

Initiation à l'algorithmique et à la programmation

TD/TP 4 – Boucles et itérations

Licence 3 – INALCO – 2026–2027

Objectifs de la séance

Suivre une boucle, utiliser `for`, `range()` et `while`, compter des caractères et réutiliser le résultat dans une fonction.

Avant de commencer

Créez un dossier TP4. Prédisez avant d'exécuter. Enregistrez les exercices 3 et 4 dans `tp4.py`, avec les définitions avant les appels. Durée indicative : 60 à 75 minutes, hors défi.

Méthode : comprendre → prédire → programmer → tester → expliquer.

Difficulté : * application directe ** plusieurs notions *** défi.

■ TD – Lire et déboguer

Exercice 1 – Prévoir les sorties et suivre les variables

[*]

Pour chaque extrait indépendant, notez les lignes affichées et le nombre d'itérations. Vérifiez ensuite avec Python.

A – Parcourir un texte

```
1 for caractere in "a b":
2     print "[" + caractere + "]"
3 print("Fin")
```

B – Parcourir des entiers

```
1 for i in range(2, 8, 2):
2     print(i)
```

C – Répéter sous condition

```
1 x = 5
2 while x > 2:
3     print(x)
4     x -= 1
5 print("Fin", x)
```

Pour C, faites un tableau avec `x` avant le test, le résultat de `x > 2` et `x` après chaque tour. Incluez le dernier test, qui arrête la boucle.

Puis : que se passe-t-il dans A avec `" "`, dans B avec `range(2, 2)` et dans C avec `x = 2` au départ ?

Exercice 2 – Deux erreurs à corriger

[**]

2.1 – On veut afficher les entiers de 1 à 3. Expliquez pourquoi cette boucle ne se termine pas. Corrigez-la avant de l'exécuter.

```
1 i = 1
2 while i <= 3:
3     print(i)
4 i += 1
```

2.2 – Cette fonction doit compter tous les "a". Elle renvoie pourtant 0 pour "banane". Repérez et corrigez l'erreur. Testez aussi une chaîne vide.

```
1 def compter_a(texte):
2     nombre = 0
3     for caractere in texte:
4         if caractere == "a":
5             nombre += 1
6     return nombre
```

Question : pourquoi l'initialisation doit-elle précéder la boucle et le `return` venir après ?

■ TP – Boucles, fonctions et textes

Exercice 3 – Compter la ponctuation

[**]

Dans `tp4.py`, écrivez `compter_ponctuation(texte)` qui renvoie le nombre de caractères appartenant à `".,;:!?"`.

1. Initialisez un compteur à zéro avant la boucle.
2. Parcourez les caractères avec `for`. Utilisez un `if` pour augmenter le compteur uniquement si le caractère appartient à `".,;:!?"`.
3. Renvoyez le total après la boucle. Ajoutez une docstring courte.

Indice : `caractere in ".,;:!?"` produit un booléen.

Test de validation

Prédisez, puis vérifiez ces résultats.

Appel	Valeur attendue
<code>compter_ponctuation("Bonjour, Ada !")</code>	2
<code>compter_ponctuation("Vraiment?! Oui.")</code>	3
<code>compter_ponctuation("bonjour")</code>	0
<code>compter_ponctuation("")</code>	0

Exercice 4 – Analyser plusieurs saisies

[**]

Après la définition de votre fonction, écrivez un programme qui demande des textes jusqu’à la saisie du mot `fin`.

1. Demandez un premier texte et retirez ses espaces aux extrémités avec `strip()`.
2. Tant que `texte.lower() != "fin"`, affichez le nombre de signes de ponctuation en appelant votre fonction, puis demandez un nouveau texte.
3. Ajoutez un compteur du nombre de textes traités. Affichez-le après la boucle. Le mot d’arrêt ne doit être ni analysé ni compté.

Tests : saisissez `Bonjour !`, puis `Oui.`, puis `fin`. Les nombres affichés doivent être 1, 1, puis un total de 2 textes.

Testez aussi `FIN` dès la première saisie : le total doit être 0. Dans cet exercice, une saisie vide compte comme un texte traité et contient zéro signe de ponctuation.

Exercice 5 – Pour aller plus loin : inverser un texte

[***]

Écrivez `inverser_texte(texte)` avec un accumulateur initialisé à `""`. À chaque tour, ajoutez le caractère **devant** le résultat déjà construit. Utilisez une boucle `for`, sans `::-1` ni `reversed()`.

Tests : `"chat"` donne `"tahc"`, `"a b"` donne `"b a"`, `""` donne `""`.

■ Bilan

- ☐ Mon fichier `tp4.py` est enregistré dans TP4.
- ☐ Je distingue les valeurs parcourues du nombre d’itérations.
- ☐ Je peux expliquer ce qui permet à ma boucle `while` de s’arrêter.
- ☐ J’ai testé zéro itération, un texte sans ponctuation et une entrée vide.

FIN DU TP 4 – sauvegardez votre travail !