L'intelligence artificielle

Séance 3 : Il y a un Bug !

Quels sont les impacts sociétaux des bugs informatiques et comment peut-on les corriger ?

Exemples de bugs aux conséquences fâcheuses :

De nos jours, les programmes informatiques complexes comptent souvent plusieurs millions de lignes de codes (C). Mal conçu, un logiciel peut nuire à la vie d'un grand nombre de personnes.

Par exemple, en 2011, le ministère français de la Défense décide d'utiliser un nouveau logiciel (nommé *Louvois*) afin de faciliter la gestion des salaires. Près de 120 000 bulletins de salaire erronés ont été édités. Beaucoup de militaires ont dû s'endetter pour faire face à ces erreurs de gestion. Les raisons de l'échec de ce programme sont connues : absence de tests approfondis, réduction du nombre de points de contrôle, complexité des 174 régimes de prime des militaires.

Exemple de programme	Nombre de lignes (en million)
Jeu *Age of Empire* de Microsoft	1,2
Gestion du robot Curiosity sur Mars	5
Informatique embarquée de l'avion F 35	24
Réseau social Facebook	50
Moteur de recherche Google	2 500

C Nombre de lignes de code de quelques programmes.

DOC **Le cas de Toyota.** Toyota a dû rappeler au garage 1,9 millions de véhicules en 2014 et 625 000 de plus en 2015 pour une mise à jour du logiciel d'une trentaine de minutes. Cette opération très coûteuse pour le constructeur a été rendue nécessaire par la présence d'un bug qui entraînait l'arrêt de la voiture dans certaines conditions.

??? Mener une recherche et trouver trois autres bugs informatiques qui ont eu de lourdes conséquences (à préciser).

POINT COURS :

▶ Un programme complexe moderne peut comporter plusieurs centaines de millions de lignes de code, ce qui rend très probable la présence de **bugs** (ou bogues). Ces erreurs peuvent conduire un programme à avoir un comportement inattendu et entraîner des conséquences graves.

Def :

• **Bug :** Erreur dans un programme entraînant un comportement inattendu lors de l'utilisation du programme.

Détection et correction d'erreurs dans un programme :

Voici un programme Python permettant de calculer la moyenne des éléments d'une liste.

```
def moyenne(liste) :
    somme = 0                    # On calcule la somme
    for val in liste :           # de tous les éléments de la liste
        somme = somme + val      # en les ajoutant un par un
    return somme / len(liste)    # On renvoie la moyenne, correspondant à cette somme
                                 # divisée par le nombre d'éléments de la liste
```

Nous allons appliquer les tests suivants qui doivent vérifier tous les comportements de cette fonction.
On va donc vérifier que moyenne([1,2,3]) renvoie bien 2.0

moyenne([1,2,3]) `#Renvoie 2.0 donc la fonction se comporte correctement`

Faisons maintenant le test avec une liste vide : moyenne([])

On obtient le message d'erreur assez clair suivant :

```
line 5, in moyenne return somme / len(liste) ZeroDivisionError: division by zero
```

Solution : Il faut ajouter une condition dans le programme car, dans le cas où la liste ne contient pas d'éléments, on ne peut pas diviser par 0.

On ajoute donc les trois lignes ci-contre au début de la fonction pour la corriger.

Puis on teste la fonction avec cette nouvelle définition :

moyenne([]) `# Renvoie maintenant None et plus une erreur`

```
if len(liste) == 0 :     # Si la liste est vide
    return None          # Alors on renvoie None
else :                   # Sinon
```

La fonction moyenne précédente n'est pas encore parfaite. Par exemple si, par erreur, on lui envoie une liste de lettres, elle renverra un message d'erreur. On peut la corriger comme suit :

```
def moyenne(liste) :
    if len(liste) == 0 :    # Cas d'une liste vide
        return None         # renvoie None
    else :                  # Cas d'une liste non vide
        somme = 0
        for val in liste :
            if type(val) == int or type(val) == float :  # Cas d'un nombre dans la liste
                somme = somme + val                      # On calcule la moyenne
            else :              # Cas d'un autre type de donnée dans la liste
                return None     # On renvoie None car on ne peut pas calculer de moyenne
    return somme / len(liste)  # Si le calcul est mené à son terme, on renvoie la moyenne
```

??? Expliquer dans un court paragraphe les problèmes posés par la fonction initiale et comment ils ont été corrigés.

Applications :

A.

Voici deux exemples de fonctions qui contiennent des bugs.

```
def val_frac(a, b) :
    return a / b
```

```
def sqrt(val) :
    return val ** 0.5   # soit val à la puissance 0.5
                        # c'est-à-dire sa racine carrée
```

1. Proposer des couples a, b et des valeurs de val permettant de tester les fonctions proposées ci-dessus.
2. Déterminer les situations qui pourraient poser problème pour ces fonctions et proposer des modifications à apporter afin d'éviter les bugs.

B. Le programme suivant est écrit avec Python. Son but est de calculer et d'afficher le carré des nombres entiers entre 1 et 20.

```
1  n=1
2  while n<21:
3      print(n*n)
4  n=n+1
```

1. Tester le programme. Que se passe-t-il ?
2. Corriger le programme.