Exercices débranchés

Structures conditionnelles et boucles

NSI – Première – Séquence 2

1 Prédire l'exécution d'une structure conditionnelle

Exercice 1: Prédire l'exécution

On exécute le programme suivant avec `note = 13` :

```
1  note = 13
2
3  if note < 10:
4      print("Insuffisant")
5  elif note < 14:
6      print("Passable")
7  elif note < 16:
8      print("Assez bien")
9  else:
10     print("Bien")
```

1. Sans exécuter ce programme sur ordinateur, indiquer ce qu'il affiche pour `note = 13`. Justifier en indiquant quelle condition est testée, et si elle est vraie ou fausse, à chaque étape.
2. Reprendre la question précédente pour `note = 16`.
3. Reprendre la question précédente pour `note = 8`.

2 Écrire une suite d'instructions conditionnelles

Exercice 2: Réduction en magasin

Un magasin applique une réduction sur le montant d'un panier, selon la règle suivante :
- si le montant est strictement supérieur à 100 euros, la réduction est de 10% ;
- sinon, si le montant est strictement supérieur à 50 euros, la réduction est de 5% ;
- sinon, il n'y a aucune réduction.

Écrire un programme Python qui :

1. demande le montant du panier à l'utilisateur (on suppose que l'entrée est un nombre valide) ;
2. calcule le montant de la réduction selon la règle ci-dessus ;
3. affiche le montant final à payer (montant initial moins la réduction).

3 Trouver et corriger l'erreur

Exercice 3: Une branche qui ne s'affiche jamais

Un élève veut afficher une appréciation selon une note sur 20. Il écrit :

```
if note >= 10:
    print("Passable")
elif note >= 16:
    print("Très bien")
else:
    print("Insuffisant")
```

Il teste son programme avec `note = 18` et s'attend à voir `Très bien` s'afficher, mais c'est `Passable` qui apparaît.

1. Expliquer précisément pourquoi le message `Très bien` ne peut **jamais** s'afficher, quelle que soit la valeur de `note`.
2. Corriger le programme pour qu'il se comporte comme prévu.

4 Boucle bornée : prédire et tracer

Exercice 4: Tracer une boucle for

On exécute le programme suivant :

```
total = 0
for i in range(1, 6):
    total = total + i
print(total)
```

1. Recopier et compléter le tableau ci-dessous, en indiquant la valeur de `i` et de `total` à chaque tour de boucle.

Tour	i	total
avant la boucle	—	
1		
2		
3		
4		
5		

2. Que fait ce programme, en une phrase ? Que va-t-il afficher ?

5 Boucle non bornée : prédire et tracer

Exercice 5: Tracer une boucle while

On exécute le programme suivant :

```
1 n = 50
2 compteur = 0
3 while n > 1:
4     n = n // 2
5     compteur = compteur + 1
6 print(compteur)
```

(rappel : `//` est la division entière, qui arrondit le résultat à l'entier inférieur)

1. Recopier et compléter le tableau ci-dessous jusqu'à la fin de la boucle.

Tour	n avant	`n > 1` ?	n après / compteur
1	50		
2			
3			
4			
5			

2. Quelle est la valeur affichée par ce programme ?
3. Pourquoi cette boucle est-elle non bornée, alors que le nombre de tours semble ici assez facile à deviner à l'avance ?

6 Exercice de synthèse : `for` ou `while` ?

Exercice 6: Choisir la bonne boucle

Pour chacune des situations suivantes, indiquer si une boucle `for` ou une boucle `while` est la plus adaptée, **justifier ce choix en une phrase**, puis écrire le programme correspondant.

1. Afficher tous les multiples de 7 de 7 à 70 (inclus).
2. Un nombre secret est fixé dans le code (par exemple `7`). Demander à l'utilisateur de proposer un nombre, en répétant la demande tant que sa proposition est incorrecte, puis afficher le nombre d'essais qui ont été nécessaires.
3. Demander un âge à l'utilisateur, et redemander tant que la valeur saisie est négative.