Algorithmique et programmation – Terminale spécialité

Chapitre 3 – Sommes et produits de termes

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

Site web : missionmaths360.com

Méthode : la variable « accumulateur »

Pour calculer $S = a_1 + a_2 + \cdots + a_n$, on utilise une variable S que l'on **initialise à** 0, puis, à chaque tour de boucle, on lui **ajoute** le terme courant : $S \leftarrow S + a_k$. Pour un **produit** $P = a_1 \times a_2 \times \cdots \times a_n$, on initialise P à 1 et on fait $P \leftarrow P \times a_k$.

Pack 1 – somme des termes d'une suite récurrente $S_n = u_0 + u_1 + \cdots + u_n$ avec $u_0 = 3$, $u_{n+1} = 2u_n$

Algorithme

Saisir n
$u \leftarrow 3$; $S \leftarrow 0$
Pour k allant de 0 à n
  $S \leftarrow S + u$
  $u \leftarrow 2u$
Fin Pour
Afficher S

Python

```
def somme(n):
    u = 3
    S = 0
    for k in range(n+1):
        S = S + u
        u = 2*u
    return S
```

TI-82

```
:Prompt N
:3→U
:0→S
:Pour(K,0,N)
:S+U→S
:2U→U
:Fin
:Disp S
```

Ordre des lignes : on ajoute d'abord le terme courant u_k à S, **puis** on passe au terme suivant. La boucle fait $n+1$ tours car la somme contient $n+1$ termes (u_0 à u_n).

Tableau d'exécution :

k	0	1	2	3	4	5
u (avant)	3	6	12	24	48	96
S (après)	3	9	21	45	93	189

Essai : `somme(5)` renvoie `189`. **Contrôle par la formule :** $S_5 = 3 \times \dfrac{2^6 - 1}{2 - 1} = 3 \times 63 = 189$. ✓

Pack 2 – somme d'une suite explicite et approximation de π $S_n = \dfrac{1}{1^2} + \dfrac{1}{2^2} + \cdots + \dfrac{1}{n^2}$

Algorithme

Saisir n
$S \leftarrow 0$
Pour k allant de 1 à n
  $S \leftarrow S + \dfrac{1}{k^2}$
Fin Pour
Afficher S

Python

```
def carres(n):
    s = 0
    for k in range(1, n+1):
        s = s + 1/k**2
    return s
```

TI-82

```
:Prompt N
:0→S
:Pour(K,1,N)
:S+1/K^2→S
:Fin
:Disp S
```

En une ligne : `sum([1/k**2 for k in range(1, n+1)])` (liste en compréhension).

Résultats : $S_{10} \approx 1{,}549\,8$ et $S_{1000} \approx 1{,}643\,9$. On admet que S_n tend vers $\dfrac{\pi^2}{6} \approx 1{,}644\,9$: ainsi $\sqrt{6\,S_{1000}} \approx 3{,}140\,6$ est une valeur approchée de π.
Astuce TI sans programme : `somme(suite(1/K^2,K,1,10))` (`somme(` et `suite(` se trouvent dans le menu 2nde stats LISTES, onglets MATH et OPS).

Pack 3 – un produit et le nombre e $\mathrm{e} \approx 1 + \dfrac{1}{1!} + \dfrac{1}{2!} + \cdots + \dfrac{1}{n!}$ où $k! = 1 \times 2 \times \cdots \times k$

Algorithme

Saisir n
$S \leftarrow 1$ (le terme $\frac{1}{0!}$)
$f \leftarrow 1$ (contiendra $k!$)
Pour k allant de 1 à n
  $f \leftarrow f \times k$
  $S \leftarrow S + \dfrac{1}{f}$
Fin Pour
Afficher S

Python

```
def approx_e(n):
    S = 1
    f = 1
    for k in range(1, n+1):
        f = f*k
        S = S + 1/f
    return S
```

TI-82

```
:Prompt N
:1→S
:1→F
:Pour(K,1,N)
:F*K→F
:S+1/F→S
:Fin
:Disp S
```

On ne recalcule pas $k!$ à chaque tour : on passe de $(k-1)!$ à $k!$ en **multipliant par** k (accumulateur produit, initialisé à 1).
Résultats : `approx_e(5)` $\approx 2{,}716\,67$; `approx_e(10)` $\approx 2{,}718\,281\,8$; à comparer à e $\approx 2{,}718\,281\,83$ (touche [2nde] [ln]).

Pack 4 – une suite définie par une somme $S_n = \dfrac{1}{n+1} + \dfrac{1}{n+2} + \cdots + \dfrac{1}{2n} = \displaystyle\sum_{k=1}^{n} \frac{1}{n+k}$

Algorithme

Saisir n
$S \leftarrow 0$
Pour k allant de 1 à n
  $S \leftarrow S + \dfrac{1}{n+k}$
Fin Pour
Afficher S

Python

```
def S(n):
    s = 0
    for k in range(1, n+1):
        s = s + 1/(n+k)
    return s

[S(n) for n in range(1, 6)]
```

TI-82

```
:Prompt N
:0→S
:Pour(K,1,N)
:S+1/(N+K)→S
:Fin
:Disp S
```

Attention : ici le terme ajouté dépend **à la fois** de k (le compteur) et de n (fixé pendant toute la boucle). Quand n change, **toute** la somme change : on ne peut pas passer de S_n à S_{n+1} en ajoutant un seul terme.
Résultats : la liste vaut environ `[0.5, 0.5833, 0.6167, 0.6345, 0.6456]` ; $S_{10} \approx 0{,}668\,8$, $S_{100} \approx 0{,}690\,7$, $S_{1000} \approx 0{,}692\,9$. La suite semble croissante et converger vers $\ln 2 \approx 0{,}693\,1$.

☞ Les erreurs classiques

- Oublier d'initialiser S à 0 (ou P à 1), ou l'initialiser **dans** la boucle : la somme repart de zéro à chaque tour.
- Se tromper sur le **nombre de termes** : de u_0 à u_n, il y en a $n+1$ (`range(n+1)`, TI : `Pour(K,0,N)`).
- Inverser les deux lignes de la boucle : on additionne alors $u_1 + \cdots + u_{n+1}$.

À vous de jouer (algorithme + Python + TI-82)

1. Calculer $1 + 2 + \cdots + 100$ avec une boucle, puis en une ligne avec `sum` et `range`.

2. Calculer $T_n = \displaystyle\sum_{k=0}^{n} (2k+1)$ pour $n = 10$. Conjecturer une formule.

3. Calculer le produit $P_n = \left(1 + \dfrac{1}{1}\right)\left(1 + \dfrac{1}{2}\right)\cdots\left(1 + \dfrac{1}{n}\right)$ pour $n = 10$. Conjecturer, puis démontrer.

4. Calculer $\displaystyle\sum_{k=1}^{10} \frac{1}{k(k+1)}$ et comparer avec $\dfrac{10}{11}$.

5. Pour $u_0 = 2$ et $u_{n+1} = 0{,}5\,u_n + 3$ (chapitre 2), calculer $u_0 + u_1 + \cdots + u_4$.

6. Calculer le produit des 5 premiers entiers impairs $1 \times 3 \times 5 \times 7 \times 9$ à l'aide d'une boucle.

7. Écrire une fonction Python `somme_liste(L)` qui renvoie la somme des éléments d'une liste `L` **sans** utiliser `sum`, à l'aide de `for x in L:`. Sur la TI, que fait `somme(L1)` ?

Réponses : 1. 5 050 ; `sum(range(1, 101))`. **2.** $T_{10} = 121 = 11^2$: $T_n = (n+1)^2$. **3.** $P_{10} = 11$: $P_n = n+1$ (produit télescopique $\frac{2}{1} \times \frac{3}{2} \times \cdots \times \frac{n+1}{n}$). **4.** $\approx 0{,}909\,09 = \frac{10}{11}$ (somme télescopique). **5.** $2 + 4 + 5 + 5{,}5 + 5{,}75 = 22{,}25$. **6.** 945 (initialiser P à 1, puis $P \leftarrow P \times (2k-1)$ pour k de 1 à 5). **7.** `S = 0`, puis `for x in L: S = S + x`, puis `return S` ; `somme(L1)` renvoie directement la somme des éléments de la liste `L1`.