

# Initiation à l'algorithmique et à la programmation

TD/TP de révision – Comprendre, compléter, programmer

Licence 3 – INALCO – 2026–2027

## Objectifs de la séance

Reprendre les bases des semaines 1 à 3, puis découvrir ou revoir les boucles à l'aide d'exemples guidés. L'objectif est de pouvoir expliquer son programme, pas seulement de le faire fonctionner.

## Avant de commencer

Créez un dossier **Revisions** et un fichier **revision.py**. Notez vos réponses sur une feuille ou dans un fichier texte. Travaillez par petites étapes : testez un bloc avant d'ajouter le suivant.

**Parcours conseillé** : pages 1 à 4 pour consolider les notions enseignées. Les boucles n'ayant pas pu être travaillées normalement en cours, la page 5 constitue une reprise guidée. La page 6 est un prolongement à faire ensuite, à votre rythme.

## ■ 1 – Variables, types et saisie

### Rappel expliqué

Une variable permet de retrouver une valeur grâce à un nom. Les guillemets indiquent du texte : 12 et "12" n'ont pas le même type.

```
1 nombre = 12                # entier : int
2 texte = "12"               # chaîne : str
3 print(nombre + 3)          # affiche 15
4 print(texte + "3")         # affiche 123
```

Avec des nombres, + additionne. Avec des chaînes, il les assemble. `input()` renvoie toujours une chaîne : pour calculer sur une saisie numérique, on la convertit, par exemple avec `int()`.

### Exercice 1 – Prévoir, puis vérifier

[Application directe]

**A. Sans Python**, complétez le tableau. Exécutez ensuite chaque expression avec `print()` et vérifiez le type avec `type()`.

| Expression                | Valeur prévue | Type prévu |
|---------------------------|---------------|------------|
| "7" + "2"                 |               |            |
| <code>int("7") + 2</code> |               |            |
| <code>7 / 2</code>        |               |            |
| <code>7 // 2</code>       |               |            |
| <code>7 % 2</code>        |               |            |
| <code>7 &gt; 2</code>     |               |            |

**Mémo** : / divise, // donne le quotient entier et % le reste. Une comparaison renvoie un booléen : `True` ou `False`.

**B. Complétez**, puis testez avec une saisie de 20. On suppose que la saisie représente bien un entier.

```
1 age_texte = input("Votre age : ")
2 age = ...           # convertir la saisie
3 print(age + 1)
```

**À expliquer** : pourquoi ne peut-on pas calculer directement `age_texte + 1` ?

## ■ 2 – Chaînes de caractères

### Observer une transformation

Les espaces font partie du texte. Les crochets ci-dessous permettent de les voir.

```
1 texte = "  Bonjour  "
2 nettoye = texte.strip()
3 print "[" + nettoye + "]" # [Bonjour]
4 print "[" + texte + "]"  # [ Bonjour ]
```

`strip()` retire les espaces blancs aux extrémités et renvoie une nouvelle chaîne. L'ancienne valeur ne change pas automatiquement. C'est aussi le cas de `lower()`, `upper()` et `replace()`.

### Exercice 2 – Manipuler un texte pas à pas

[Guidé]

**A. Prédisez chaque résultat**, puis vérifiez. Dans cet extrait, quelle instruction ne conserve pas son résultat ?

```
1 texte = "  Un CHAT  "
2 texte.lower()
3 print "[" + texte + "]"
4 mot = texte.strip()
5 print(mot)
6 print(mot.lower())
7 print(mot.replace(" ", ""))
```

**B. Complétez** les deux transformations, en les gardant sur des lignes séparées.

```
1 texte = input("Un texte : ")
2 sans_espaces_externes = ...
3 texte_nettoye = ...
4 print "[" + texte_nettoye + "]"
```

#### Test de validation

Avec la saisie " Un CHAT ", vous devez obtenir [un chat]. L'espace entre les mots reste présent. Une saisie vide doit produire [].

### Repérer les positions

| Caractère | b | o | n | j | o | u | r |
|-----------|---|---|---|---|---|---|---|
| Indice    | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

`mot[0]` désigne le premier caractère. `mot[debut:fin]` extrait une partie du mot, en excluant l'indice `fin`.

**C. Avec** `mot = "bonjour"`, prévoyez `mot[0]`, `mot[-1]`, `mot[:3]` et `mot[3:]`. Puis écrivez une expression qui extrait les deux derniers caractères.

**D. Comparez** `""[:3]` et `""[0]` avec Python. Laquelle fonctionne ? Expliquez la différence entre extraire une tranche et accéder à un caractère.

### ■ 3 – Conditions et choix

#### Rappel expliqué

= affecte une valeur. == compare deux valeurs. Dans une chaîne `if/elif/else`, Python exécute la première branche dont la condition est vraie, ou le `else` si aucune ne l'est.

```

1 mot = "chat"
2 if len(mot) < 5:
3     print("court")
4 else:
5     print("au moins 5 caracteres")
6 print("Fin")

```

Ici, `len(mot)` vaut 4. Le test est vrai : Python affiche `court`, puis `Fin`. L'indentation indique les instructions de chaque branche.

#### Exercice 3 – Construire un classement

[Guidé, puis autonome]

On veut afficher `court` pour moins de 5 caractères, `moyen` de 5 à 8 inclus, `long` au-delà de 8. Les espaces comptent.

##### A. Complétez d'abord sans programmer.

| Longueur | Inférieure à 5 ? | Entre 5 et 8 ? | Catégorie |
|----------|------------------|----------------|-----------|
| 4        | oui              | non            | court     |
| 5        |                  |                |           |
| 8        |                  |                |           |
| 9        |                  |                |           |

##### B. Complétez le code, puis testez avec des textes de 4, 5, 8 et 9 caractères.

```

1 texte = input("Un texte : ")
2 longueur = len(texte)
3 if longueur < ...:
4     print("court")
5 elif longueur <= ...:
6     print("moyen")
7 else:
8     print(...)

```

**À expliquer :** pourquoi n'a-t-on pas besoin de répéter `longueur >= 5` dans le `elif` ?

**C. À vous de programmer.** Demandez un mot, retirez les espaces aux extrémités et mettez-le en minuscules. Affichez `salutation` s'il vaut `bonjour` ou `bonsoir`, et `autre mot` sinon. Utilisez `or`.

#### En cas de blocage

Comparez la variable à chaque valeur : `mot == "bonjour" or mot == "bonsoir"`. Un test comme `mot == "bonjour" or "bonsoir"` ne réalise pas les deux comparaisons.

**Vos tests :** " Bonjour ", "BONSOIR", "salut" et "". Notez le résultat attendu avant chaque exécution.

## ■ 4 – Fonctions et résultats

### Observer le trajet d’une valeur

```

1 def doubler(nombre):
2     resultat = nombre * 2
3     return resultat
4
5 valeur = doubler(3)
6 print(valeur + 1)           # affiche 7

```

1. L’appel fournit l’argument 3 au paramètre `nombre`.
2. La fonction calcule 6 et le renvoie avec `return`.
3. Le programme stocke ce résultat dans `valeur`, puis affiche 7.

Définir la fonction ne l’exécute pas. Les variables `nombre` et `resultat` sont locales à la fonction.

### Exercice 4 – Afficher ou renvoyer ?

[Comparer et corriger]

#### A. Prédisez les lignes affichées, puis exécutez.

```

1 def doubler(nombre):
2     print(nombre * 2)
3
4 valeur = doubler(3)
5 print(valeur)

```

**B. Corrigez** pour que la fonction renvoie le double. Après l’appel, affichez `valeur + 1` : le résultat doit être 7.

**À retenir :** `print()` affiche une information. `return` fournit une valeur au programme appelant et termine l’appel. Sans `return` exécuté, la fonction renvoie `None`.

### Exercice 5 – Écrire et tester vos fonctions

[Guidé, puis autonome]

#### A. Complétez dans `revision.py` :

```

1 def nettoyer(texte):
2     """Retire les espaces externes et met en minuscules."""
3     sans_espaces_externes = ...
4     return ...

```

**B. Sans modèle à compléter**, écrivez `categorie_texte(texte)` qui renvoie "court", "moyen" ou "long" selon les seuils de la page 3. Reprenez vos conditions, en remplaçant les affichages par des valeurs de retour.

#### Test de validation

`nettoyer(" CHAT ")` doit renvoyer "chat".  
`categorie_texte("chat")` doit renvoyer "court".  
`categorie_texte("bonjour")` doit renvoyer "moyen".  
 Testez aussi les longueurs 5, 8 et 9, ainsi qu’une chaîne vide.

**C. Combinez les fonctions :** demandez un texte, nettoyez-le avec `nettoyer()`, puis passez le résultat à `categorie_texte()`. Affichez enfin le texte nettoyé et sa catégorie.

## ■ 5 – Les boucles : reprise guidée

Cette partie peut être une découverte. Prenez le temps de suivre chaque tour avant d’essayer d’écrire une boucle seul. Une itération correspond à une exécution du bloc répété.

**for** : parcourir les caractères

```
1 nombre = 0
2 for caractere in "a b":
3     if caractere == " ":
4         nombre += 1
5 print(nombre)
```

Le compteur commence à zéro. Le **for** fournit les caractères dans l’ordre. Le **if** décide si le compteur augmente.

### Exercice 6 – Suivre, compléter et tester

[Guidé]

**A. Complétez la trace** du programme ci-dessus.

| Tour | Caractère | Égal à l’espace ? | Compteur après le tour |
|------|-----------|-------------------|------------------------|
| 1    | "a"       | non               | 0                      |
| 2    | " "       |                   |                        |
| 3    | "b"       |                   |                        |

**B. Complétez la fonction** pour compter les points. **return** vient après la boucle afin de renvoyer le compte complet.

```
1 def compter_points(texte):
2     nombre = ...
3     for caractere in ...:
4         if ...:
5             nombre += ...
6     return nombre
```

**Tests** : "Oui. Non." donne 2, "bonjour" donne 0 et "" donne 0. Expliquez pourquoi le dernier test fonctionne même sans aucun tour.

**range** : répéter avec des entiers

```
1 for i in range(1, 4):
2     print(i)
```

Le programme affiche 1, 2, 3, sur trois lignes. La borne finale est exclue. **C. Modifiez-le** pour afficher les entiers de 1 à 5 inclus.

**while** : répéter tant qu’une condition est vraie

```
1 mot = input("Un mot (fin pour arreter) : ")
2 while mot != "fin":
3     print(len(mot))
4     mot = input("Un autre mot : ")
```

**D. Simulez les saisies** chat, oui, fin. Notez les longueurs affichées. Quelle ligne permet de changer la valeur testée ? Que se passe-t-il si l’on saisit **fin** immédiatement ?

## ■ 6 – Petit projet : une fiche de texte

À faire après les parties précédentes. Réutilisez les fonctions `nettoyer()`, `categorie_texte()` et `compter_points()`. Le but est de construire un programme étape par étape, en testant chaque version.

### Étape 1 – Une seule saisie

Reprenez le programme de la page 4. Ajoutez l’affichage de la longueur et du nombre de points. **Ne mettez pas encore de boucle.**

#### Test de validation

Pour la saisie " Bonjour. ", vérifiez les quatre informations :

|                  |          |
|------------------|----------|
| Texte nettoyé    | bonjour. |
| Longueur         | 8        |
| Catégorie        | moyen    |
| Nombre de points | 1        |

### Étape 2 – Plusieurs saisies

Quand la première version fonctionne, utilisez ce squelette. Les points de suspension sont des zones à remplacer, pas du code terminé.

```

1 texte = nettoyer(input("Un texte (fin pour arreter) : "))
2 while texte != "fin":
3     ... # reprendre les affichages de l'étape 1
4     texte = ... # demander et nettoyer la saisie suivante
5 print("Termine")

```

Testez "CHAT", "Bonjour.", puis " FIN ". Le programme doit afficher deux fiches puis **Termine**. Le mot d’arrêt ne doit pas avoir de fiche.

### Étape 3 – Compter les textes traités

1. Avant la boucle, initialisez `nombre_textes` à zéro.
2. Dans la boucle, augmentez-le de 1 pour chaque texte traité.
3. Après la boucle, affichez le total.

**Avant de tester**, complétez les deux dernières lignes du tableau.

| Saisies successives       | Nombre de textes attendu |
|---------------------------|--------------------------|
| chat, puis fin            | 1                        |
| fin dès le début          |                          |
| Une saisie vide, puis fin |                          |

**Convention** : une saisie vide compte comme un texte traité : longueur 0, catégorie **court**, zéro point. Elle ne provoque pas l’arrêt.

## ■ Mon bilan

- ☐ Je sais expliquer la différence entre `=` et `==`.
- ☐ Je sais conserver le résultat d’une méthode de chaîne.
- ☐ Je sais utiliser une valeur renvoyée par une fonction.
- ☐ Je peux suivre une boucle et expliquer ce qui permet son arrêt.

**Pour préparer la reprise** : notez une notion comprise, une difficulté et une question à poser à l’enseignant.