** par 1,2.

```
def ttc(ht):
    return ht * 1.2

def ht(prix_ttc):
    return prix_ttc / 1.2
```

$$\text{ht}(96) = \frac{96}{1,2} = 80 \text{ €}$$

Vérification : $80 \times 1,2 = 96$. Calculer $96 \times 0,8 = 76,80 \text{ €}$ est faux : retirer 20 % de 96 n'annule pas une hausse de 20 % appliquée à 80, car les deux pourcentages ne portent pas sur la même base.

Réponse. 80 € hors taxes (et non 76,80 €).

Correction de l'exercice 11 - Deux résistances en parallèle

La formule donne l'inverse de R : on calcule la somme des inverses, puis on en prend l'inverse.

```
def equivalente(r1, r2):
    return 1 / (1 / r1 + 1 / r2)
```

Avec 100 et 150 Ω , $\frac{1}{100} + \frac{1}{150} = \frac{3}{300} + \frac{2}{300} = \frac{5}{300} = \frac{1}{60}$, donc

$$R = 60 \Omega$$

Les parenthèses au dénominateur sont indispensables : sans elles, Python renverrait $\frac{1}{r_1} + \frac{1}{r_2} \approx 0,0167$. **Contrôle** : la résistance équivalente est plus petite que chacune des deux, comme il se doit en parallèle ; deux résistances égales donneraient la moitié de

l'une.

Réponse. *equivalente(100, 150)* renvoie 60 Ω.

Correction de l'exercice 12 - Arrondir proprement

Le rendement est le quotient de la puissance utile par la puissance absorbée :

$$\eta = \frac{P_{\text{utile}}}{P_{\text{absorbée}}}$$

```
def rendement(pu, pa):
    return round(pu / pa * 100, 1)
```

$\frac{1\,800}{2\,400} = 0,75$, soit 75 en pourcentage ; arrondi au dixième, la fonction renvoie 75,0. Le second argument de *round* indique le nombre de décimales conservées.

Contrôle : un rendement est toujours inférieur à 100 %. Ici, $2\,400 - 1\,800 = 600$ W sont perdus, le plus souvent en chaleur.

Réponse. *rendement(1800, 2400)* renvoie 75,0, soit 75 %.

Correction de l'exercice 13 - Contrôler une liste de cotes

On reprend le test de conformité dans une boucle, avec un compteur initialisé à 0 avant la boucle.

```
def nb_conformes(L):
    c = 0
    for x in L:
        if abs(x - 25) <= 0.4:
            c = c + 1
    return c
```

Cote	25,1	24,5	25,4	25,0	24,55	25,6
Écart	0,1	0,5	0,4	0	0,45	0,6
Conforme	oui	non	oui	oui	non	non

La cote 25,4 a un écart exactement égal à 0,4 : l'inégalité large $\leq$ l'accepte. La fonction renvoie 3, soit une fréquence de

$$\frac{3}{6} = 0,5$$

Un lot à 50 % de conformité impose un réglage immédiat de la machine.

Réponse. *nb_conformes(L)* renvoie 3 (une cote sur deux est conforme).

Correction de l'exercice 14 - Corriger une fonction

Première erreur : *s* est initialisée à 1. La somme est alors majorée de 1 ; un accumulateur de somme part de 0.

Seconde erreur : le *return* est **dans** la boucle, indenté sous le *for*. La fonction s'arrête donc dès le premier tour. Il faut le sortir d'un niveau d'indentation.

```
def moyenne(L):
    s = 0
    for v in L:
        s = s + v
    return s / len(L)
```

Sur [14, 9, 21, 16], la version corrigée renvoie $\frac{60}{4} = 15$, alors que la version fautive renvoyait $\frac{1+14}{4} = 3,75$: elle n'avait additionné que la première valeur, plus l'initialisation erronée.

Réponse. *s* doit être initialisée à 0, et le *return* doit être **après** la boucle.

Correction de l'exercice 15 - Deux fonctions qui s'appellent

a) Le débit est le volume divisé par la durée, $Q = \frac{V}{t}$:

```
def debit(v, t):
    return v / t
```

debit(450, 12) renvoie 37,5 L/min.

b) Le temps de remplissage est le volume divisé par le débit, $t = \frac{V}{Q}$:

```
def duree(v, q):
    return v / q
```

c) On combine les deux appels, *duree(3000, debit(450, 12))* :

$$t = \frac{3\,000}{37,5} = 80 \text{ min}$$

soit 1 h 20 min. **Contrôle** par proportionnalité : 3 000 L représentent $\frac{3\,000}{450} \approx 6,67$ fois 450 L, et $6,67 \times 12 \approx 80$ min.

Réponse. a) 37,5 L/min · b) $t = \frac{V}{Q}$ · c) 80 min, soit 1 h 20.

Correction de l'exercice 16 - Synthèse : un module de contrôle énergétique

a) Le rendement est le quotient de la puissance utile par la puissance absorbée, en pourcentage : *rendement(1.8, 2.4)* renvoie $\frac{1,8}{2,4} \times 100 = 75$.

```
def rendement(pu, pa):
    return pu / pa * 100
```

```
def conso(p, h):
    return p * h
```

b) L'énergie est la puissance absorbée multipliée par la durée : *conso(2.4, 3)* renvoie 7,2 kWh par jour.

c) Sur l'année, $7,2 \times 250 = 1\,800$ kWh, d'où un coût de
 $1\,800 \times 0,18 = 324$ € par an

d) Le nouveau moteur absorbe $\frac{1,8}{0,85} \approx 2,118$ kW, soit environ 1 588 kWh par an et 286 € : l'économie est d'environ 38 € par an. L'amortissement prendrait $\frac{1\,900}{38} = 50$ ans : l'investissement n'est pas rentable pour cet usage, il ne le deviendrait qu'avec un fonctionnement beaucoup plus long.

Réponse. a) 75 % · b) 7,2 kWh/jour · c) 324 € par an · d) 38 € économisés par an, soit 50 ans : investissement non rentable.