Mise au point et débogage

Langages et programmation

Exercice 1 *QCM*

Pour chaque question, une seule réponse est correcte.

1. Quelle instruction Python permet de vérifier une condition et d'interrompre le programme si elle est fausse ?

   A. `try condition`

   B. `raise condition`

   C. `assert condition`

   D. `if not condition: exit()`

2. Pourquoi la condition `while lst[i] < 100 and i < len(lst)` est-elle dangereuse ?

   A. `len(lst)` recalcule la longueur à chaque itération

   B. La condition devrait utiliser `or` au lieu de `and`

   C. Il manque une instruction `break`

   D. Python accède à `lst[i]` avant de vérifier que `i` est un indice valide

3. Que vaut `0.1 + 0.2 == 0.3` en Python ?

   A. `True`

   B. `False`

   C. `None`

   D. Une erreur est levée

4. Quel est le résultat de ce code ?

```
x = [1, 2, 3]
y = x
y.append(4)
print(len(x))
```

   A. 3

   B. 4

   C. Une erreur `AttributeError`

   D. Une erreur `TypeError`

Exercice 2 *Identifier les erreurs*

Pour chaque extrait de code, identifier le type d'erreur que Python va lever et expliquer pourquoi.

a)

```
def saluer(nom)
    print(f"Bonjour {nom}")
```

b)

```
notes = [15, 12, 18]
print(notes[3])
```

c)

```
age = input("Votre âge : ")
annee_naissance = 2026 - age
```

d)

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

print(diviser(10, 0))
```

e)

```
dico = {"a": 1, "b": 2}
print(dico["c"])
```

Exercice 3 *Tracer l'exécution avec `print` (exercice guidé)*

La fonction suivante est censée renvoyer l'indice du maximum d'une liste, et l'on veut vérifier son comportement sur des cas limites.

```
def indice_max(L):
    i_max = 0
    for i in range(len(L)):
        if L[i] > L[i_max]:
            i_max = i
    return i_max
```

a) Testez `indice_max([3, 7, 2, 9, 1])`. Quel résultat obtenez-vous ? Est-il correct ?

b) Testez `indice_max([5, 5, 5])`. Quel résultat obtenez-vous ? Est-ce le comportement attendu ?

c) Ajoutez des `print` dans la boucle pour tracer les valeurs de `i`, `L[i]`, `i_max` et `L[i_max]` à chaque itération. Exécutez sur `[3, 7, 2, 9, 1]` et copiez la trace.

d) Testez maintenant `indice_max([9, 1, 9, 1])`. Quel indice est renvoyé ? Si l'on souhaite renvoyer le **dernier** indice du maximum (et non le premier), quelle modification faut-il apporter ?

Exercice 4 *Les pièges des flottants*

a) Exécutez les expressions suivantes en Python et notez les résultats :

```
0.1 + 0.2
0.1 + 0.2 == 0.3
1.0 + 1e-16 == 1.0
```

Expliquez chaque résultat.

b) La fonction suivante est censée vérifier si un triangle est rectangle en utilisant le théorème de Pythagore. Pourquoi peut-elle échouer avec certaines valeurs ?

```
def est_rectangle(a, b, c):
    """Vérifie si un triangle de côtés a, b, c est rectangle."""
    cotes = sorted([a, b, c])
    return cotes[0]**2 + cotes[1]**2 == cotes[2]**2
```

c) Réécrivez la fonction en utilisant une comparaison avec tolérance. On utilisera `abs(x - y) < epsilon` avec `epsilon = 1e-9`.

d) Testez les deux versions avec le triangle de côtés 0,5, 1,2 et 1,3. Que constatez-vous ?

Exercice 5 *Écrire des assertions et des tests*

a) Écrire des assertions de précondition pour la fonction `moyenne(L)` : la liste ne doit pas être vide et tous ses éléments doivent être des nombres.

b) Compléter les docstrings de la fonction suivante avec des doctests couvrant les cas normaux et les cas limites :

```
def est_palindrome(s):
    """
    Renvoie True si la chaîne s est un palindrome, False sinon.

    >>> # Écrire les doctests ici
    """
    return s == s[::-1]
```

c) Pour la fonction `tri_insertion(L)`, proposer au moins quatre jeux de tests pertinents (liste quelconque, liste vide, liste déjà triée, liste avec doublons) et justifier le choix de chacun.

Exercice 6 *Chasse aux bugs – les six causes classiques*

Chacun des programmes suivants contient un bug relevant d'une cause classique. Identifier l'erreur, nommer sa cause et proposer une correction.

a) Indentation incorrecte :

```
def compter_voyelles(mot):
    voyelles = "aeiouy"
    compteur = 0
    for i in range(len(mot)):
        if mot[i] in voyelles:
            compteur += 1
        return compteur
```

b) Effet de bord :

```python
def inverser_liste(L):
    resultat = L
    resultat.reverse()
    return resultat

ma_liste = [1, 2, 3]
inversee = inverser_liste(ma_liste)
print(ma_liste)
```

c) Choix d'inégalité :

```python
def somme_entiers(n):
    s = 0
    i = 1
    while i < n:
        s += i
        i += 1
    return s

print(somme_entiers(5))  # Attendu : 15, obtenu : ?
```

d) Conditionnelle non exhaustive :

```python
def mention(moyenne):
    if moyenne >= 16:
        return "Très bien"
    elif moyenne >= 14:
        return "Bien"
    elif moyenne >= 12:
        return "Assez bien"

print(mention(10))  # Que se passe-t-il ?
```

e) Mauvais nommage :

```python
list = [4, 7, 2, 9]
somme = sum(list)
list = list + [5]     # Que se passe-t-il ici ?
```

f) Comparaison de flottants :

```python
x = 0.1 + 0.1 + 0.1
if x == 0.3:
    print("Égal")
else:
    print("Différent")  # Quel message s'affiche ?
```

Exercice 7 *Gestion des exceptions*

a) Écrire une fonction `saisie_entier()` qui demande à l'utilisateur de saisir un entier et redemande tant que la saisie n'est pas valide. Utiliser `try/except`.

b) Écrire une fonction `lire_fichier(chemin)` qui renvoie le contenu d'un fichier sous forme de chaîne, ou `"Fichier introuvable"` si le fichier n'existe pas.

c) Définir une exception personnalisée `NoteInvalideError` et une fonction `ajouter_note(notes, n)` qui lève cette exception si `n` n'est pas entre 0 et 20.

Exercice 8 *Synthèse – débogage complet d'un programme*

Le programme suivant est censé lire une liste de notes séparées par des virgules, calculer la moyenne et afficher le résultat avec une mention. Il contient **deux bugs** et deux situations d'erreur non gérées. Les trouver, les corriger et ajouter une gestion d'erreur appropriée.

```
def calculer_moyenne(chaine):
    notes = chaine.split(",")
    somme = 0
    for note in notes:
        somme += note
    moyenne = somme / len(notes)
    if moyenne >= 10:
        mention = "admis"
    elif moyenne >= 8:
        mention = "rattrapage"
    return f"Moyenne : {moyenne}/20 ({mention})"

resultat = calculer_moyenne("12,15,8,14,9")
print(resultat)
```