Éviter les nombres magiques dans le code

En programmation et en algorithmique, de nombreuses erreurs ou incompréhensions ne viennent pas de la complexité du code, mais de sa lisibilité et de sa maintenabilité. Parmi les mauvaises pratiques les plus courantes figure l'utilisation de *nombres magiques*.

5TTR 6TTR Découverte

> En programmation et en algorithmique, de nombreuses erreurs ou incompréhensions ne viennent pas de la complexité du code, mais de sa lisibilité et de sa maintenabilité.

Parmi les mauvaises pratiques les plus courantes figure l'utilisation de *nombres magiques*.

Qu'est-ce qu'un nombre magique ?

Un **nombre magique** est une valeur numérique écrite directement dans le code **sans explication claire de sa signification**.

```
if score >= 42:
    print("Niveau suivant")
```

À la lecture, rien n'indique pourquoi `42` est utilisé. Ce nombre a un sens fonctionnel, mais ce sens est **implicite**, donc fragile.

Pourquoi est-ce une mauvaise pratique ?

Les nombres magiques posent plusieurs problèmes concrets :

- **Lisibilité réduite** Le code ne s'explique plus de lui-même. Le lecteur doit deviner l'intention.
- **Maintenance difficile** Si la valeur doit changer, il faut la retrouver partout dans le code, avec un risque d'oubli.
- **Risque d'erreurs** Modifier un nombre à un endroit et pas à un autre peut introduire des bugs subtils.
- **Couplage fort à une règle métier** La logique métier est disséminée dans le code au lieu d'être centralisée.

Bonne pratique : utiliser des constantes nommées

La solution consiste à remplacer les nombres magiques par des **constantes explicites**.

```python
SCORE_MINIMUM_NIVEAU_SUIVANT = 42

if score >= SCORE_MINIMUM_NIVEAU_SUIVANT:
    print("Niveau suivant")
```

Avantages immédiats :

- le code devient **auto-documenté** ;
- la règle métier est **centralisée** ;
- la modification est **simple et sûre**.

Exemple en algorithmique

```
TEMPS_MAX_ATTENTE = 30

tant que temps < TEMPS_MAX_ATTENTE
    attendre 1 seconde
fin tant que
```

Ici, la valeur `30` n'est plus arbitraire : elle représente clairement une contrainte fonctionnelle.

Cas fréquents de nombres magiques à éviter

- limites (`100` , `255` , `1024`)
- durées (`60` , `1000` , `30`)
- seuils (`0.8` , `50` , `18`)
- tailles (`8` , `16` , `32`)
- codes d'état (`1` , `2` , `-1`)

Dès qu'un nombre **a un sens métier ou logique**, il mérite un nom.

Bonnes pratiques complémentaires

- Regrouper les constantes en début de fichier ou dans un module dédié
- Utiliser des constantes plutôt que des commentaires explicatifs
- Préférer des noms longs et explicites à des noms courts ambigus

À retenir

> Un bon code ne cherche pas seulement à fonctionner, il cherche à être **LU, COMPRIS ET MODIFIÉ**.

Éliminer les nombres magiques est une règle simple qui améliore immédiatement la qualité du code, quel que soit le langage utilisé.