Suivi de l'exécution d'un programme

1 Trace d'exécution

Méthode Savoir exécuter « mentalement » un programme est une compétence de base en programmation. Cela te permet non seulement de comprendre le code qu'un autre développeur a écrit et que tu souhaites réutiliser, mais aussi, lorsque tu débogues ton propre programme, de pouvoir trouver les erreurs commises en lisant le code.

Pour suivre l'exécution pas à pas d'un programme, on peut réaliser sa **trace d'exécution**. Cela consiste à prendre une « photo » de toutes les variables de ce programme aux instants suivants :

- au début ;
- à chaque instruction modifiant l'état de l'interprète ;
- à la fin.

On obtient ainsi la séquence des états correspondant à chaque instructions exécutées. On la présente généralement dans un tableau où chaque ligne contient un numéro désignant l'instruction exécutée, l'état de l'interprète après exécution de l'instruction et les éventuelles interactions avec l'utilisateur au cours de l'exécution.

Par exemple, le programme ci-dessous

```
  # Calcul de la division euclidienne d'un entier
1 nombre = int(input("Entrez le nombre entier à diviser : "))
2 diviseur = int(input("Entrez le diviseur : "))
3 quotient = nombre // diviseur
4 reste = nombre % diviseur
5 print(nombre, "=", diviseur, "x", quotient, "+", reste)
```

produira la trace d'exécution suivante dans le cas où l'utilisateur ait saisi `12` ; `Entrée` puis `5` ; `Entrée` :

Ligne	État	Entrées/Sorties
1	`nombre` `12`	*Entrées :* 12 *Sorties :* Entrez le nombre entier à diviser :
2	`nombre` `12` `diviseur` `5`	*Entrées :* 5 *Sorties :* Entrez le diviseur :

3	`nombre` `12`
	`diviseur` `5`
	`quotient` `2`	

4	`nombre` `12`
	`diviseur` `5`
	`quotient` `2`
	`reste` `2`	

5	`nombre` `12`
	`diviseur` `5`
	`quotient` `2`
	`reste` `2`
		Sorties : $12 = 5 \times 2 + 2$

À toi de jouer Réalise la trace d'exécution du programme ci-dessous en choisissant les entrées saisies par l'utilisateur puis modifie le programme en y insérant des instructions `print` te permettant d'observer les valeurs des variables au cours de l'exécution du programme.

```
  # Calcul de l'âge
1 saisie = input("Entrez votre année de naissance : ")
2 annee = int(saisie)
3 age = 2048 - annee
4 print("Vous aurez", age, "ans en 2048.")
```

2 Débogage d'un programme

« Aussi brillant en programmation que tu sois,
Des erreurs immanquablement tu commettras. »

- Ton prof de NSI

Lors de l'écriture de programme, la plus grande partie du temps est, en général, consacrée à l'identification et à la correction des erreurs ou *bogues* ; c'est ce qu'on appelle le *débogage* du programme.

On distingue trois grands types d'erreur :

- les *erreurs de syntaxe* : elles apparaissent lorsque le code que tu as tapé ne respecte pas les régles « grammaticales » du langage. L'interprète ne comprend alors pas l'instruction et il stoppe immédiatement l'exécution du programme. Ce sont les erreurs les plus facile à identifier puisque l'interprète t'indique précisement où est-ce que le code pose problème.
- *erreurs d'exécution* : le programme se bloque ou s'arrête en cours de d'exécution ; le code est syntaxiquement correct, l'interprète peut le lire et commencer l'exécution du programme jusqu'à rencontrer une erreur de programmation (type de variable incorrecte, division par zéro, lecture d'un fichier inexistant...).
- les *erreurs sémantiques* : le code est syntaxiquement correct et le programme s'exécute sans aucune erreur apparente (du point de vue de la machine). Néanmoins le programme ne réalise pas le résultat que tu souhaite obtenir ; le problème est que, ce que tu lui as dit de faire, ne correspond pas à ce que tu veux qu'il fasse. Ce sont les erreurs les plus difficile à identifier puisse que l'interprète est incapable de les détecter. Il faut alors analyser patiemment ce qui sort de la machine et se représenter une par une les opérations qu'elle a effectuées, à la suite de chaque instruction.

À faire en binôme Reprends le précédent programme et insères-y, tour à tour, une erreur de chaque type. À chaque erreur introduite, passe le code à ton camarade, sans lui indiquer le type d'erreur à trouver, qui devra analyser et corriger le programme pour obtenir le résultat voulu.

Plus Ultra Recommence l'exercice précédent avec ton propre programme. Le correcteur du programme devra, en plus des erreurs, trouver à quoi sert le programme, uniquement en lisant le code.