

Initiation à l’algorithmique et à la programmation

TD/TP 2 – Opérateurs et conditions

Licence 3 – INALCO – 2026–2027

Objectifs de la séance

Déboguer, prévoir les sorties, suivre les branches de `if/elif/else` et écrire des conditions simples et combinées.

Avant de commencer

Créez un dossier TP2. Prédisez avant d’exécuter. Enregistrez les exercices de programmation dans des fichiers séparés.

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

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

■ TD – Révision et débogage

Exercice 1 – Trois familles d’erreurs

[*]

Pour chaque extrait, indiquez si l’erreur est de syntaxe, d’exécution ou de logique. Corrigez-la puis vérifiez votre programme. Chaque extrait est indépendant.

1.1 – Salutation attendue : afficher Bonjour, Ada.

```
1 prenom = "Ada"
2 print("Bonjour, " + prenom
```

1.2 – Âge attendu : si l’utilisateur saisit 20, afficher 21.

```
1 age = input("Votre age : ")
2 print(age + 1)
```

1.3 – Prix attendu : appliquer une réduction de 20% à un prix de 100 euros ; résultat attendu : 80.0.

```
1 prix = 100
2 reduction = 20
3 prix_final = prix - reduction / 100
4 print(prix_final)
```

1.4 – Sortie attendue : afficher Alice Martin avec un espace.

```
1 prenom = "Alice"
2 nom = "Martin"
3 print(prenom + nom)
```

Exercice 2 – Prévoir des expressions

[*]

Sans exécuter le code, notez chaque ligne affichée et la valeur finale des variables. Vérifiez ensuite avec Python. Expliquez toute différence.

```
1.
1 x = 2
2 x += 3
3 x *= 2
```

```

4 print(x)
5 print(x == 10)

```

```

1 mot = "chat"
2 print(len(mot) >= 4)
3 print(mot == "Chat")
4 print(len(mot) == 4 and mot != "chien")

```

3.

```

1 a = 7
2 b = 2
3 print(a / b)
4 print(a // b)
5 print(a % b)

```

■ TD – Lire et comprendre les conditions

Exercice 3 – Tracer des branches

[**]

Pour chaque code, faites un tableau avec les valeurs testées, le résultat de chaque condition évaluée et les lignes affichées. Ne supposez pas que tous les tests sont exécutés.

1. Prédisez la sortie pour `mot = "chat"`, puis `"chien"`, puis `"rat"`.

```

1 mot = "chat"
2 if len(mot) < 4:
3     print("court")
4 elif len(mot) == 4:
5     print("quatre lettres")
6 else:
7     print("long")
8 print("analyse terminée")

```

2. Prédisez la sortie pour `n = 6`, puis `n = 7`. Pourquoi les deux programmes ci-dessous ne sont-ils pas équivalents ?

```

1 # Programme A
2 n = 6
3 if n > 0:
4     print("positif")
5 if n % 2 == 0:
6     print("pair")
7
8 # Programme B
9 n = 6
10 if n > 0:
11     print("positif")
12 elif n % 2 == 0:
13     print("pair")

```

3. Prédisez la sortie pour `mot = "chat"`, `"chien"` et `"rat"`. Associez chaque `else` à son `if` grâce à l'indentation.

```

1 mot = "chat"
2 if len(mot) >= 4:
3     if mot == "chat":
4         print("chat reconnu")
5     else:
6         print("autre mot long")
7 else:
8     print("mot court")

```

■ TP – Corriger et programmer

Exercice 4 – Corriger des conditions

[**]

Le programme suivant doit afficher `Mot reconnu` pour `banane`, `Autre mot` sinon. Repérez toutes les erreurs puis testez avec `banane` et `poire`.

```
1 mot = input("Un mot : ")
2 if mot = "banane"
3     printf("Mot reconnu")
4 else:
5 print("Autre mot")
```

Puis : que faudrait-il modifier pour reconnaître aussi `Banane` ? Essayez une condition avec `or`.

Exercice 5 – Écrire des programmes

[**]

Pour chaque programme, testez les cas limites indiqués et expliquez pourquoi vous les avez choisis.

1. **Longueur d’un mot.** Demandez un mot avec `input()`, calculez sa longueur avec `len()` et affichez `court` si elle est strictement inférieure à 5, `moyen` si elle est entre 5 et 8 inclus, `long` sinon. Testez des mots de 4, 5, 8 et 9 lettres.
2. **Identifier une salutation.** Demandez un mot. Affichez `salutation du matin` pour `bonjour`, `salutation du soir` pour `bonsoir`, `salutation inconnue` sinon. Testez les trois issues, puis observez ce qui se passe avec `Bonjour`.
3. **Nombre de caractères attendu.** Demandez le nombre de caractères annoncé par l’utilisateur avec `int(input(...))`, puis demandez un mot. Affichez `correct` si la longueur annoncée est égale à `len(mot)`, `trop petit` si elle est inférieure, `trop grand` sinon. Testez les trois cas.

Exercice 6 – Pour aller plus loin

[***]

Demandez un mot et une catégorie annoncée (`court` ou `long`). On considère un mot `long` à partir de 6 caractères. Affichez `oui` si la catégorie correspond au mot et `non` sinon. Que choisir pour une catégorie différente de `court` et `long` ? Gérez ce troisième cas explicitement.

■ Bilan

- ☐ Mes programmes sont enregistrés dans TP2.
- ☐ J’ai prédit les sorties avant de les vérifier.
- ☐ J’ai testé des valeurs limites et toutes les branches.
- ☐ Je peux expliquer le rôle de l’indentation et distinguer `=` de `==`.

En cas de blocage

Relisez le message d’erreur ; repérez la ligne concernée ; affichez les valeurs et leurs types ; comparez avec le cours ; demandez un indice si nécessaire.

FIN DU TP 2 – sauvegardez votre travail !