Algorithmique et programmation – Terminale spécialité

# Chapitre 1 – Les bases : variables, boucles, conditions

Pr. Elakili · Algorithme · Python · TI-82 · Octobre 2026

Site web : missionmaths360.com

## Ce que dit le programme de Terminale

Le programme demande de savoir **écrire, lire et compléter** des algorithmes en langage naturel et en **Python**, en particulier avec des **listes**, pour : calculer des termes et des sommes de termes d'une suite, **rechercher un seuil**, **approcher une limite ou une constante**, et approcher une solution d'équation par **dichotomie**. La calculatrice (ici la **TI-82**) sert à vérifier et aux questions « à l'aide de la calculatrice ».

## 1. Le dictionnaire des trois langages

| Notion | Algorithme | Python | TI-82 (menu en français) |
|---|---|---|---|
| affectation | $u \leftarrow 2$ | `u = 2` | `2→U` (touche `sto→`) |
| saisie d'une valeur | **Saisir** $n$ | `n = int(input())` | `Prompt N` (`prgm` E/S) |
| affichage | **Afficher** $u$ | `print(u)` | `Disp U` (`prgm` E/S) |
| boucle « $n$ fois » | **Pour** $k$ de 1 à $n$ | `for k in range(1, n+1):` | `Pour(K,1,N)` (`prgm` CTL) |
| boucle « tant que » | **Tant que** $u \leqslant A$ | `while u <= A:` | `TantQue U≤A` |
| test | **Si** … **alors** … **sinon** | `if …: else:` | `Si … Alors … Sinon` |
| fin d'un bloc | **Fin Pour**, **Fin Si**… | retour de l'indentation | `Fin` (**obligatoire**) |
| comparaisons | $\leqslant$, $<$, $=$, $\neq$ | `<=`, `<`, `==`, `!=` | `2nde` `math` (TEST) |
| puissance, produit | $2^n$, $0{,}5 \times u$ | `2**n`, `0.5*u` | `2^N`, `0.5U` |
| fonction | $f(x) = x^3 + x - 1$ | `def f(x): return …` | on la range dans `Y1` (`f(x)`) |

**Remarques.** • Sur une TI réglée en anglais : `For(`, `While`, `If`, `Then`, `Else`, `End`. • Sur la TI, `Pour(K,1,N)` va **jusqu'à** $N$ **inclus**, alors qu'en Python `range(1, n+1)` s'arrête à $n$ : `range(n)` donne $0, 1, \ldots, n-1$. • En Python et sur la TI, la virgule décimale s'écrit avec un **point** : `0.5`.

### ☞ Les pièges de syntaxe

• En Python, `=` **affecte** et `==` **compare** ; les deux-points `:` et l'**indentation** (4 espaces) sont obligatoires après `def`, `for`, `while`, `if`, `else`. • $a^3$ s'écrit `a**3` et surtout pas `a^3` (qui donne autre chose en Python !). • Sur la TI, chaque `Pour(`, `TantQue` ou `Si…Alors` doit être refermé par son propre `Fin`.

## 2. Prise en main de la TI-82

### Les manipulations à connaître

- **Créer un programme :** `prgm` → menu **NOUV** → `Créer nouveau`, taper le nom (le clavier est déjà en lettres), `entrer`. Chaque ligne commence par `:`.
- **Insérer une instruction :** dans l'éditeur, `prgm` puis menu **CTL** (`Si`, `Pour(`, `TantQue`, `Fin`) ou menu **E/S** (`Prompt`, `Disp`).
- **Taper une lettre :** `alpha` puis la touche marquée de cette lettre ; **la flèche** → : touche `sto→`.
- **Quitter l'éditeur :** `2nde` `mode` ; **lancer :** `prgm` → **EXEC** → choisir → `entrer` `entrer` ; **modifier :** `prgm` → **EDIT**.
- **Utiliser une fonction :** la taper dans `f(x)` en `Y1=`, puis dans un programme écrire `Y1(A)` pour $f(a)$ ; `Y1` s'obtient par `var` → menu **VAR Y** → `Fonction`.

*Sur la TI-82 Advanced Edition Python, les scripts Python de ce cours se tapent aussi tels quels dans l'application Python.*

## 3. Trois packs pour démarrer

### Pack 1 – un test

On saisit un réel $a$ ; afficher $f(a)$ pour $f(x) = x^3 + x - 1$ et dire si $f(a)$ est positif

**Algorithme**

**Saisir** $a$
$y \leftarrow a^3 + a - 1$
**Afficher** $y$
**Si** $y \geqslant 0$ **alors**
  **Afficher** « positif »
**Sinon**
  **Afficher** « négatif »
**Fin Si**

**Python**

```
def f(x):
    return x**3 + x - 1

def signe(a):
    y = f(a)
    if y >= 0:
        return y, "positif"
    else:
        return y, "négatif"
```

**TI-82**

```
:Prompt A
:Y1(A)→Y
:Disp Y
:Si Y≥0
:Alors
:Disp "POSITIF"
:Sinon
:Disp "NEGATIF"
:Fin
```

(avec Y1=X^3+X-1)

**Essais :** `signe(0.5)` renvoie `(-0.375, 'négatif')` et `signe(1)` renvoie `(1, 'positif')`. Comme $f(0{,}5) < 0 < f(1)$, cette fonction (strictement croissante) s'annule entre 0,5 et 1 : on la retrouvera au **chapitre 6 (dichotomie)**.

### Pack 2 – la boucle « Pour »

Tableau de valeurs de $f$ pour $x = 0, 1, \ldots, 5$

**Algorithme**

**Pour** $x$ allant de 0 à 5
  $y \leftarrow x^3 + x - 1$
  **Afficher** $x$ et $y$
**Fin Pour**

**Python**

```
for x in range(6):
    print(x, f(x))
# ou en une liste :
[f(x) for x in range(6)]
```

**TI-82**

```
:Pour(X,0,5)
:Disp X,Y1(X)
:Fin
```

**Résultat :** $f(0) = -1$, $f(1) = 1$, $f(2) = 9$, $f(3) = 29$, $f(4) = 67$, $f(5) = 129$ ; la liste vaut `[-1, 1, 9, 29, 67, 129]`. On sait **à l'avance** combien de tours faire : boucle « Pour ».
**Sans programme sur la TI :** entrer `Y1=X^3+X-1` dans [f(x)], régler [2nde][fenêtre] (déf table) : `DébTable=0`, `PasTable=1`, puis afficher la table [2nde][graphe].

### Pack 3 – la boucle « Tant que »

Combien de fois doubler 1 pour dépasser 1 000 ?

**Algorithme**

$u \leftarrow 1$ ; $n \leftarrow 0$
**Tant que** $u \leqslant 1000$
  $u \leftarrow 2u$
  $n \leftarrow n + 1$
**Fin Tant que**
**Afficher** $n$

**Python**

```
def doubler():
    u = 1
    n = 0
    while u <= 1000:
        u = 2*u
        n = n + 1
    return n
```

**TI-82**

```
:1→U
:0→N
:TantQue U≤1000
:2U→U
:N+1→N
:Fin
:Disp N
```

**Tableau d'exécution :**

| n | 0 | 1 | 2 | 3 | … | 9 | 10 |
|---|---|---|---|---|---|---|---|
| u | 1 | 2 | 4 | 8 | … | 512 | 1024 |

**Résultat :** `10`, car $2^9 = 512 \leqslant 1\,000$ et $2^{10} = 1\,024 > 1\,000$. On **ne sait pas** à l'avance combien de tours faire : boucle « Tant que » et un **compteur** $n$. C'est le modèle de la **recherche de seuil** (chapitre 4).

### À vous de jouer

(écrire chaque fois l'algorithme, le script Python et le programme TI-82)

**1.** Afficher les valeurs de $g(x) = x^2 - 4x + 1$ pour $x$ entier allant de $-2$ à 2.
**2.** Combien de fois faut-il multiplier 1 par 1,5 pour dépasser 100 ?
**3.** Que renvoie l'algorithme : $u \leftarrow 100$ ; $n \leftarrow 0$ ; **Tant que** $u > 1$ : $u \leftarrow u/3$ ; $n \leftarrow n + 1$ ; **Fin** ; **Afficher** $n$ ?
**Réponses : 1.** 13 ; 6 ; 1 ; $-2$ ; $-3$ (Python : `for x in range(-2, 3):` ; TI : `Pour(X,-2,2)`). **2.** 12 fois ($1{,}5^{11} \approx 86{,}5$ et $1{,}5^{12} \approx 129{,}7$). **3.** 5, car $3^4 = 81 < 100 < 243 = 3^5$.