

1 Problématique

Un serveur *World of Warcraft* représente 5,5 millions de lignes de code, *Windows 7* 40 millions et *Facebook* plus de 62 millions.

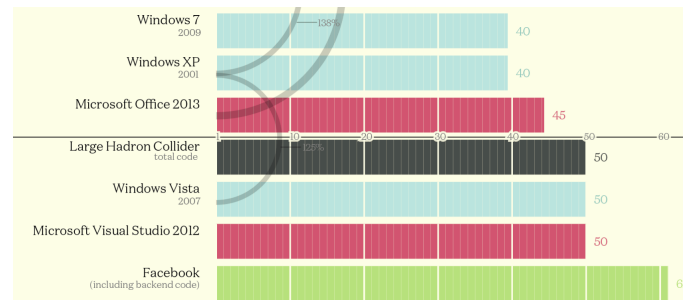


FIGURE 1 – Infographie complète sur informationisbeautiful

Il est impératif que chaque bloc de code (boucle, fonction...) réalise correctement la tâche qui lui est assignée.

Quels systèmes de contrôle peut-on mettre en place dans un programme informatique ?

2 Variant de boucle : terminaison

```

1 i = 10
2 while i >= 0:
3     print(i)
4     print("Boum")

```

Code 1 – Boucle infinie

Le code 1 ne termine jamais. Nous pouvons supposer que ce n'était pas la volonté initiale du programmeur.

À retenir

Un *variant de boucle* est une expression dont la valeur varie à chaque itération de la boucle.

Pour chaque boucle construite, il est donc nécessaire de déterminer son *variant* afin de prouver sa **terminaison**.

```

1 i = 10
2 while i >= 0:
3     print(i)
4     i = i - 1
5 print("Boum")

```

Code 2 – Compte à rebours

Dans le code 2, i est un variant de la boucle : à chaque itération sa valeur décroît de 1, elle deviendra donc négative.

Activité 1 : L'algorithme 3 permet de déterminer si *mot* est un palindrome.

```
1  Fonction: palindrome(mot)
2  Debut
3  i = 0
4  j = longueur(mot) - 1
5  TantQue i <= j faire
6  Si mot_i = mot_j alors
7      i = i + 1
8      j = j - 1
9  sinon
10     Renvoyer Faux
11  Fin Si
12  Fin TantQue
13  Renvoyer Vrai
14  Fin
```

Code 3 – Palindrome

1. Sur papier, dérouler l'algorithme pour les mots *radar* puis *sauras*.
2. Écrire cet algorithme en Python.
3. Montrer que l'expression $\delta = j - i$ est un variant de la boucle.

3 Invariant de boucle : correction

Une boucle peut terminer mais cependant ne pas réaliser ce à quoi elle est destinée.

À retenir

Un *invariant de boucle* est une propriété qui si elle est vraie avant l'itération, le reste après son exécution.

```
1  def somme_des_entiers(n):
2      s = 0
3      for i in range(1, n+1):
4          s += i
5      return s
```

Code 4 – Somme des premiers entiers

Le code 4 calcule la somme $1 + 2 + \dots + n$. Nous pouvons montrer par récurrence que l'expression

$$s = 1 + 2 + \dots + i - 1$$

est un invariant de la boucle :

- Avant la première itération, $i = 1$ et $s = 0$; l'expression est vérifiée.
- On suppose que la condition est vérifiée au rang i .
- On montre alors qu'elle est vraie au rang $i + 1$.

Remarque

À la dernière itération, $i = n + 1$; la boucle n'est alors pas exécutée et l'invariant est bien vérifié :

$$s = 1 + 2 + \dots + (n + 1) - 1$$

$$s = 1 + 2 + \dots + n$$

Activité 2 : La division de l'entier a par l'entier b définie par Euclide consiste à trouver deux entiers q et r tels que

$$a = q \times b + r$$

avec $0 \leq r < b$

Le code 5 détermine ces deux entiers.

```
1 def division(a, b):
2     q = 0
3     r = a
4     while r >= b:
5         q += 1
6         r -= b
7     return (q, r)
```

Code 5 – Division euclidienne

1. Dérouler à la main la fonction pour $a = 20$ et $b = 6$.
2. Montrer que l'expression $a = q \times b + r$ est un invariant de la boucle.

4 Préconditions

Reprenons la fonction `somme_des_entiers` et passons lui le paramètre `-2`.

```
1 print(somme_des_entiers(-2))
```

Code 6 – Somme d'entiers négatifs ?

Le résultat est inattendu ou plus exactement il ne correspond pas à ce que nous voudrions obtenir. La fonction telle que nous l'avons écrite n'est adaptée qu'à des valeurs de n positives.

À retenir

Une **assertion** permet d'effectuer un test et lève une erreur si la condition n'est pas remplie.

```
1 def somme_des_entiers(n):  
2     assert n >= 0, "n doit être positif ou nul"  
3     s = 0  
4     for i in range(1, n+1):  
5         s += i  
6     return s
```

Code 7 – Précondition

Activité 3 : Compléter la fonction comme le montre le code 7 et tester à nouveau le code 6.

L'erreur levée donne des informations sur le chemin qui a conduit à cette erreur.

Activité 4 : Une division par zéro est une opération qui n'existe pas. Modifier alors la fonction **division** pour qu'elle définisse une precondition nécessaire à son exécution.