



Je tape mon programme Python — encadrer $\sqrt{2}$

Guide pas à pas · Thonny / Capytale · on encadre $\sqrt{2}$ sans utiliser la touche racine

Nom : _____

Classe : _____

Poste n° : _____

Objectif. Écrire un programme qui **encadre** $\sqrt{2}$ aussi précisément qu'on veut, **par balayage** : on avance petit à petit sans jamais dépasser. C'est la même idée qu'à la main, en beaucoup plus rapide.

1 Comprendre l'idée



On cherche le plus grand nombre a (avec n décimales) tel que $a \times a \leq 2$. On teste 0 ; 0,1 ; 0,2 ... pour trouver l'encadrement au dixième, puis au centième, etc. Astuce du programme : on travaille avec des **entiers** en multipliant par $p = 10^n$, pour éviter les erreurs de virgule.

2 Taper la fonction



Dans l'**Éditeur**, tape :

```
def encadrer(x, n):  
    p = 10 ** n  
    a = 0  
    while a * a <= x * p * p:  
        a = a + 1  
    a = a - 1  
    return a / p, (a + 1) / p
```

⚠ Attention à l'**indentation** : la ligne `a = a + 1` est **dans** le `while` (8 espaces). N'oublie pas les **deux-points** après `encadrer(x, n)` et après `x * p * p`.

x → le nombre dont on cherche la racine (ici 2)

n → le nombre de décimales voulues

while → « tant que a^2 ne dépasse pas $x \cdot p^2$ »

return → renvoie l'encadrement a/p et $(a+1)/p$

🔍 Pourquoi $p = 10^n$ et des entiers ?

L'ordinateur calcule **mal avec les virgules** : il trouve $1,4 \times 1,4 = 1,959999...$ au lieu de 1,96 ! Pour éviter ces erreurs, on ne teste pas $(a/p)^2 \leq 2$ mais la même chose **sans virgule** : on multiplie par p^2 , ce qui donne $a \times a \leq 2 \times p \times p$. Ainsi **a** reste un **entier** (0, 1, 2, 3...) et tous les calculs sont **exacts**.

Exemple au dixième ($p = 10$) : on cherche le plus grand a avec $a^2 \leq 200$ → c'est **a = 14** (car $14^2 = 196 \leq 200$ mais $15^2 = 225 > 200$), d'où $1,4 \leq \sqrt{2} < 1,5$.

3 Tester au dixième



Saute une ligne (sans indentation) et ajoute :

```
print(encadrer(2, 1))
```

Enregistre puis  Exécuter (F5). Dans la console :

Console / Shell

>>>

```
(1.4, 1.5)
```

Ce que j'obtiens :

Ça veut dire : $1,4 \leq \sqrt{2} < 1,5$. 👍

4 De plus en plus précis



Remplace le print par une boucle qui affiche l'encadrement pour 0, 1, 2, 3, 4 décimales :

```
for n in range(0, 5):
    bas, haut = encadrer(2, n)
    print(bas, "<= racine(2) <", haut)
```

Console / Shell

>>>

```
1.0 <= racine(2) < 2.0
1.4 <= racine(2) < 1.5
1.41 <= racine(2) < 1.42
1.414 <= racine(2) < 1.415
1.4142 <= racine(2) < 1.4143
```

On gagne une décimale à chaque ligne : la machine « resserre » l'encadrement.

5 À moi d'essayer



Change le nombre : essaie `encadrer(3, 2)` puis `encadrer(10, 2)`. Qu'obtiens-tu ?

Tu dois trouver $1,73 \leq \sqrt{3} < 1,74$ et $3,16 \leq \sqrt{10} < 3,17$.

SOS Si ça ne marche pas — regarde le message d'erreur :

- **IndentationError** → vérifie les 4 espaces (fonction) et les 8 espaces (dans le `while`).
- Le programme **ne s'arrête pas** → vérifie que `a = a + 1` est bien dans la boucle : sans elle, `a` ne change jamais.
- **SyntaxError** → un `:` oublié après `def ...`) ou après `x * p * p`.