# Python - C: les différences de syntaxe

Pour les élèves ayant appris les bases de Python, découvrir le langage C peut être déstabilisant au début en raison de différences syntaxiques. Dans cet article, nous allons expliquer en détail ces différences sur des aspects clés comme les déclarations de variables, les blocs de code, les conditions, les boucles et les fonctions d'affichage. À la fin, vous comprendrez comment passer de Python à C avec des exemples concrets.

5TTR 6TTR Découverte

# Instructions (retour à la ligne vs `;`)

## En python

En Python, les instructions sont séparées par un retour à la ligne:

```python
annee_courante = 40
annee_naissance = 20

age = annee_courante - annee_naissance

```

En C, par contre, les instructions sont séparées par des '`;`' (points-virgules):

```c
int annee_courante = 40;
int annee_naissance = 20;

int age = annee_courante - annee_naissance;
```

Les **retours à la ligne** ne sont pas obligatoires en C, même s'ils restent **FORTEMENT recommandés pour raisons de lisibilité**.

# Blocs de code : indentations vs accolades `{}`

## En Python

Les blocs de code sont définis par **l'indentation**. Aucune accolade n'est nécessaire.

**Exemple :**

```python
if x > 0:
    print("x est positif")
    print("Fin du bloc")
```

## En C

En C, les blocs de code sont délimités par des **accolades** `{}`. L'indentation est optionnelle mais recommandée pour améliorer la lisibilité.

**Exemple équivalent en C :**

```c
if (x > 0) {
    printf("x est positif");
    printf("Fin du bloc");
}
```

---

# Conditions ( `elif` vs `else if` )

## En Python

En Python, lorsqu'une condition `if` est fausse, vous pouvez tester une autre condition en utilisant `elif`. C'est une syntaxe compacte qui permet d'enchaîner plusieurs tests dans un même bloc.

**Exemple :**

```python
age = 20

if age < 10:
    print("Enfant")
elif age < 18:
    print("Adolescent")
elif age < 30:
    print("Jeune adulte")
else:
    print("Adulte")
```

## En C

En C, il n'existe pas de mot-clé unique équivalent à `elif`. Vous devez utiliser `else if` avec un espace. Chaque bloc `else if` est suivi de la condition entre parenthèses et utilise des accolades `{}` pour le code.

**Exemple équivalent en C :**

```c
int age = 20;

if (age < 10) {
    printf("Enfant");
} else if (age < 18) {
    printf("Adolescent");
} else if (age < 30) {
    printf("Jeune adulte");
} else {
    printf("Adulte");
}
```

## Points à noter

- En Python, `elif` est plus compact, alors que le C utilise deux mots-clés séparés (`else if`).
- En C, les parenthèses autour des conditions sont obligatoires.
- Les accolades `{}` délimitent les blocs de code en C, tandis que l'indentation est suffisante en Python.

---

# Afficher des informations ("à l'écran")

Les micro-contrôlleurs n'ont généralement pas d'écran. L'affichage des informations (en mode développement) se fait généralement sur l'ordinateur hôte auquel le micro-contrôlleur est connecté.

## En Python

En Python, la fonction `print()` est utilisée pour afficher des messages ou des valeurs dans la console. Vous pouvez facilement afficher plusieurs valeurs en les séparant par des virgules.

**Exemple :**

```python
temperature = 22.5
print("La température est :", temperature)
```

## En C - La fonction `printf`

La fonction `printf` (pour **print formatted**) sert à **afficher du texte ou des valeurs** à l'écran. Elle fait partie de la bibliothèque `stdio.h`, qu'il faut donc inclure au début du programme :

```c
#include <stdio.h>
```

Sa syntaxe de base est :

```c
printf("texte à afficher");
```

Tu peux afficher plusieurs types de données en utilisant des **spécificateurs de format** :

- `%d` → un entier
- `%f` → un nombre décimal (float)
- `%c` → un caractère
- `%s` → une chaîne de caractères

Exemple :

```c
int age = 16;
printf("J'ai %d ans.\n", age);
```

Le `\n` à la fin du texte sert à **aller à la ligne suivante**.

# Déclaration des variables

## En Python

Pas besoin de déclarer explicitement le type de la variable : Python le déduit automatiquement lors de l'assignation.

**Exemple :**

```python
x = 10         # Entier
y = 22.5       # Flottant
message = "OK" # Chaîne de caractères
```

## En C

En C, il faut **toujours déclarer le type** de chaque variable avant de l'utiliser. Cette règle permet au compilateur d'optimiser la mémoire.

**Exemple équivalent en C :**

```c
int x = 10;          // Entier
float y = 22.5;      // Flottant
char message[] = "OK"; // Chaîne de caractères
```

## Différence clé

En Python, le typage est dynamique et flexible. En C, le typage est strict et doit être spécifié à l'avance.

---

# Opérateurs logiques : `and`, `or` vs `&&`, `||`

## En Python

Les opérateurs logiques en Python sont écrits en toutes lettres : `and` pour ET, `or` pour OU.

**Exemple :**

```python
if x > 0 and y < 10:
    print("x est positif ET y est inférieur à 10")

if x > 0 or y < 10:
    print("x est positif OU y est inférieur à 10")
```

## En C

En C, les opérateurs logiques sont symboliques :

- `&&` pour ET
- `||` pour OU

**Exemple équivalent en C :**

```c
if (x > 0 && y < 10) {
    printf("x est positif ET y est inférieur à 10");
}

if (x > 0 || y < 10) {
    printf("x est positif OU y est inférieur à 10");
}
```

## Différence clé

Python utilise des mots-clés (`and`, `or`), alors que C utilise des symboles (`&&`, `||`).

---

# Boucle `while`

## En Python

Une boucle `while` s'écrit simplement et utilise l'indentation pour définir le bloc de code.

**Exemple :**

```python
x = 0
while x < 5:
    print(x)
    x += 1
```

## En C

En C, la boucle `while` est similaire, mais les blocs de code sont définis par des accolades `{}`.

**Exemple équivalent en C :**

```c
int x = 0;
while (x < 5) {
    printf("%d", x);
    x += 1;
}
```

# Conclusion

Les différences entre Python et le C se situent principalement au niveau de la syntaxe. Le C est plus strict avec ses exigences sur les types, l'utilisation des accolades `{}` et les points-virgules `;`, tandis que Python privilégie la lisibilité et la flexibilité. Cependant, une fois ces différences comprises, vous pourrez facilement adapter vos compétences en Python pour écrire des programmes robustes en C. La pratique régulière et des exemples concrets, comme ceux présentés ici, vous aideront à maîtriser ces deux langages.