

Python : premiers programmes

Tout ce dont vous avez besoin pour ce module, dans un seul document. Cochez au fur et à mesure.

U6-S1-M1

module

Élève

document

MaisonDesJeux

scénario 1

B · 3 h

format et durée

Ce qu'il faut savoir avant de commencer	3
C'est quoi l'extension d'un fichier ?	3
C'est quoi un terminal ?	3
Comment écrit-on un nombre à virgule ?	3
Vérifiez-vous	3
Variables, types et premières erreurs	5
Capsule 1 sur 4 Écrire et lancer un programme	5
Capsule 2 sur 4 Ranger des valeurs : variables et types	7
Capsule 3 sur 4 Calculer, comparer, mettre en forme	10
Capsule 4 sur 4 Lire un message d'erreur	12
Ce qu'on retient	15
Variables, types et premières erreurs	16
L'essentiel	16
Les mots à connaître	16
Les types	16
La méthode pour une erreur	17
Un exemple concret	17
Les pièges	18
Douze micro-exercices et trois pannes	19
PARTIE A — Démarrer	20
PARTIE B — Variables, calculs, conversions	23
PARTIE C — Comparer et mettre en forme	26
PARTIE D — Trois pannes	29
POUR ALLER PLUS LOIN	32
CE QU'IL FAUT RETENIR	33
Tous les documents du module	34
A1 — Le dossier de travail	34
A2 — Les douze énoncés	34
A3 — Extrait de la documentation de Python	38
A4 — Les trois programmes en panne	40

Ce qu'il faut savoir avant de commencer

Ce qu'il faut savoir avant de commencer. Lisez-le si vous avez un doute. Environ 10 minutes.

Aucun prérequis en programmation : on part de zéro. Trois rappels vous aideront à vous servir du poste.

C'est quoi l'extension d'un fichier ?

C'est la fin du nom du fichier, après le point. Elle dit quel genre de fichier c'est.

Exemple : `photo.jpg` est une image, `ex01.py` est un programme Python.

C'est quoi un terminal ?

Un terminal est une fenêtre où l'on tape des commandes au clavier. On tape la commande, puis on appuie sur Entrée. La réponse s'affiche juste en dessous.

Sous Windows, le terminal s'appelle PowerShell. VS Code en ouvre un en bas de sa fenêtre.

Comment écrit-on un nombre à virgule ?

En programmation, la virgule s'écrit avec un **point**. On écrit `12.5`, pas `12,5`.

La virgule sert à autre chose : séparer deux valeurs.

Vérifiez-vous

☐

1. Quelle est l'extension du fichier `tarifs.py` ?

☐

2. Vous avez tapé une commande dans le terminal. Sur quelle touche appuyez-vous pour la lancer ?

- ☐ 3. Comment écrit-on vingt-neuf euros quatre-vingt-dix dans un programme ?
-

Réponses

1. `.py` : c'est un programme Python.
2. Sur la touche Entrée.
3. `29.90` , avec un point.

Variables, types et premières erreurs

COURS-U6-S1-01

Léa Fournier est animatrice bénévole à MaisonDesJeux, une association de jeux de société. Chaque mercredi, elle calcule à la main les cotisations des nouveaux adhérents.

Bruno, un autre bénévole, lui a écrit un petit programme pour faire le calcul. Léa le lance. L'écran se remplit de lignes rouges en anglais, et le programme s'arrête.

Léa vous appelle : « Le programme de Bruno est cassé. Tu peux regarder ? »

À la fin de la séance, vous saurez écrire et lancer un petit programme en Python. Vous saurez aussi lire un message d'erreur pour trouver la ligne qui ne va pas.

LE PLAN DE LA SÉANCE

- 1 Écrire et lancer un programme.
- 2 Ranger des valeurs : variables et types.
- 3 Calculer, comparer, mettre en forme.
- 4 Lire un message d'erreur.

LA QUESTION DE DÉPART

Votre GPS vous dit : « Faites demi-tour. » Vous l'ignorez. Il ne plante pas : il recalcule. Le programme de Bruno, lui, s'arrête net. Pourquoi ?

CAPSULE 1 SUR 4

Écrire et lancer un programme

Avant de réparer le programme de Bruno, il faut savoir comment un programme marche. Que se passe-t-il quand on le lance ?

Un programme est une liste d'instructions, écrite dans un fichier texte. Python ne lit pas ce fichier tout seul : il a besoin d'un **INTERPRÉTEUR**. C'est le logiciel qui lit le programme et exécute ses instructions, une ligne après l'autre.

Pensez à une recette de cuisine. La recette ne fait rien toute seule. Il faut quelqu'un qui lit chaque étape, dans l'ordre, et qui la fait.

Pour écrire le programme, on utilise un **ENVIRONNEMENT DE DÉVELOPPEMENT**, qu'on appelle aussi IDE, pour *Integrated Development Environment*. C'est un éditeur de texte fait pour le code : il colore les mots, il signale les fautes, et il ouvre un terminal. Ici, on utilise VS Code.

Voici le chemin d'un programme :

- 1 **Écrire** : on tape les instructions dans un fichier `.py`, dans VS Code.
- 2 **Enregistrer** : Ctrl+S. Un fichier non enregistré n'est pas celui que Python lit.
- 3 **Lancer** : dans le terminal, on tape `python` suivi du nom du fichier.
- 4 **Lire** : le résultat, ou une erreur, s'affiche dans le terminal.

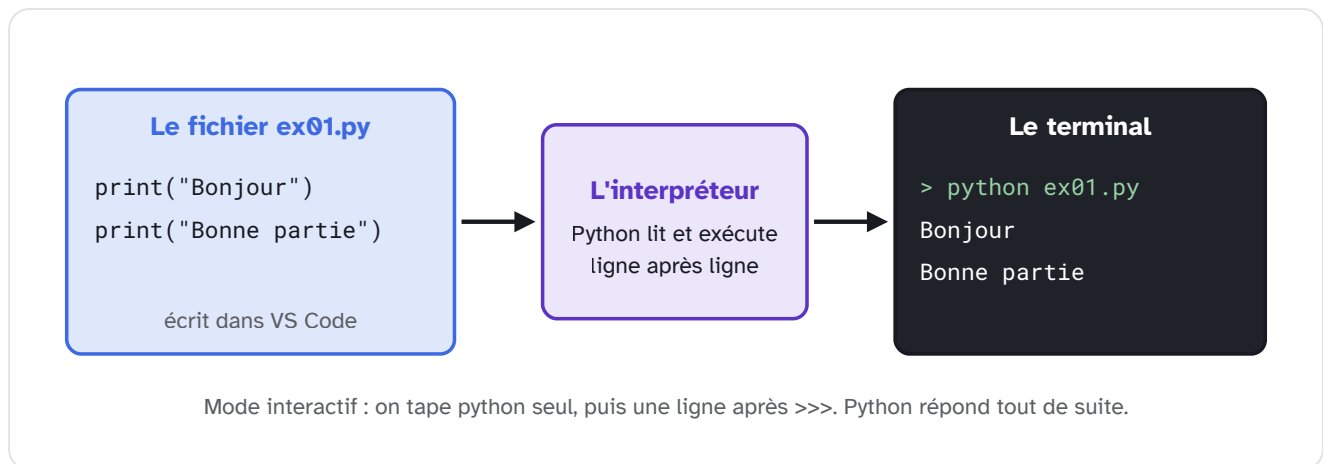


Schéma • Le programme est écrit dans un fichier, l'interpréteur l'exécute, le résultat s'affiche dans le terminal

Deux instructions servent dès le début. `print()` affiche à l'écran : c'est une **sortie**. `input()` pose une question et attend la réponse au clavier : c'est une **entrée**.

```
nom = input("Votre prénom : ")  
print("Bonjour", nom)
```

On peut aussi lancer Python seul, sans fichier : c'est le mode interactif. Python affiche `>>>`, on tape une ligne, il répond tout de suite.

C'est pratique pour essayer. Mais rien n'est gardé quand on ferme.

ON LE FAIT ENSEMBLE — LE PREMIER PROGRAMME DE LÉA

1. Léa veut afficher « Bonne partie ! ». Quelle instruction ?
2. Elle l'écrit dans `accueil.py`. Que fait-elle avant de lancer ?
3. Que tape-t-elle dans le terminal ?
4. Elle veut que le programme demande le prénom du joueur. Quelle instruction ?

Exemple

Pourquoi

✓	Je modifie, j'enregistre, puis je relance.	Python lit le fichier tel qu'il est enregistré sur le disque.
✗	« J'ai corrigé, mais l'erreur est toujours là. » sans avoir enregistré	Python a relu l'ancienne version du fichier.
✗	Taper <code>print("Bonjour")</code> dans le terminal PowerShell, sans lancer Python	PowerShell ne parle pas Python. Il faut un fichier lancé avec <code>python</code> , ou le mode interactif.

CHECKPOINT

À l'oral, 3 minutes. Chacun note sa réponse sur sa feuille, puis on corrige.

1. Quel est le rôle de l'interpréteur ?
2. Quelle instruction affiche, et quelle instruction lit le clavier ?

CAPSULE 2 SUR 4

Ranger des valeurs : variables et types

On sait lancer un programme. Mais pour calculer une cotisation, il faut garder des valeurs : un prix, un âge, un nom. Où les range-t-on ?

On les range dans des **VARIABLES**. Une variable est un nom posé sur une valeur, comme une étiquette sur une boîte.

On l'écrit avec le signe égal : `prix = 25`. Ce signe ne veut pas dire « égal » comme en maths. Il veut dire « range 25 dans la boîte nommée prix ».

Chaque valeur a un **TYPE** : sa nature. Python ne traite pas de la même façon un texte et un nombre.

Type	Ce que c'est	Exemple
<code>int</code>	un nombre entier	<code>4</code> , <code>340</code>
<code>float</code>	un nombre à virgule, écrit avec un point	<code>12.5</code> , <code>29.90</code>
<code>str</code>	du texte, entre guillemets	<code>"Catane"</code> , <code>"17"</code>
<code>bool</code>	vrai ou faux	<code>True</code> , <code>False</code>

La fonction `type()` donne le type d'une valeur : `type(12.5)` affiche `<class 'float'>`.

Attention au piège du début : `input()` rend **toujours** du texte. Si Léa tape 17, le programme reçoit `"17"`, entre guillemets. Pour calculer, il faut une **CONVERSION** : changer le type d'une valeur.

- 1 `int("17")` donne le nombre entier `17`.
- 2 `float("29.90")` donne le nombre à virgule `29.9`.
- 3 `str(340)` donne le texte `"340"`.

Sans conversion, attention : entre deux textes, `+` colle. `"3" + "4"` donne `"34"`.

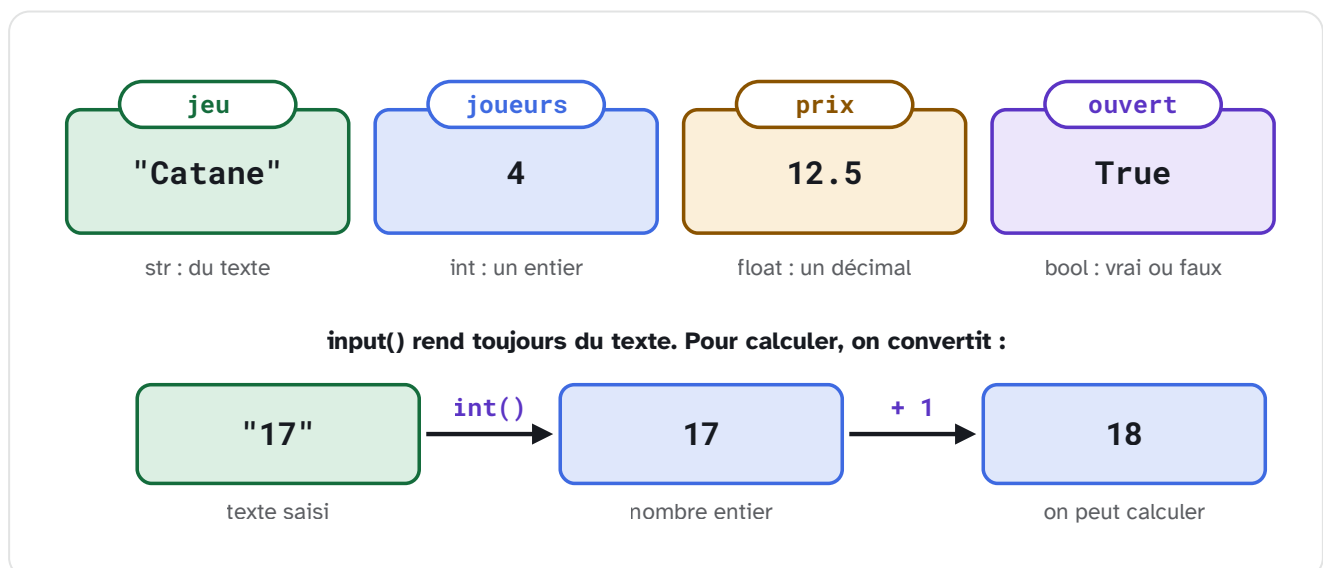


Schéma - Une variable est une étiquette posée sur une valeur ; chaque valeur a un type ; `int()` convertit un texte en nombre

INSTANT HISTOIRE — 1996 : ARIANE 5 ET LE NOMBRE TROP GRAND

Le 4 juin 1996, la fusée Ariane 5 explose 40 secondes après son premier décollage.

Un programme repris d'Ariane 4 range une vitesse dans une variable trop petite. Ariane 5 va plus vite : le nombre ne tient plus. L'ordinateur de bord reçoit des valeurs absurdes.

Le programme marchait sur Ariane 4. Pourquoi a-t-il échoué sur Ariane 5 ?

ON LE FAIT ENSEMBLE — DEUX TABLES DE JOUEURS

Le programme demande les joueurs de deux tables, puis les additionne : `t1 = input("Table 1 : ")`, `t2 = input("Table 2 : ")`, `print(t1 + t2)`. Léa tape 3, puis 4.

1. Quel est le type de `t1` ?
2. Qu'affiche le programme ?
3. Comment corriger ?

Exemple

Pourquoi

✓ <code>age = int(input("Votre âge : "))</code>	La saisie est convertie tout de suite en nombre entier.
✗ <code>int("29.90")</code>	Le texte contient un point : ce n'est pas un entier. Il faut <code>float()</code> .
✗ <code>prix = 12,5</code>	La virgule ne fait pas un nombre décimal. Il faut écrire <code>12.5</code> .

CHECKPOINT

À l'oral, 3 minutes. Chacun note sa réponse sur sa feuille, puis on corrige.

1. Donnez le type de chaque valeur : `25`, `"25"`, `25.0`.
2. Pourquoi faut-il convertir ce que rend `input()` avant de calculer ?

Calculer, comparer, mettre en forme

On sait garder des valeurs. Comment calculer avec elles, puis afficher un résultat propre ?

On utilise des **OPÉRATEURS** : des signes qui font un calcul ou une comparaison entre deux valeurs.

Opérateur	Ce qu'il fait	Exemple	Résultat
<code>+</code> , <code>-</code> , <code>*</code>	addition, soustraction, multiplication	<code>4 * 25</code>	<code>100</code>
<code>/</code>	division, toujours à virgule	<code>25 / 2</code>	<code>12.5</code>
<code>//</code>	division entière : combien de fois	<code>17 // 4</code>	<code>4</code>
<code>%</code>	le reste de la division entière	<code>17 % 4</code>	<code>1</code>
<code>==</code> , <code>!=</code>	égal, différent	<code>16 == 18</code>	<code>False</code>
<code><</code> , <code><=</code> , <code>></code> , <code>>=</code>	plus petit, plus grand	<code>16 < 18</code>	<code>True</code>

Une comparaison rend toujours `True` ou `False`. Attention : `=` range une valeur, `==` compare deux valeurs.

Pour afficher un résultat propre, on utilise une **chaîne formatée** : un texte précédé de la lettre `f`. Dans ce texte, chaque `{ }` est remplacée par une valeur.

- 1 On écrit `f` devant les guillemets.
- 2 On met le nom de la variable entre accolades : `{prix}`.
- 3 On ajoute la mise en forme après deux-points : `{prix:.2f}` affiche deux chiffres après le point.

Le programme

```
print(f"Demi-tarif : {prix:.2f} €")
```

prix vaut 12.5

```
Demi-tarif : 12.50 €
```

Le f devant les guillemets : les { } sont remplacées par la valeur.
:.2f veut dire : deux chiffres après le point.

Schéma • Dans une chaîne formatée, chaque paire d'accolades est remplacée par une valeur mise en forme

ON LE FAIT ENSEMBLE — LES TABLES DU TOURNOI

17 joueurs arrivent. Chaque table accueille 4 joueurs.

1. Combien de tables complètes ? Quel calcul ?
2. Combien de joueurs restent en attente ?
3. Une table a-t-elle plus de 3 joueurs ?
4. Écrivez l'affichage « 4 tables, 1 en attente » avec une chaîne formatée.

Exemple

Pourquoi

- | | |
|--|--|
| ✓ <code>f"Total : {total:.2f} €"</code> affiche
Total : 12.50 € | La valeur est mise en forme avec deux chiffres après le point. |
| ✗ <code>print(age = 18)</code> pour savoir si l'âge vaut 18 | <code>=</code> range une valeur. Pour comparer, il faut <code>==</code> :
<code>print(age == 18)</code> . |
| ✗ <code>"Total : " + total</code> avec <code>total</code> entier | On ne colle pas un texte et un nombre. Utilisez une chaîne formatée. |

CHECKPOINT

À l'oral, 3 minutes. Chacun note sa réponse sur sa feuille, puis on corrige.

1. Que donnent `25 / 2`, `25 // 2` et `25 % 2` ?
2. Avec `prix = 6`, qu'affiche `print(f"{prix:.2f} €")` ?

CAPSULE 4 SUR 4

Lire un message d'erreur

Revenons aux lignes rouges de Léa. Elles font peur. Pourtant, elles disent exactement où est le problème.

Il y a deux grandes familles d'erreurs. Une **ERREUR DE SYNTAXE** est une faute d'écriture : une parenthèse oubliée, un guillemet manquant. Python la voit en lisant le fichier, avant de commencer.

Rien ne s'exécute, même pas la première ligne.

Une **ERREUR D'EXÉCUTION** arrive pendant que le programme tourne. Chaque ligne est bien écrite, mais l'une d'elles ne peut pas se faire : un nom inconnu, un texte qu'on ajoute à un nombre.

Les lignes d'avant ont déjà tourné. Le programme s'arrête net à la ligne fautive.

C'est comme une lettre. Une faute d'orthographe se voit avant de l'envoyer. Une mauvaise adresse ne se voit qu'au moment où le facteur ne trouve pas la maison.

Un message d'erreur se lit toujours de la même façon :

- 1 **La dernière ligne** : le type d'erreur, puis le message. Par exemple `TypeError`, puis l'explication.
- 2 **La ligne** `File` : le nom du fichier et le numéro de la ligne fautive.
- 3 **La ligne recopiée** : le code exact qui a échoué, souligné par des `^` ou des `~`.
- 4 **On corrige** cette ligne, on enregistre, on relance.

Traceback (most recent call last):

File "...\\panne3.py", line 6, in <module>

print("Total à payer : " + total + " euros")

TypeError: can only concatenate str

(not "int") to str

— 1. Le fichier et la ligne

— 2. La ligne fautive, recopiée

— 3. Le type, puis le message

On lit de bas en haut : d'abord le type et le message, puis la ligne.

Erreur de syntaxe

rien ne s'exécute, pas même la ligne 1

Erreur d'exécution

le programme s'arrête à la ligne fautive

Schéma • Un message d'erreur donne le fichier, la ligne, la ligne fautive, le type et le message

Type d'erreur	Famille	Ce qu'il veut dire
<code>SyntaxError</code>	syntaxe	le code est mal écrit
<code>NameError</code>	exécution	ce nom de variable n'existe pas
<code>TypeError</code>	exécution	cette opération ne marche pas avec ces types
<code>ValueError</code>	exécution	la valeur ne convient pas, comme <code>int("douze")</code>

INSTANT HISTOIRE — 1947 : LE PREMIER « BUG » ÉTAIT UN VRAI INSECTE

En 1947, à l'université Harvard, l'équipe de Grace Hopper cherche pourquoi un calculateur fait des erreurs.

Elle trouve un papillon de nuit coincé dans une pièce de la machine. L'équipe le colle dans son cahier de bord, avec la mention « premier vrai cas de bug trouvé ».

Pourquoi l'équipe a-t-elle noté la panne dans un cahier ?

ON LE FAIT ENSEMBLE — LE MESSAGE DU PROGRAMME DE BRUNO

Le message : File "cotisation.py", line 5, puis la ligne `print("Total : " + total)`, puis `TypeError: can only concatenate str (not "int") to str`.

1. Quel est le type d'erreur ?
2. Quelle ligne est fautive ?
3. Que veut dire le message ?
4. Comment corriger ?

Exemple

Pourquoi

- | | | |
|---|--|---|
| ✓ | Je lis la dernière ligne, puis le numéro de ligne. | Le type et le message disent quoi chercher. Le numéro dit où. |
| ✗ | « Il y a une erreur, je recopie tout le programme. » | Le message donne la ligne : on corrige seulement celle-là. |
| ✗ | « La ligne indiquée est toujours la ligne fautive. » | Pour une parenthèse jamais fermée, Python montre la ligne où elle s'ouvre. La faute peut aussi venir d'une ligne au-dessus. |

CHECKPOINT

À l'oral, 3 minutes. Chacun note sa réponse sur sa feuille, puis on corrige.

1. Un programme n'affiche rien et montre `SyntaxError`. Pourquoi la première ligne n'a-t-elle pas tourné ?
2. Dans un message d'erreur, où lit-on le numéro de la ligne fautive ?

Ce qu'on retient

SI VOUS NE DEVEZ RETENIR QUE TROIS CHOSES

1. Écrivez, enregistrez, lancez avec `python fichier.py`, puis lisez le terminal.
2. Convertissez ce que rend `input()` avant de calculer : `int()` pour un entier, `float()` pour un décimal.
3. Lisez une erreur par le bas : le type, le message, puis le numéro de ligne.

Revenons à Léa. Les lignes rouges n'étaient pas une panne mystérieuse. Elles disaient le type d'erreur, le fichier et la ligne.

Vous savez maintenant les lire. Vous corrigez une seule ligne, vous enregistrez, vous relancez : le calcul des cotisations marche.

Variables, types et premières erreurs

COURS-U6-S1-01

L'essentiel

- **Interpréteur** : le logiciel Python qui lit le programme et l'exécute, ligne après ligne.
- **Variable** : un nom posé sur une valeur. `prix = 25` range 25 sous le nom `prix`.
- **Type** : la nature d'une valeur : `int`, `float`, `str`, `bool`.
- **Conversion** : `input()` rend toujours du texte. On le convertit avant de calculer.
- **Erreur** : de syntaxe, rien ne tourne. D'exécution, le programme s'arrête à la ligne fautive.

Les mots à connaître

Mot	En clair
Environnement de développement	L'éditeur fait pour le code, ici VS Code.
<code>print()</code>	Affiche à l'écran : une sortie.
<code>input()</code>	Lit le clavier : une entrée. Rend toujours un <code>str</code> .
<code>int()</code> , <code>float()</code> , <code>str()</code>	Convertissent en entier, en décimal, en texte.
Opérateur	Un signe qui calcule ou compare : <code>+</code> , <code>//</code> , <code>%</code> , <code>==</code> , <code><</code> .
Chaîne formatée	<code>f"Total : {total:.2f} €"</code> : les <code>{ }</code> sont remplacées par les valeurs.

Les types

Type	Exemple	Pour
<code>int</code>	<code>340</code>	compter

Type	Exemple	Pour
<code>float</code>	<code>12.5</code>	un prix, une moyenne
<code>str</code>	<code>"Catane"</code>	du texte
<code>bool</code>	<code>True</code>	le résultat d'une comparaison

La méthode pour une erreur

1. **Lisez** la dernière ligne : le type d'erreur et le message.
2. **Trouvez** la ligne `File` : le fichier et le numéro de ligne.
3. **Regardez** la ligne recopiée et ce qui est souligné.
4. **Corrigez** cette ligne, **enregistrez**, **relancez**.

Un exemple concret

Le programme de Léa calcule une cotisation.

```
age = input("Votre âge : ")
print("Dans un an :", age + 1)
```

```
TypeError: can only concatenate str (not "int") to str
```

Étape	Ce qu'on fait	Résultat
1	Type d'erreur	<code>TypeError</code> : erreur d'exécution, de type
2	Ligne fautive	la ligne 2 : <code>age + 1</code>
3	Cause	<code>age</code> est un <code>str</code> , rendu par <code>input()</code>
4	Correction	<code>age = int(input("Votre âge : "))</code>

Conclusion : avec la conversion, Léa tape 17 et le programme affiche `Dans un an : 18`.

Les pièges

- **✗ Oublier d'enregistrer** → Python relance l'ancienne version.
- **✗ Calculer avec `input()` sans convertir** → `"3" + "4"` donne `"34"`.
- **✗ Écrire `12,5`** → en Python, on écrit `12.5`.
- **✗ Confondre `=` et `==`** → `=` range, `==` compare.

Douze micro-exercices et trois pannes

LABO-U6-S1-01

Contexte — Vous êtes prestataire bénévole chez MaisonDesJeux, une association de jeux de société. Léa Fournier, animatrice, veut de petits programmes pour les soirées : accueil, tables, tarifs.

Avant d'écrire pour elle, vous vous entraînez sur douze exercices courts. Puis vous réparez trois programmes en panne, écrits par Bruno Sanchez, un bénévole.

RÈGLE IA DE LA SÉANCE

Scénario 1 : pas d'IA pendant ce labo. Vous avez les annexes, la documentation de Python et le web classique.

Si vous avez un doute, demandez au formateur ou à votre binôme.

À la fin de ce labo, vous saurez écrire et lancer un programme court, convertir et mettre en forme des valeurs. Vous saurez aussi lire un message d'erreur et corriger la ligne fautive.

Comment ça se valide — à chaque point de contrôle, vous montrez votre résultat au formateur. Il valide, puis vous passez à la suite.

Les mots de la séance

Mot	Ce que ça veut dire
Interpréteur	Le logiciel Python qui lit le programme et l'exécute, ligne après ligne.
Variable	Un nom posé sur une valeur : <code>prix = 25</code> .
Type	La nature d'une valeur : <code>int</code> entier, <code>float</code> décimal, <code>str</code> texte, <code>bool</code> vrai ou faux.
Conversion	Changer le type d'une valeur : <code>int("17")</code> donne <code>17</code> .
Chaîne formatée	Un texte précédé de <code>f</code> , où les <code>{ }</code> sont remplacées par des valeurs.
Message d'erreur	Les lignes que Python affiche quand il s'arrête : le fichier, la ligne, le type d'erreur.

Ce dont vous avez besoin

- Votre poste Windows, avec Python, VS Code et PowerShell.
- [Annexe A1](#) : le dossier de travail, dans le fichier `A1_premiers_pas.zip`.
- [Annexe A2](#) : les douze énoncés.
- [Annexe A3](#) : un extrait de la documentation de Python.
- [Annexe A4](#) : les trois programmes en panne.

En binôme — le pilote tape, le lecteur lit les énoncés et les annexes à voix haute, et vérifie chaque résultat. On inverse à la partie C.

Le plan du labo

Partie	Étapes	Ce que vous obtenez	Temps
A — Démarrer	1 à 5	un premier programme qui tourne	25 min
B — Variables, calculs, conversions	6 à 11	les exercices 2 à 7	35 min
C — Comparer et mettre en forme	12 à 17	les exercices 8 à 12, et les conversions expliquées	30 min
D — Trois pannes	18 à 22	trois programmes réparés, chaque message expliqué	30 min

PARTIE A — Démarrer

Étapes 1 à 5, environ 25 min. À la fin, votre premier programme s'affiche dans le terminal.

Besoin d'un rappel ? Revoir le cours : « [Écrire et lancer un programme](#) ».

01 Installer le dossier de travail

☐ Téléchargez `A1_premiers_pas.zip` depuis l'[annexe A1](#).

- ☐ Faites un clic droit sur le fichier, puis « Extraire tout ». Avec « Parcourir », choisissez votre dossier « Documents ».

Vous devez voir : un dossier `premiers_pas` dans « Documents ». Il contient douze fichiers, de `ex01.py` à `ex12.py`, et un dossier `pannes`.

Si vous voyez un dossier `premiers_pas` dans un autre dossier `premiers_pas` : utilisez celui qui contient directement `ex01.py`.

02 Ouvrir le dossier dans VS Code

- ☐ Ouvrez VS Code. Menu « Fichier », puis « Ouvrir le dossier ». Choisissez `Documents\premiers_pas`.
- ☐ Ouvrez un terminal : menu « Terminal », puis « Nouveau terminal ». Tapez :

```
ls
```

Vous devez voir : le dossier `pannes` et les douze fichiers `ex01.py` à `ex12.py`, dans la dernière colonne. Le terminal est dans le dossier `premiers_pas`.

03 Vérifier Python

- ☐ Dans le terminal, tapez :

```
python --version
```

Vous devez voir : `Python 3.` suivi d'un numéro, par exemple `Python 3.12.6`.

Si vous voyez « python n'est pas reconnu », ou si une fenêtre du Microsoft Store s'ouvre : tapez `py` à la place de `python`. Exemple : `py ex01.py`.

04 Essayer le mode interactif

☐ Tapez `python` seul, puis Entrée. Python affiche deux lignes d'information, puis `>>>`.

☐ Tapez ces deux lignes, chacune suivie d'Entrée :

```
2 + 3
print("Bonne partie !")
```

☐ Tapez `exit()` pour revenir au terminal.

Vous devez voir : `5`, puis `Bonne partie !`. Après `exit()`, l'invite redevient `PS C:\...\premiers_pas>`.

05 Écrire l'exercice 1

Voici le premier exercice, fait en entier. Les suivants se font de la même façon.

☐ Ouvrez `ex01.py`. Sous les deux lignes qui commencent par `#`, que Python ignore, tapez :

```
print("Bienvenue à MaisonDesJeux !")
print("Ouvert le mercredi, le vendredi et le samedi.")
```

☐ Enregistrez avec Ctrl+S, puis tapez dans le terminal :

```
python ex01.py
```

Vous devez voir :

```
Bienvenue à MaisonDesJeux !  
Ouvert le mercredi, le vendredi et le samedi.
```

Si vous voyez l'ancien résultat après une modification : le fichier n'est pas enregistré. Un petit rond sur l'onglet du fichier le signale. Faites Ctrl+S.

POINT DE CONTRÔLE A

- ☐ Montrez au formateur l'exercice 1 qui tourne dans le terminal.
- ☐ Le formateur a validé. Passez à la partie B.

PARTIE B — Variables, calculs, conversions

Étapes 6 à 11, environ 35 min. À la fin, les exercices 2 à 7 tournent.

Besoin d'un rappel ? Revoir le cours : « [Ranger des valeurs : variables et types](#) ».

Pour chaque exercice : lisez son énoncé dans l'[annexe A2](#), écrivez le code dans le fichier, enregistrez, lancez. L'[annexe A2](#) donne ce que vous devez voir.

06 Exercice 2 : trois variables et leur type

- ☐ Écrivez l'exercice 2 dans `ex02.py`, puis lancez `python ex02.py`.

Vous devez voir : `Catane 4 75`, puis trois lignes qui commencent par `<class`.

07 Exercice 3 : le prix de quatre adhésions

- ☐ Écrivez l'exercice 3 dans `ex03.py`, puis lancez-le.

Vous devez voir : `Total : 100`, puis `Demi-tarif : 12.5`, puis `<class 'float'>`.

Pourquoi ? 25 est un `int`. Pourquoi `25 / 2` donne-t-il un `float` ?

Votre réponse :

08 Exercice 4 : l'âge dans un an

L'exercice 4 est la première conversion. On le fait en deux temps : d'abord sans conversion, pour voir l'erreur.

- ☐ Dans `ex04.py`, tapez cette première version, enregistrez et lancez. Tapez 17 quand Python demande l'âge.

```
age = input("Votre âge : ")
print("Dans un an, vous aurez", age + 1, "ans")
```

- ☐ Notez le type d'erreur de la dernière ligne : _____
- ☐ Remplacez la première ligne par `age = int(input("Votre âge : "))`. Enregistrez et relancez avec 17.

Vous devez voir : Dans un an, vous aurez 18 ans .

09 Exercice 5 : les joueurs de deux tables

☐ Écrivez l'exercice 5 d'abord sans conversion, comme à l'étape 8. Lancez-le avec 3, puis 4.

☐ Notez ce qu'il affiche : _____

☐ Ajoutez les conversions. Relancez avec 3, puis 4.

Vous devez voir : Joueurs en tout : 7 avec les conversions.

10 Exercice 6 : le prix de deux exemplaires

☐ Écrivez l'exercice 6 sans conversion. Lancez-le avec 29.90 . Notez le résultat :

☐ Essayez avec `int()` . Lancez avec 29.90 . Notez le type d'erreur :

☐ Dans l'[annexe A3](#), partie « Convertir », trouvez la fonction qui convertit "29.90" en nombre. Utilisez-la, puis relancez.

Vous devez voir : Deux exemplaires : 59.8 .

Pourquoi ? Pourquoi `int("29.90")` refuse-t-il de convertir, alors que `int("17")` marche ?

Votre réponse :

11 Exercice 7 : répartir les joueurs en tables

☐ Écrivez l'exercice 7 dans `ex07.py`, avec les opérateurs `//` et `%`. Lancez-le.

Vous devez voir : 4 tables complètes, 1 joueur en attente.

POINT DE CONTRÔLE B

☐ Montrez au formateur les exercices 2 à 7 qui tournent. Répondez à sa question sur l'un d'eux.

☐ Le formateur a validé. Inversez les rôles, puis passez à la partie C.

PARTIE C — Comparer et mettre en forme

Étapes 12 à 17, environ 30 min. À la fin, les douze exercices tournent, et vous savez expliquer les trois conversions.

Besoin d'un rappel ? Revoir le cours : [« Calculer, comparer, mettre en forme »](#).

12 Exercices 8 et 9 : comparer

☐ Écrivez l'exercice 8 dans `ex08.py`, puis lancez-le.

☐ Écrivez l'exercice 9 dans `ex09.py`. Lancez-le deux fois : avec `MDJ2026`, puis avec `mdj2026`.

Vous devez voir : `True` puis `False` pour l'exercice 8. Pour l'exercice 9, `True`, puis `False` avec les minuscules.

13 Expliquer les trois conversions

Revenez sur les exercices 4, 5 et 6. Pour chacun, expliquez ce qui s'est passé sans conversion.

☐ Remplissez la grille. La première ligne est faite pour vous.

Exercice	Sans conversion, j'ai obtenu	Pourquoi	La conversion qui corrige
4	<code>TypeError</code>	<code>age</code> est un texte. On ne peut pas ajouter le nombre 1 à un texte.	<code>int()</code>
5	_____	_____	_____
6	_____	_____	_____

Vous devez avoir : trois résultats différents : une erreur, deux textes collés en `34`, un texte répété.

14 Trouver dans la doc

Les exercices 11 et 12 imposent une mise en forme. L'[annexe A3](#) donne les codes à écrire après les deux-points, dans les accolades.

☐ Ouvrez l'[annexe A3](#), partie « Mettre en forme ».

☐ Trouvez le code pour afficher deux chiffres après le point. Notez-le :

☐ Trouvez le code pour caler un texte à gauche sur 10 caractères. Notez-le :

☐ Trouvez le code pour caler un prix à droite sur 8 caractères, avec deux chiffres après le point. Notez-le : _____

Vous devez avoir : trois codes courts, à écrire après les deux-points dans les accolades.

15 Exercice 10 : la fiche d'un jeu

☐ Écrivez l'exercice 10 avec une chaîne formatée. Lancez-le.

Vous devez voir : Jeu : Catane | Joueurs : 3 à 4 | Durée : 75 min , au caractère près.

16 Exercice 11 : le demi-tarif

☐ Écrivez l'exercice 11. Lancez-le.

Vous devez voir : Demi-tarif adulte : 12.50 €, puis Demi-tarif jeune : 6.00 €.

17 Exercice 12 : le tableau des tarifs

☐ Écrivez l'exercice 12, avec les codes de l'étape 14. Lancez-le.

Vous devez voir : trois lignes dont les prix sont alignés à droite, comme le gabarit de l'[annexe A2](#).

Pourquoi ? Pourquoi une chaîne formatée est-elle plus sûre que `"Total : " + total` pour afficher un nombre ?

Votre réponse :

POINT DE CONTRÔLE C

- ☐ Montrez au formateur votre grille de l'étape 13 et les exercices 10 à 12.
- ☐ Le formateur a validé. Passez à la partie D.

PARTIE D — Trois pannes

Étapes 18 à 22, environ 30 min. À la fin, les trois programmes de Bruno marchent, et vous savez dire ce que voulait dire chaque message.

Besoin d'un rappel ? Revoir le cours : [« Lire un message d'erreur »](#).

Les trois programmes sont dans le dossier `pannes`. Leur code est aussi dans l'[annexe A4](#). Chacun contient une seule erreur, d'une nature différente.

18 Panne 1, faite avec vous

☐ Tapez dans le terminal :

```
python pannes\panne1.py
```

Vous devez voir : un message qui finit par `SyntaxError: '(' was never closed`. Rien d'autre ne s'affiche, même pas `Bonjour Léa`.

☐ Lisez la ligne d'exemple de la grille. Elle est faite pour vous.

Panne	Type d'erreur	Ligne	La ligne fautive	Ma correction
1	<code>SyntaxError</code>	4	<code>print("Bienvenue à MaisonDesJeux !")</code>	ajouter la parenthèse fermante à la fin
2	_____	_____	_____	_____
3	_____	_____	_____	_____

☐ Ouvrez `pannes\panne1.py`, corrigez la ligne 4, enregistrez, relancez.

Vous devez voir : `Bonjour Léa`, `Bienvenue à MaisonDesJeux !`, `Bonne partie !`.

19 Panne 2

☐ Lancez `python pannes\panne2.py`. Remplissez la ligne 2 de la grille. Python peut proposer le bon nom : `Did you mean...`.

☐ Corrigez, enregistrez, relancez.

Vous devez voir : Adhérents cette saison : 352 .

20 Panne 3

☐ Lancez `python pannes\panne3.py` . Remplissez la ligne 3 de la grille.

☐ Corrigez avec une chaîne formatée : `{total}` , sans code de mise en forme. Enregistrez, relancez.

Vous devez voir : Calcul du prix des adhésions... , puis Total à payer : 100 euros .

21 Dire ce que voulait dire chaque message

☐ Pour chaque panne, écrivez en une phrase, avec vos mots, ce que le message voulait dire. La première est faite pour vous.

Panne	Le message voulait dire
-------	-------------------------

1	Une parenthèse ouverte à la ligne 4 n'a jamais été fermée.
---	--

2	
---	-------

3	
---	-------

Vous devez avoir : une phrase sans mot anglais pour chaque panne.

22 Syntaxe ou exécution ?

- ☐ Avant la correction, la panne 1 n'affichait rien. La panne 3 affichait sa première ligne. Notez la famille de chaque erreur : _____

Vous devez avoir : une famille pour chacune des trois pannes, et deux familles différentes en tout.

Pourquoi ? La ligne 3 de la panne 1 était juste. Pourquoi n'a-t-elle pas affiché `Bonjour Léa` ?

Votre réponse :

POINT DE CONTRÔLE D

- ☐ Montrez au formateur vos grilles des étapes 18 et 21, et les trois programmes qui tournent.
- ☐ Le formateur a validé. Le labo est terminé.

POUR ALLER PLUS LOIN

10 à 15 min, si vous avez fini avant la fin de la séance.

Bruno a écrit une quatrième ligne pour Léa. Elle ne produit aucun message d'erreur, mais le résultat est faux.

```
moyenne = 25 + 12 / 2
print("Prix moyen :", moyenne)
```

-
- ☐ Créez `bonus.py` avec ces deux lignes. Lancez-le. Notez le résultat :
- _____
-
- ☐ La moyenne de 25 et 12 vaut 18.5. Avec l'ordre des opérations de l'[annexe A3](#), corrigez la ligne 1 par des parenthèses. Relancez.
-
- ☐ Écrivez en une phrase pourquoi Python n'a affiché aucune erreur :
- _____
-

CE QU'IL FAUT RETENIR

1. Écrivez, enregistrez, lancez, lisez : à chaque modification, on refait les quatre gestes.
2. `input()` rend du texte : convertissez avec `int()` ou `float()` avant de calculer.
3. Un message d'erreur se lit par le bas : le type et le message, puis le fichier et la ligne.

Tous les documents du module

A1 — Le dossier de travail

↓ Télécharger A1_premiers_pas.zip

Source : fichiers écrits pour le module. Fichier [A1_premiers_pas.zip](#) à télécharger ci-dessus. Utilisée dans : LABO étapes 1 et 2.

Le fichier [A1_premiers_pas.zip](#) contient un dossier `premiers_pas` :

Élément	Ce que c'est
<code>ex01.py</code> à <code>ex12.py</code>	un fichier par exercice. Chacun contient seulement deux lignes de commentaire : le titre de l'exercice, et la place où écrire votre code.
<code>pannes\panne1.py</code>	le premier programme en panne de Bruno
<code>pannes\panne2.py</code>	le deuxième programme en panne
<code>pannes\panne3.py</code>	le troisième programme en panne

Une ligne qui commence par `#` est un **commentaire** : Python ne l'exécute pas. Elle sert à expliquer le code.

Le mode d'emploi

1. Téléchargez le zip, puis extrayez-le dans « Documents » : clic droit, « Extraire tout ».
2. Ouvrez le dossier `Documents\premiers_pas` dans VS Code.
3. Pour repartir de zéro, supprimez le dossier `premiers_pas` et extrayez de nouveau le zip.

A2 — Les douze énoncés

Source : énoncés écrits pour le module. Tarifs : document de scénario, § 5 « Volumétrie et règles métier ». Jeu Catane : document de scénario, § 5, contenu de la ludothèque. Utilisée dans : LABO étapes 5 à 17.

Pour chaque exercice : écrivez le code dans le fichier du même numéro, enregistrez, lancez avec `python exNN.py` . Le résultat doit être celui de « Vous devez voir », au caractère près. Ce que vous tapez au clavier est écrit après le deux-points de la question.

Exercice 1 — Afficher un message d'accueil

Affichez ces deux lignes :

```
Bienvenue à MaisonDesJeux !  
Ouvert le mercredi, le vendredi et le samedi.
```

Exercice 2 — Trois variables et leur type

Rangez dans trois variables le nom d'un jeu, `"Catane"` , le nombre maximum de joueurs, `4` , et la durée d'une partie en minutes, `75` . Appelez-les `jeu` , `joueurs_max` et `duree` .

Affichez les trois valeurs sur une ligne, avec un seul `print` . Puis affichez le type de chacune avec `type()` , une par ligne.

Vous devez voir :

```
Catane 4 75  
<class 'str'>  
<class 'int'>  
<class 'int'>
```

Exercice 3 — Le prix de quatre adhésions

Une adhésion adulte coûte 25 €. Rangez ce prix dans la variable `prix_adulte` .

Calculez le prix de quatre adhésions adultes, et affichez `Total :` suivi du prix. Puis calculez le demi-tarif d'une adhésion, et affichez `Demi-tarif :` suivi du demi-tarif. Affichez enfin le type du demi-tarif.

Vous devez voir :

```
Total : 100  
Demi-tarif : 12.5  
<class 'float'>
```

Exercice 4 — L'âge dans un an

Demandez l'âge avec `input("Votre âge : ")`. Affichez l'âge qu'aura la personne dans un an.

Vous devez voir, en tapant 17 :

```
Votre âge : 17
Dans un an, vous aurez 18 ans
```

Exercice 5 — Les joueurs de deux tables

Demandez le nombre de joueurs avec `input("Joueurs à la table 1 : ")`, puis avec `input("Joueurs à la table 2 : ")`. Affichez `Joueurs en tout :` suivi du total.

Vous devez voir, en tapant 3 puis 4 :

```
Joueurs à la table 1 : 3
Joueurs à la table 2 : 4
Joueurs en tout : 7
```

Exercice 6 — Le prix de deux exemplaires

Demandez le prix avec `input("Prix du jeu : ")`. Affichez `Deux exemplaires :` suivi du prix multiplié par 2.

Vous devez voir, en tapant 29.90 :

```
Prix du jeu : 29.90
Deux exemplaires : 59.8
```

Exercice 7 — Répartir les joueurs en tables

17 joueurs arrivent au tournoi. Une table accueille 4 joueurs.

Calculez le nombre de tables complètes avec `//`, et les joueurs en attente avec `%`. Affichez la phrase du résultat ci-dessous.

Vous devez voir :

```
4 tables complètes, 1 joueur en attente
```

Exercice 8 — Tarif jeune ou pas ?

Le tarif jeune est pour les moins de 18 ans. Rangez `16` dans la variable `age`. Affichez le résultat de la comparaison « age plus petit que 18 », puis de « age plus grand ou égal à 18 ».

Vous devez voir :

```
True
False
```

Exercice 9 — Le code du coffre à jeux

Le coffre à jeux s'ouvre avec le code `MDJ2026`.

Demandez le code avec `input("Code du coffre à jeux : ")`. Affichez le résultat de la comparaison entre le code tapé et `"MDJ2026"`. Python affiche `True` si c'est le bon code, `False` sinon.

Vous devez voir, en tapant MDJ2026 :

```
Code du coffre à jeux : MDJ2026
True
```

Exercice 10 — La fiche d'un jeu

Rangez dans quatre variables : `jeu` vaut `"Catane"`, `j_min` vaut `3`, `j_max` vaut `4`, `duree` vaut `75`. Affichez la fiche avec une seule chaîne formatée. Le signe `|` se tape avec AltGr + 6.

Gabarit imposé :

```
Jeu : Catane | Joueurs : 3 à 4 | Durée : 75 min
```

Exercice 11 — Le demi-tarif

Calculez le demi-tarif adulte, à partir de 25, et le demi-tarif jeune, à partir de 12. Affichez-les avec deux chiffres après le point.

Gabarit imposé :

```
Demi-tarif adulte : 12.50 €
Demi-tarif jeune : 6.00 €
```

Exercice 12 — Le tableau des tarifs

Affichez les trois tarifs de l'association : Jeune 12 €, Adulte 25 €, Famille 55 €.

Le nom est calé à gauche sur 10 caractères. Le prix est calé à droite sur 8 caractères, avec deux chiffres après le point. Une espace sépare le prix du signe €.

Gabarit imposé :

```
Jeune      12.00 €
Adulte     25.00 €
Famille    55.00 €
```

A3 — Extrait de la documentation de Python

Source : *documentation officielle de Python 3*, docs.python.org, *fonctions natives et mini-langage de format, traduite et raccourcie pour le module*. Utilisée dans : LABO étapes 10 et 14.

Afficher et lire

Fonction	Ce qu'elle fait	Exemple
<code>print(a, b)</code>	Affiche les valeurs, séparées par une espace, puis passe à la ligne.	<code>print("Total :", 100)</code> affiche <code>Total : 100</code>
<code>input(question)</code>	Affiche la question, attend une saisie terminée par Entrée. Rend toujours un <code>str</code> .	<code>nom = input("Prénom : ")</code>
<code>type(valeur)</code>	Donne le type de la valeur.	<code>type(4)</code> affiche <code><class 'int'></code>

Convertir

Fonction	Ce qu'elle fait	Exemple	Refuse
<code>int(x)</code>	Rend un nombre entier.	<code>int("17")</code> donne <code>17</code>	<code>int("29.90")</code> , <code>int("douze")</code> : <code>ValueError</code>

Fonction	Ce qu'elle fait	Exemple	Refuse
<code>float(x)</code>	Rend un nombre à virgule.	<code>float("29.90")</code> donne <code>29.9</code>	<code>float("29,90")</code> : <code>ValueError</code>
<code>str(x)</code>	Rend du texte.	<code>str(340)</code> donne <code>"340"</code>	rien

Calculer et comparer

Opérateur	Sens	Exemple
<code>+</code> <code>-</code> <code>*</code>	addition, soustraction, multiplication	<code>4 * 25</code> donne <code>100</code>
<code>/</code>	division : rend toujours un <code>float</code>	<code>25 / 2</code> donne <code>12.5</code>
<code>//</code>	division entière	<code>17 // 4</code> donne <code>4</code>
<code>%</code>	reste de la division entière	<code>17 % 4</code> donne <code>1</code>
<code>==</code> <code>!=</code>	égal, différent	<code>"A" == "a"</code> donne <code>False</code>
<code><</code> <code><=</code> <code>></code> <code>>=</code>	comparaisons	<code>16 < 18</code> donne <code>True</code>

Comme en maths, `*`, `/`, `//` et `%` passent avant `+` et `-`. Les parenthèses passent avant tout.

Avec du texte, `+` colle deux textes : `"3" + "4"` donne `"34"`. Et `*` répète un texte : `"ab" * 2` donne `"abab"`.

Mettre en forme

Une chaîne formatée commence par `f` devant les guillemets. Chaque `{valeur}` est remplacée par la valeur. On peut ajouter un code de mise en forme après deux-points : `{valeur:code}`.

Code	Effet	Exemple	Résultat
<code>.2f</code>	deux chiffres après le point	<code>f"{12.5:.2f}"</code>	<code>12.50</code>
<code><10</code>	calé à gauche sur 10 caractères	<code>f"[{'Jeune':<10}]"</code>	<code>[Jeune]</code>
<code>>8</code>	calé à droite sur 8 caractères	<code>f"[{12:>8}]"</code>	<code>[12]</code>
<code>>8.2f</code>	calé à droite sur 8, deux chiffres après le point	<code>f"[{12:>8.2f}]"</code>	<code>[12.00]</code>

Les crochets des exemples servent seulement à voir les espaces.

Les erreurs les plus courantes

Erreur	Quand	Exemple
<code>SyntaxError</code>	le code est mal écrit : Python refuse de commencer	une parenthèse jamais fermée
<code>NameError</code>	un nom utilisé n'existe pas	une faute de frappe dans un nom de variable
<code>TypeError</code>	une opération ne marche pas avec ces types. Dans le message, <i>concatenate</i> veut dire coller bout à bout.	<code>"Total : " + 100</code>
<code>ValueError</code>	le type est bon, mais pas la valeur	<code>int("douze")</code>

A4 — Les trois programmes en panne

Source : *programmes écrits pour le module. Fichiers dans le dossier `pannes` du zip de l'[annexe A1](#). Utilisée dans : LABO étapes 18 à 20.*

Bruno Sanchez a écrit ces trois programmes pour Léa Fournier. Chacun contient une seule erreur. Les numéros de ligne sont donnés à gauche : ils ne font pas partie du code.

panne1.py

```
1 # Panne 1 : le message d'accueil de la salle de jeux
2 nom = "Léa"
3 print("Bonjour", nom)
4 print("Bienvenue à MaisonDesJeux !")
5 print("Bonne partie !")
```

panne2.py

```
1 # Panne 2 : le nombre d'adhérents de l'association
2 nb_adherents = 340
3 nb_nouveaux = 12
4 total = nb_adherent + nb_nouveaux
5 print("Adhérents cette saison :", total)
```

panne3.py

```
1  # Panne 3 : le prix de quatre adhésions adultes
2  print("Calcul du prix des adhésions...")
3  prix_adulte = 25
4  nb_adultes = 4
5  total = prix_adulte * nb_adultes
6  print("Total à payer : " + total + " euros")
```