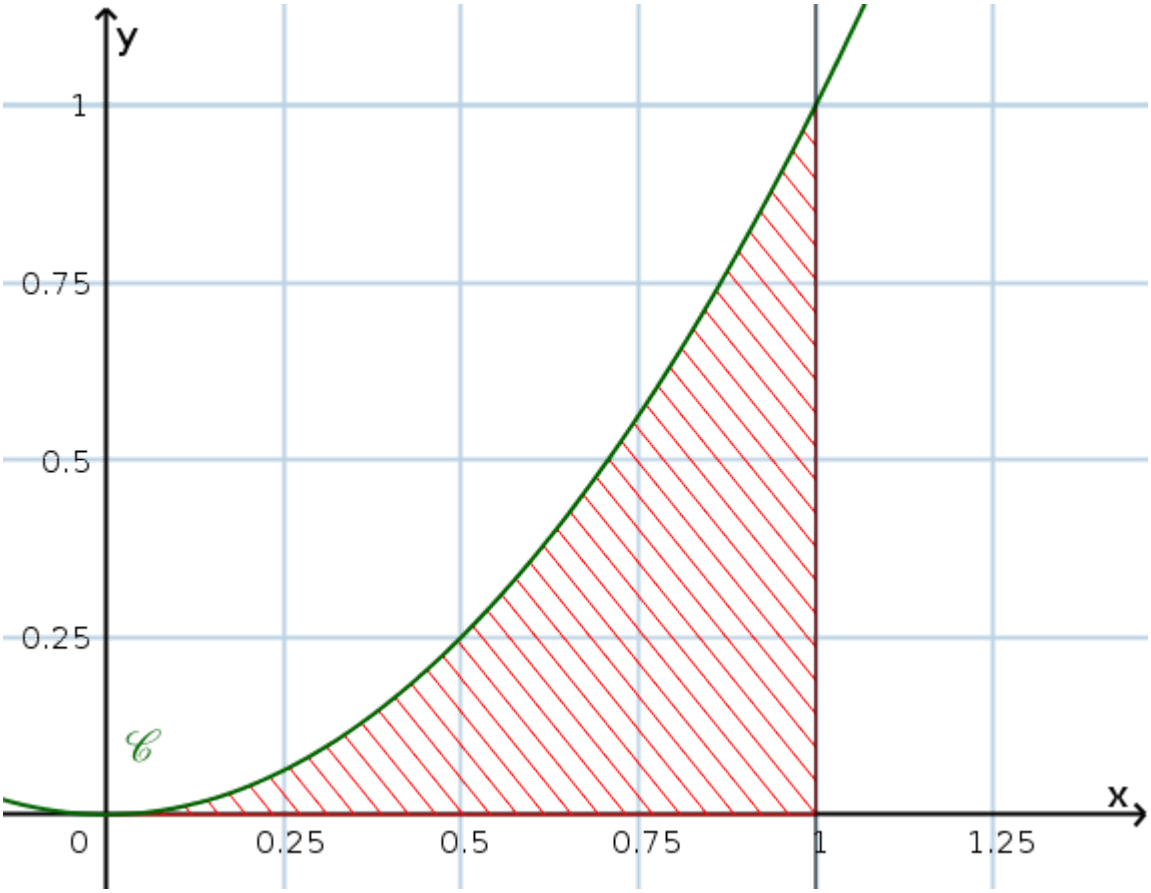
Table des matières

Problématique

On considère dans cette activité la fonction f définie sur $[0;3]$ par $f(x)=x^2$ et $\mathscr{C}$ sa courbe représentative dans un repère orthonormé.

L'objectif de cette activité est de calculer l'aire du domaine délimité

- d'une part par les droites d'équation $x=0$ et $x=1$
- d'autre part par l'axe des asbcisses et la courbe $\mathscr{C}$



En d'autre termes, nous cherchons à déterminer l'aire hachurée en rouge sous la courbe $\mathcal{C}$ pour x compris entre 0 et 1.

```
# On définit dans la mémoire de python notre fonction f
def f(x):
    return x**2
```

Méthodes

Nous allons explorer 2 méthodes, mais il en existe bien d'autres.

La méthode de Monte-Carlo

Il s'agit d'une méthode probabiliste.

On prend des points aléatoirement dans le carré de côté 1. La proportion de points situés sous la courbe approche l'aire.

Voici une fonction python **à compléter** :

```
from random import random # on importe la fonction raandom() depuis la bibliothèque random

def monte_carlo(n):
    '''Cette foncion retourne la proportion des n points générés aléatoirement qui se situent sous la courbe représentant la fonction f précédemment définie'''
    compteur = 0  # on va compter au fur et à mesure de la génération des points
    for i in range(n): # on répète n fois
        x = random()   # x est une abscisse aléatoire de l'intervalle [0 ; 1]
        y = random()   # y est une ordonnée aléatoire de l'intervalle [0 ; 1]
        if y < f(x) :  # si le point est sous la courbe
            compteur += 1   # On incrémente le compteur de 1 ( x+=1 est un raccourci de x=x+1)
    return (compteur / n)  # on renvoie la fréquence des points sous la courbe.
```

Nous allons tester cette fonction dans la cellule ci-dessous :

```
monte_carlo(10)
```

```
0.4
```

Relancez la cellule précédente.

- Si les résultats vous paraissent incompatibles avec le dessin appelez-moi.
- Sinon exécutez la cellule ci-dessous qui va appeler la fonction 21 fois en augmentant progressivement le nombre de points.

```
for i in range (7):
    for j in range(3):
        print("Avec", 10**(i+1), "points, on approche l'aire ",monte_carlo(10**(i+1)))
```

```
Avec 10 points, on approche l'aire  0.4
Avec 10 points, on approche l'aire  0.3
Avec 10 points, on approche l'aire  0.3
Avec 100 points, on approche l'aire  0.34
Avec 100 points, on approche l'aire  0.31
Avec 100 points, on approche l'aire  0.33
Avec 1000 points, on approche l'aire  0.337
Avec 1000 points, on approche l'aire  0.318
Avec 1000 points, on approche l'aire  0.307
Avec 10000 points, on approche l'aire  0.3411
Avec 10000 points, on approche l'aire  0.3346
Avec 10000 points, on approche l'aire  0.3314
Avec 100000 points, on approche l'aire  0.33437
Avec 100000 points, on approche l'aire  0.3351
```

```
Avec 100000 points, on approche l'aire  0.33201
Avec 1000000 points, on approche l'aire  0.333035
Avec 1000000 points, on approche l'aire  0.333649
Avec 1000000 points, on approche l'aire  0.333956
Avec 10000000 points, on approche l'aire  0.3333612
Avec 10000000 points, on approche l'aire  0.3333387
Avec 10000000 points, on approche l'aire  0.3335661
```

On constate que la proportion semble converger vers une valeur qui doit être l'aire, mais que cette convergence est très lente.

En effet avec 10 millions de points, on ne semble sûr que de la troisième décimale.

Nous allons essayer de trouver une méthode plus efficace.

La méthode des rectangles

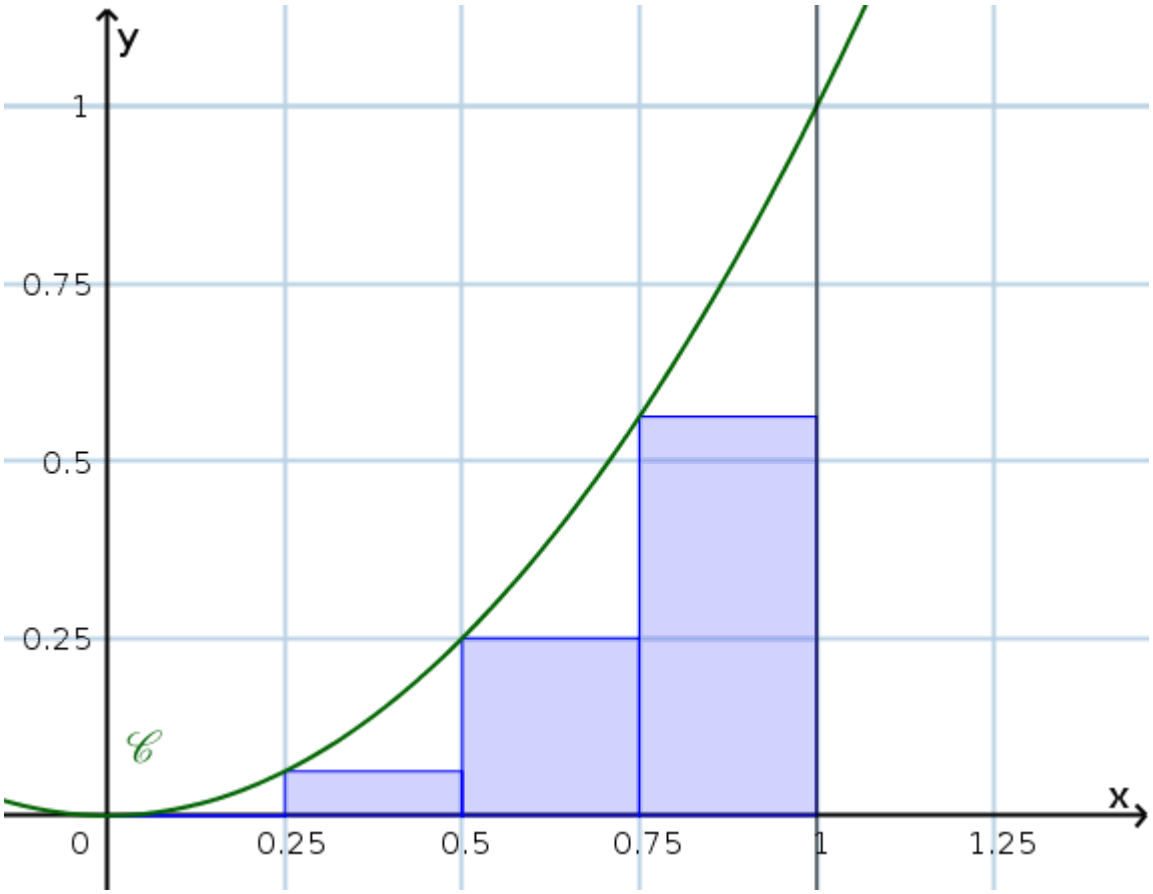
L'idée va être d'encadrer l'aire recherchée par deux aires que l'on peut calculer aisément. On utilise pour cela des figures très élémentaires : des rectangles.

Nous allons calculer un minorant de l'aire en inscrivant des rectangles sous la courbe (voir rectangles rouges) et un majorant de l'aire en construisant des rectangles au dessus de la courbe (voir rectangles bleus).

En effet, l'aire d'un rectangle est facile à calculer. Il suffit donc de faire la somme des aires de chacun des rectangles pour obtenir un encadrement de l'aire recherchée.

Rectangles inférieurs (bleus)

On cherche à déterminer l'aire du rectangle dont la base est située entre les abscisses x et $x+dx$ (dx désignant la largeur de chaque rectangle) et dont la hauteur est telle que le rectangle soit situé juste au dessous de la courbe $\mathscr{C}$.



- Calculez l'aire bleue dessinée ci-dessus.
  - vous pouvez utiliser la cellule de code au-dessous de l'espace de réponse comme **calculatrice**.
  - la **fonction f est déjà** en mémoire
- Exprimez l'aire de ces rectangles, en fonction de x, dx et de la fonction f.

```
0.25*f(0)+0.25*f(0.25)+0.25*f(0.5)+0.25*f(0.75)
```

```
0.21875
```

ici dx = 0,25.

L'aire bleue est 0,25 × 0 + 0,25 × f(0,25) + 0,25 × f(0,5) + 0.25 × f(0,75) soit 0 + 0,25 × 0,25² + 0,25 × 0,5² + 0,25 × 0,75² = 0,21875

L'aire de chacun des rectangles est $dx \times f(x)$ pour x allant de 0 à 0,75 par pas de dx=0,25

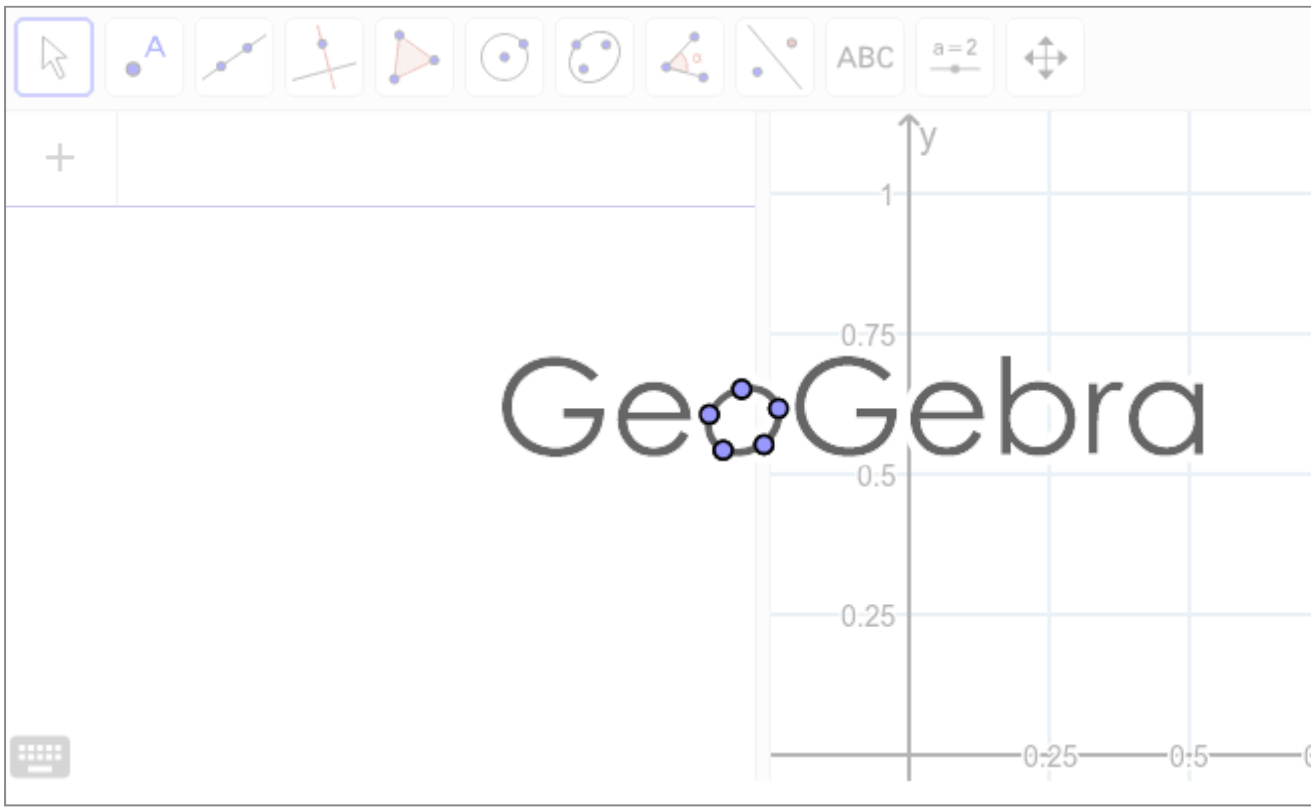
```
######################################################################
#  un premier programme à compléter pour automatiser les calculs  #
######################################################################

n = 4        # n contient le nombre de rectangles
dx = 1 / n     # calcul de dx la largeur des rectangles
inf = 0      # la variable inf va contenir le cumul des aires au fur et à mesure
for i in range(n):  # pour chaque rectangle
#    print(i)         # sert à voir que i commence à 0 et non à 1 !
    x = i * dx       # calcul de x pour f croissante
    inf = inf + dx * f(x)    # cumul des aires
print (inf)     # affichage de l'aire inférieure
```

```
0.21875
```

Avec Geogebra

Exécutez la cellule ci-dessous.

```
%%html
<iframe scrolling="no" title="Introduction calcul integral"
src="https://www.geogebra.org/material/iframe/id/cjjtwens/width/846/height/400/bord
er/888888/sfsb/true/smb/false/stb/true/stbh/false/ai/true/asb/false/sri/false/rc/tr
ue/ld/false/sdz/false/ctl/false" width="846px" height="400px" style="border:0px;">
</iframe>
```



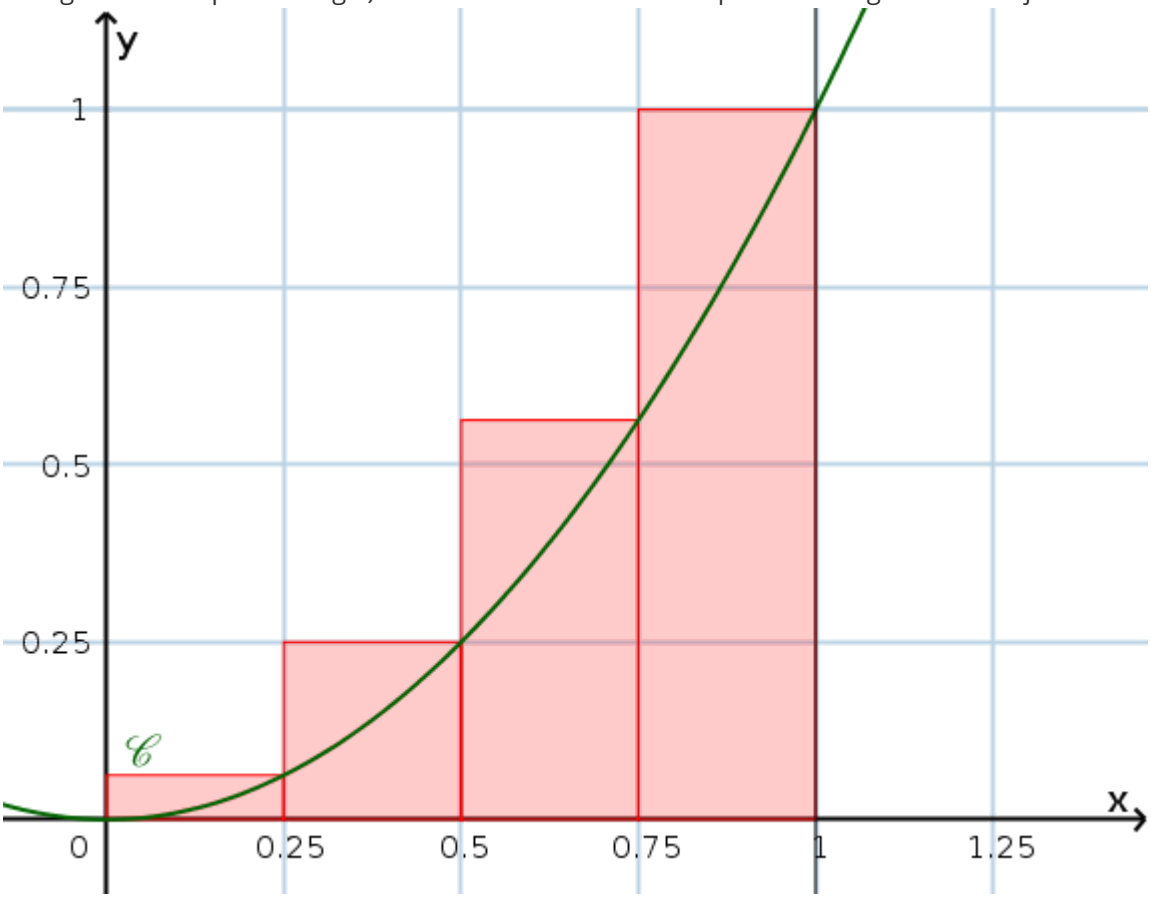
Réalisez les actions ci-dessous dans la cellule **Géogebra** ci-dessus.

- *Saisissez* `x²` dans l'espace de saisie (à droite du signe + et à gauche du repère) **puis** *validez*
- *Créez* un curseur `n` allant de 1 à 100 avec un incrément de 1
- *Saisissez* `Inf=SommeInférieure[f,0,1,n]` **puis** *validez*.
- *Déplacez* le curseur et *observez* les affichages. Complétez la conjecture ci-dessous :

Plus n augmente, plus l'aire des rectangles s'approche **de l'aire cherchée tout en restant inférieure**.

Rectangles supérieurs (rouges)

On cherche à déterminer l'aire du rectangle dont la base est située entre les abscisses x et $x+dx$ (dx désignant la largeur de chaque rectangle) et dont la hauteur est telle que le rectangle soit situé juste sous la courbe $\mathscr{C}$.



- Calculez l'aire rouge dessinée ci-dessus.
- Exprimez l'aire de ces rectangles, en fonction de x, dx et de la fonction f.

```
0.25*f(.25)+0.25*f(0.5)+0.25*f(0.75)+0.25*f(1)
```

```
0.46875
```

ici dx = 0,25.

L'aire rouge est 0,25 × f(0,25) + 0,25 × f(0,5) + 0.25 × f(0,75) + 0,25 × f(1) + soit 0,25 × 0,25² + 0,25 × 0,5² + 0,25 × 0,75² + 0,25 × 1 = 3,5 + 0,5 × 3,75 = 0,46875

L'aire de chacun des rectangles est $dx \times f(x)$ pour x allant de 0,25 à 1 **ou bien**

L'aire de chacun des rectangles est $dx \times f(x + dx)$ pour x allant de 0 à 0,75

```
#  un deuxième programme pour automatiser les calculs
#########################################################

n = 4        # n contient le nombre de rectangles
dx = 1 / n     # calcul de dx la largeur des rectangles
sup = 0      # la variable sup va contenir le cumul des aires au fur et à mesure
for i in range(n):  # pour chaque rectangle
    x = (i+1) * dx          # calcul de x pour f croissante
    sup = sup + dx * f(x)    # cumul des aires
print (sup)     # affichage de l'aire inférieure
```

```
0.46875
```

Avec Geogebra

Dans la cellule où Géogebra est actif :

- Masquez l'affichage de `Inf`.
- Saisissez `Sup=SommeSupérieure[f,0,1,n]`.
- Déplacez le curseur et observer les affichages. Complétez la conjecture ci-dessous :

Plus n augmente, plus l'aire des rectangles s'approche de **l'aire cherchée tout en restant supérieure**.

L'aire cherchée semble être de l'ordre de $\frac{1}{3}$

Dans la cellule où Géogebra est actif :

- Masquez l'affichage de `Sup`
- Saisissez `Intégrale[f,0,1]`.
- Cela confirme notre dernière conjecture.

Calculs avec python

```
# définition des paramètres nécessaires

a = 0 # borne inférieure des abscisses
b = 1 # borne supérieure des abscisses
n = 4 # Nombre de rectangles
```

- Complétez la fonction ci-dessous dont le but est de calculer la somme des aires des rectangles sous la courbe (rouges) et la somme des aires des rectangles au dessus de la courbe (bleus).

```python
def encadre(f, a, b, n): # on définit la fonction qui reçoit 4 paramètres
    dx = (b-a)/n      # largeur d'un rectangle
    sommeSup = 0  # Aire des rectangles supérieurs
    sommeInf = 0  # Aire des rectangles inférieurs
    x = a         # Bord gauche du rectangle considéré
    for i in range(n):  # Répéter n fois
        sommeSup = sommeSup + dx * f(x + dx) # On cumule les aires des rectangles supérieurs
        sommeInf = sommeInf + dx * f(x)     # On cumule les aires des rectangles inférieurs
        x = x + dx
    return(sommeInf, sommeSup) # la fonction renvoie deux nombres qui encadrent l'aire cherchée
```

```python
# et on appelle la fonction une fois
encadre(f,a,b,n) # Les 4 paramètres f, a, b et n existent en mémoire
```

```
(0.21875, 0.46875)
```

Ce qui doit confirmer vos calculs précédents.

Pour voir comment se comporte cet encadrement, on le calcule plusieurs fois en augmentant le nombre de rectangles.

Validez la cellule ci-dessous pour effectuer ce travail.

```python
for i in range(7): # pour i allant de 0 à 7
    e = encadre(f,a,b,10**i) # Calculer les bornes de l'encadrement de l'aire avec 10^i rectangles
    print ("Pour", 10**i,"rectangle(s), l'aire est comprise entre", e[0], "et", e[1])
```

```
Pour 1 rectangle(s), l'aire est comprise entre 0.0 et 1.0
Pour 10 rectangle(s), l'aire est comprise entre 0.2849999999999999 et 0.3849999999999999
Pour 100 rectangle(s), l'aire est comprise entre 0.32835000000000036 et 0.3383500000000004
Pour 1000 rectangle(s), l'aire est comprise entre 0.33283350000000095 et 0.33383350000000095
Pour 10000 rectangle(s), l'aire est comprise entre 0.3332833349999431 et 0.3333833349999431
Pour 100000 rectangle(s), l'aire est comprise entre 0.33332833334937134 et 0.3333383333493713
Pour 1000000 rectangle(s), l'aire est comprise entre 0.33333283333399866 et 0.3333338333339987
```

Comment semble se comporter les deux suites encadrant l'aire ?

Les deux suites semblent converger vers le même nombre proche de $\frac{1}{3}$

Modifiez ci-dessous le programme précédant `for i in ...` pour qu'il calcule et affiche **l'amplitude** des 7 encadrements précédants.

```
# Modifiez le Copié collé du programme précédant ci-dessous :
for i in range(7): # pour i allant de 0 à 7
    e = encadre(f,a,b,10**i) # Calculer les bornes de l'encadrement de l'aire avec 10^i rectangles
    print ("Pour", 10**i,"rectangle(s), l'amplitude de l'encadrement est", e[1]-e[0])
```

```
Pour 1 rectangle(s), l'amplitude de l'encadrement est 1.0
Pour 10 rectangle(s), l'amplitude de l'encadrement est 0.09999999999999998
Pour 100 rectangle(s), l'amplitude de l'encadrement est 0.010000000000000009
Pour 1000 rectangle(s), l'amplitude de l'encadrement est 0.0010000000000000009
Pour 10000 rectangle(s), l'amplitude de l'encadrement est 9.999999999998899e-05
Pour 100000 rectangle(s), l'amplitude de l'encadrement est 9.99999999995449e-06
Pour 1000000 rectangle(s), l'amplitude de l'encadrement est 1.0000000000287557e-06
```

Complétez :

En multipliant le nombre de rectangles par 10, l'amplitude de l'encadrement est quasiment divisée par 10.

Pour obtenir une valeur approchée de l'aire à 10^{-10} près, il faudrait utiliser 10^{11} rectangles, soir cent milliards de rectangles.

Bilan

Dès qu'une fonction est positive sur un intervalle $[a;b]$, on peut calculer l'aire délimitée par

- l'axe des abscisses
- la courbe représentant la fonction
- la droite d'équation $x = a$.
- la droite d'équation $x = b$. Il est donc légitime d'utiliser une notation reprenant ces 4 éléments.

Notation mathématiques

La notation indiquant l'aire hachurée sous la courbe que nous venons d'approcher est $\int_0^1 x^2 \, dx = \frac{1}{3}$

Cette notation a été inventée par Gottfried Wilhelm von Leibniz.

- le symbole $\int$ peut être vu comme la première lettre "S" du mot somme.
- 0 est la valeur de a, la borne inférieure de l'intervalle.
- 1 est la valeur de b, la borne supérieure de l'intervalle.
- x^2 est le procédé de calcul de l'image de x par la fonction.
- dx joue dans cette notation,
    - le même rôle que dans la méthode des rectangles
    - la largeur du rectangle dont la hauteur est x^2
    - mais nous lui attribuerons plus tard une autre signification, plus fondamentale.

Application à une autre fonction

En utilisant la fonction encadre - éventuellement en définissant une nouvelle fonction - déterminez une valeur approchée de $\int_0^{\frac{\pi}{2}} sin(x)\ dx$ avec une précision de l'ordre de 10^{-4}.

```
from math import pi
from math import sin

def g(x):
    return sin(x)

encadre(g, 0, pi/2, 10**4)
```

```
(0.9999214581275232, 1.0000785377602026)
```

D'après les fonctions l'aire correspondant à $\int_0^{\frac{\pi}{2}} sin(x)\,dx = 1$ à 10^{-4} près

Méthode des trapèzes

On peut remplacer les rectangles par des trapèzes.

On perd l'encadrement mais on accélère la convergence vers l'aire.

Si vous avez le temps, essayez ci-dessous.

```
def trapeze(f, a, b, n): # on définit la fonction qui reçoit 4 paramètres
    # f le nom de la fonction, a et b les bornes de l'intervalle et n le nombre de rectangles
    dx = (b-a)/n        # largeur d'un rectangle
    aire = 0      # aire des trapèzes
    x = a         # Bord gauche du trapèze considéré
    for i in range(n):  # Répéter n fois
        aire = aire + 0.5* dx * (f(x)+f(x+dx)) # On cumule les aires des rectangles supérieurs
        x = x + dx                  # On passe à l'abscisse suivante
    return(aire) # la fonction renvoie un nombre qui approche l'aire cherchée
```

Testez votre fonction `trapeze()` et comparez avec les encadrements obtenus par la méthode des rectangles.

```
for i in range(7): # pour i allant de 0 à 7
    print ("Pour", 10**i,"trapèze(s), l'aire est environ", trapeze(f,0,1,10**i))
```

```
Pour 1 trapèze(s), l'aire est environ 0.5
Pour 10 trapèze(s), l'aire est environ 0.33499999999999996
Pour 100 trapèze(s), l'aire est environ 0.33335000000000037
Pour 1000 trapèze(s), l'aire est environ 0.33333350000000034
Pour 10000 trapèze(s), l'aire est environ 0.33333333499994316
Pour 100000 trapèze(s), l'aire est environ 0.3333333333493709
Pour 1000000 trapèze(s), l'aire est environ 0.3333333333339986
```

La convergence est plus rapide.

Avec 1000 trapèzes, on obtient 6 décimales exactes, alors que l'encadrement par les rectangles était au millième.