

**RAPPORT DU TP CALCUL DIFFERENTIEL**

**LIMITES ET DERIVEES**

**UE : CALCUL DIFFERENTIEL**

**CHARGE : M. Tcha AKONA**

**Nom Prénoms**

DOH Kodzo Benjamin(GM\_S2)

DJOSSOU Kokou Armand Light(IA&BD\_S2)

SEGUE Yao Freeman(IA\_BD\_S2)

SIGBEGNON kossi Jacque (LF\_GC\_S2)

**Année académique : 2024-2025**

Table des matières

[INTRODUCTION ET OBJECTIF GENERAL 3](#_Toc201054968)

[1.1-la concentration d’un médicament dans le sang 3](#_Toc201054969)

[1.1.1 Outils et définition de la fonction 3](#_Toc201054970)

[1.1.2-Implémentation en Python pour identifier les 3](#_Toc201054971)

[périodes d’efficacité du médicament 3](#_Toc201054972)

[1.1.4-l’execution de la fonction actif pour a=0 et b=6 4](#_Toc201054973)

[1.1.5- Représentation graphiquement la fonction qui modélise la concentration du produit 5](#_Toc201054974)

[1.1.6 recherches du Maximin 5](#_Toc201054975)

[1.1.7 analyses du graphe 6](#_Toc201054976)

[1.1.7.1 calcule du dérivé 6](#_Toc201054977)

[1.1.7.2 calcule et représentation de la tangente en x = 4 6](#_Toc201054978)

[1.1.7.3 études de la position relatif de la courbe et de la tangente 7](#_Toc201054979)

[1.1.8-Étude de la rapidité de diminution du médicament dans le sang entre 2h et 6h 8](#_Toc201054980)

[1.2 implémentation du code complet 9](#_Toc201054981)

[1.3 Recherche de la limite des fonctions 10](#_Toc201054982)

[1.3.1 utilisation de numpy et matplotlib 10](#_Toc201054983)

[1.3.2 Utilisation de sympy 14](#_Toc201054984)

[CONCLUSION 15](#_Toc201054985)

# INTROSUCTION ET OBJECTIF GENERALE

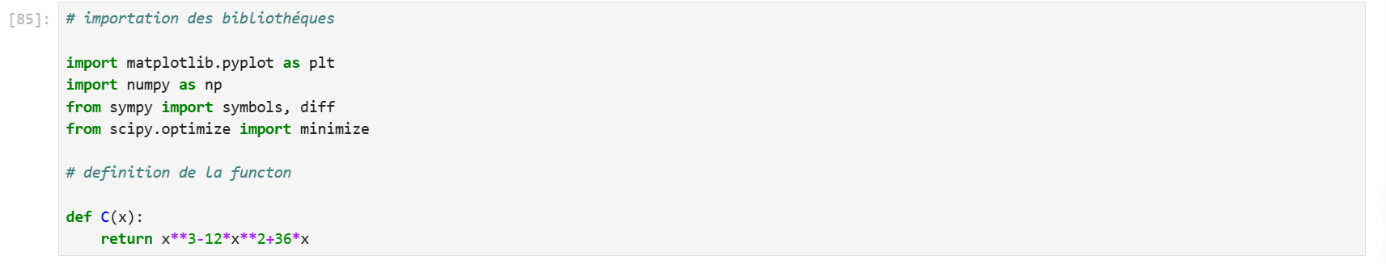
Ce rapport technique présente une exploration approfondie des outils numériques de Python, notamment les bibliothèques **NumPy**, **SymPy**, **SciPy** et **Matplotlib**, appliquées à la résolution de problèmes mathématiques et scientifiques. L’objectif principal est de mettre en évidence l’efficacité de ces outils dans l’analyse de fonctions, la modélisation de phénomènes réels et la représentation graphique des résultats. Nous étudierons notamment l’application de méthodes numériques telles que **la dichotomie** et **la méthode de Newton-Raphson** pour la résolution d’équations non linéaires, ainsi que l’analyse des propriétés de fonctions, incluant le **calcul de dérivées**, **l’étude de limites** et **la tracée de courbes**. Le rapport comprendra également une étude de cas portant sur la concentration d’un médicament dans le sang, illustrant de manière concrète l’utilisation de ces bibliothèques pour modéliser et analyser un phénomène réel à l’aide d’outils numériques performants.

# 1.1-la concentration d’un médicament dans le sang

La concentration du produit actif dans le sang, en milligrammes par litres de sang, est modélisée par la fonction *C*(*x*) = *x*3 −12*x*2 +36*x*, ou *x* ∈ [0; 6]. Et on sait aussi que le produits le produit actif est efficace si la concentration dans le sang est supérieur à 5*mg*/*l.*

# 1.1.1 Outils et définition de la fonction

Dans cette première partie, nous aurons besoin d’utiliser certaines bibliothèques Python telles que **NumPy**, **SymPy**, **SciPy** et **Matplotlib**. Afin de les exploiter dans notre programme, il est nécessaire de les importer au préalable. Nous définirons ensuite la fonction de concentration C(x), qui modélise l'évolution du médicament dans le sang. L’importation des bibliothèques ainsi que la définition de la fonction se font comme suit :

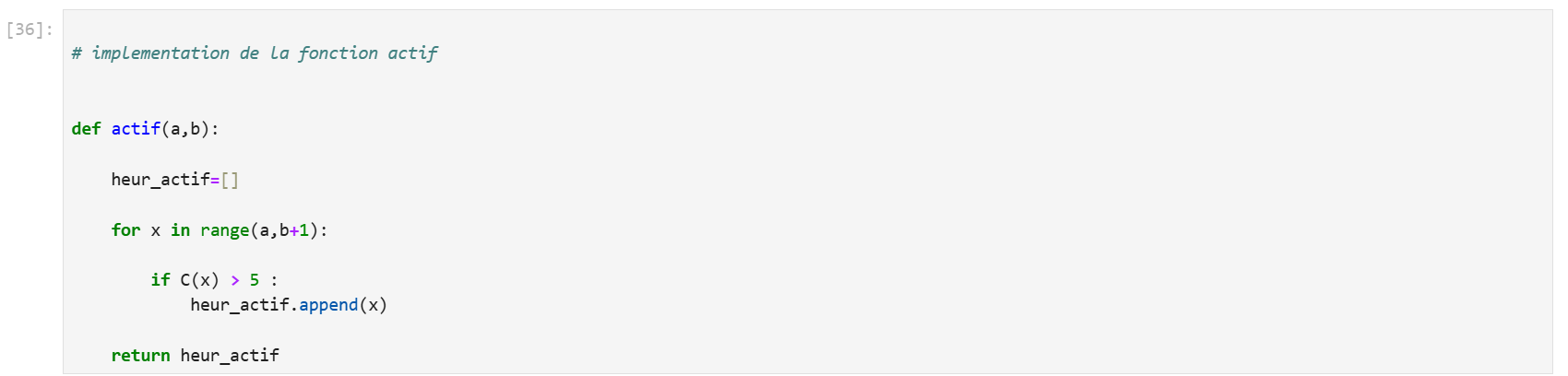


# 1.1.2-Implémentation en Python pour identifier les périodes d’efficacité du médicament

Le produit actif étant considéré comme efficace lorsque sa concentration dans le sang est **supérieure à 5 mg/L**, il est donc naturel de chercher les instants *x* ∈ [0; 6]. Pour lesquels la fonction de concentration vérifie la condition suivante :

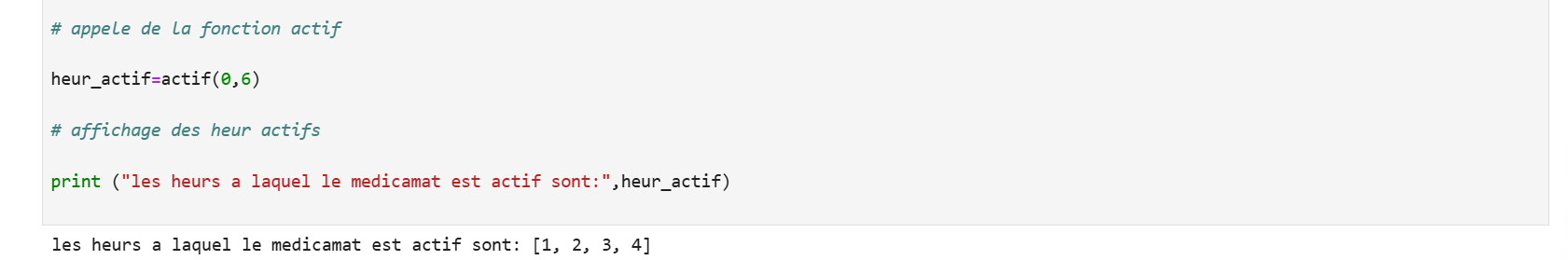
*C*(*x*) 5*mg*/*l* soit *x*3 −12*x*2 +36*x* > 5

Grâce aux bibliothèques Python, notamment **NumPy** et **SymPy**, il devient facile d’identifier numériquement les valeurs de *x* dans l’intervalle étudié qui satisfont cette inégalité. Cela permet de déterminer précisément les périodes pendant lesquelles le médicament reste actif dans le sang. Nous allons créer une fonction nommée **actif** qui prendra en paramètre les bornes d’un intervalle de temps. À l’intérieur de cette fonction, nous définirons une liste appelée **heur\_actif**(que la fonction retournera par après ), destinée à stocker toutes les valeurs de *x* pour lesquelles la concentration du médicament dans le sang est **strictement supérieure à 5 mg/L**, c’est-à-dire lorsque la condition *C*(*x*) 5*mg*/*l* est vérifiée, çà s’implémente comme suite :



# 1.1.4-l’execution de la fonction actif pour a=0 et b=6

Nous pouvons ensuite appeler facilement notre fonction *actif*, en lui fournissant simplement les bornes de l’intervalle à étudier. Dans notre cas, nous choisissons les valeurs a=0 et b=6, correspondant à l’intervalle de temps pendant lequel l’on observe l’évolution de la concentration.  
L’appel et l’exécution de la fonction se font comme suit :

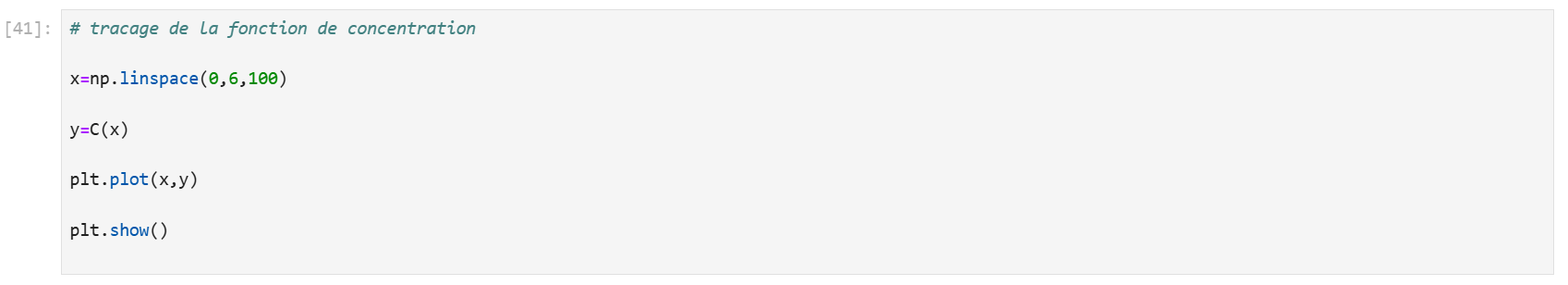


***Interprétation***

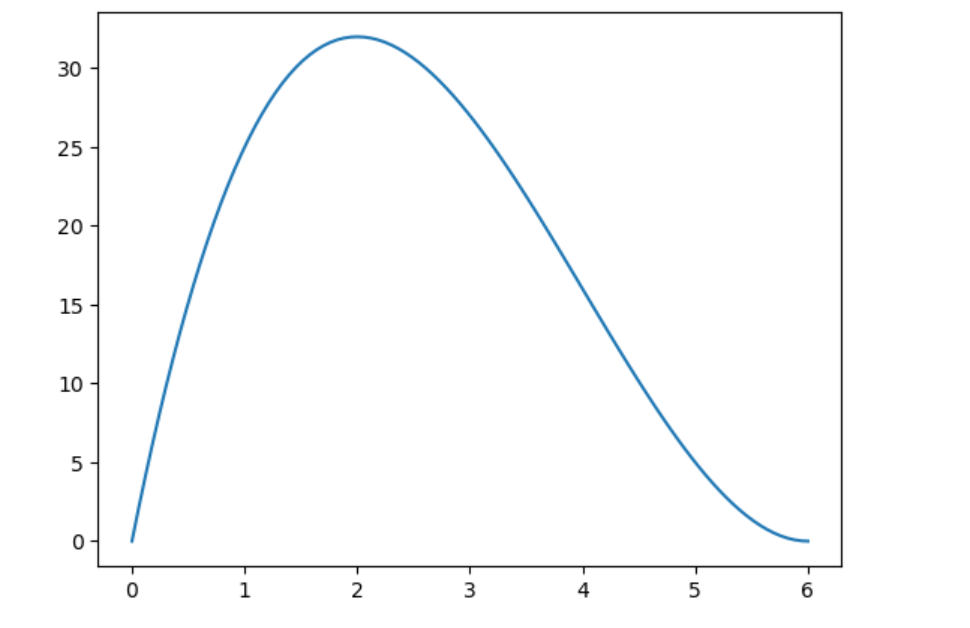
D’après les résultats obtenus, le médicament introduit dans l’organisme est actif entre la première et la quatrième heure. En effet, durant cet intervalle, la concentration du principe actif dans le sang reste supérieure à 5 mg/L. Au-delà de la quatrième heure, cette concentration passe en dessous du seuil d’efficacité, ce qui indique que le médicament commence à perdre ses effets thérapeutiques.

# 1.1.5- Représentation graphiquement la fonction qui modélise la concentration du produit

Afin d’avoir une vision plus intuitive de l’évolution de la concentration du médicament dans le sang, nous procédons à la représentation graphique de la fonction *C*(*x*) = *x*3 −12*x*2 +36*x*, ou *x* ∈ [0; 6]. Cette visualisation nous permet d’identifier clairement les périodes pendant lesquelles la concentration dépasse le seuil d’efficacité. La courbe ainsi obtenue met en évidence le comportement croissant puis décroissant de la concentration au fil du temps ça s’obtient comme suite :



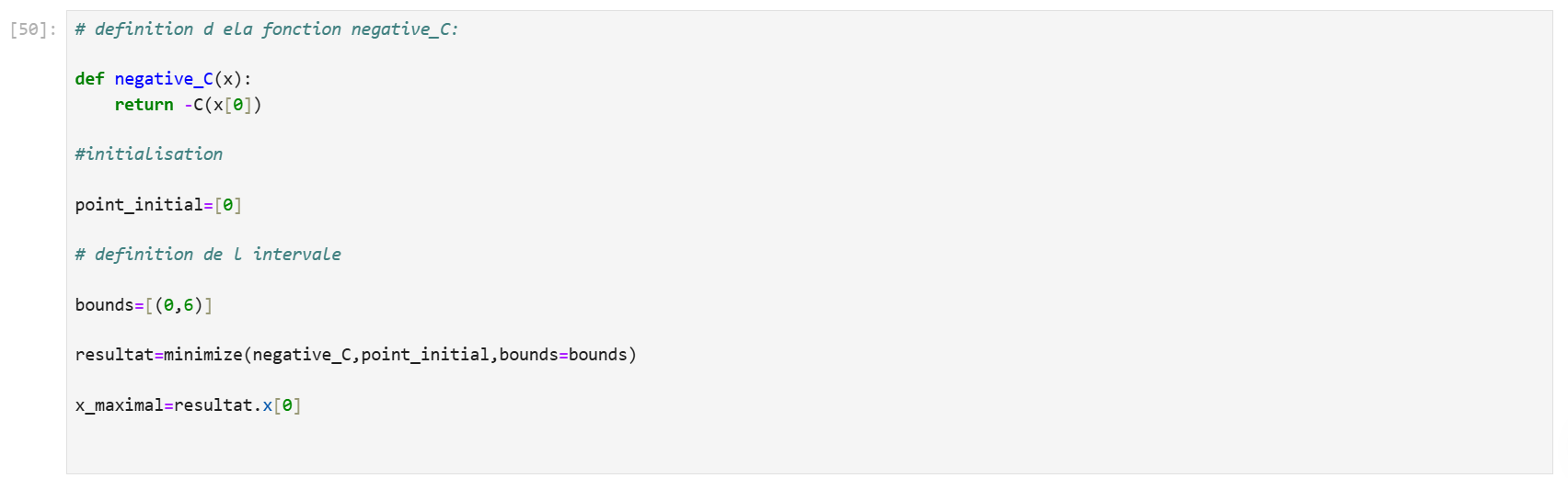
Apres l’exécution on obtient :



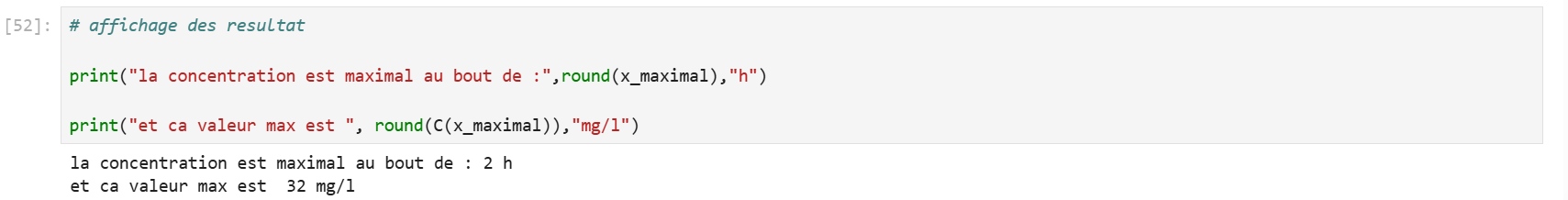
# 1.1.6 recherches du Maximin

Pour déterminer le maximum de la fonction de concentration *C*(*x)*, nous allons utiliser la bibliothèque *scipy* de Python. L’un des modules de cette bibliothèque, *scipy.optimize*, permet de minimiser une fonction. Étant donné que nous cherchons ici un maximum, nous allons définir une nouvelle fonction nommée *negative\_C,* qui retourne l’opposée de la fonction *C*(*x*). Ainsi, minimiser -C(x) revient à maximiser C(x).Nous devons ensuite spécifier une valeur initiale (appelée *point\_initial*) à partir de laquelle l’algorithme commencera la recherche du minimum. Dans notre cas, nous choisissons 0. Nous devons également définir un intervalle de recherche à l’aide du paramètre *bounds*, qui correspond ici à [0;6], l’intervalle défini dans l’énoncé.

L’ensemble de cette démarche est traduit en Python par le code suivant



Nous pouvons essayez d’afficher les résultats de ce code

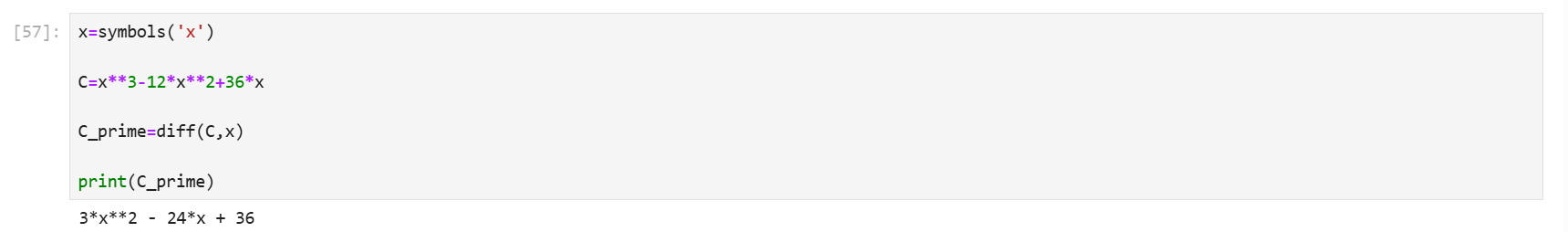


# 1.1.7 analyses du graphe

# 1.1.7.1 calcule du dérivé

La fonction *C*(*x)* , étant un polynôme, est définie sur l’ensemble des réels . L’intervalle d’étude [0;6] étant inclus dans , on en déduit que *C*(*x)* est continue et dérivable sur [0;6],car toute fonction polynomiale est de classe sur .

Python nous permet de caller la fonction dériver a l’aide =du bibliothèque *numpy* il nous faut importe la bibliothèques *numpy* ce qui est fait dans la partie *1.1.1* alors le calcule de la fonction dériver su fait comme suite :

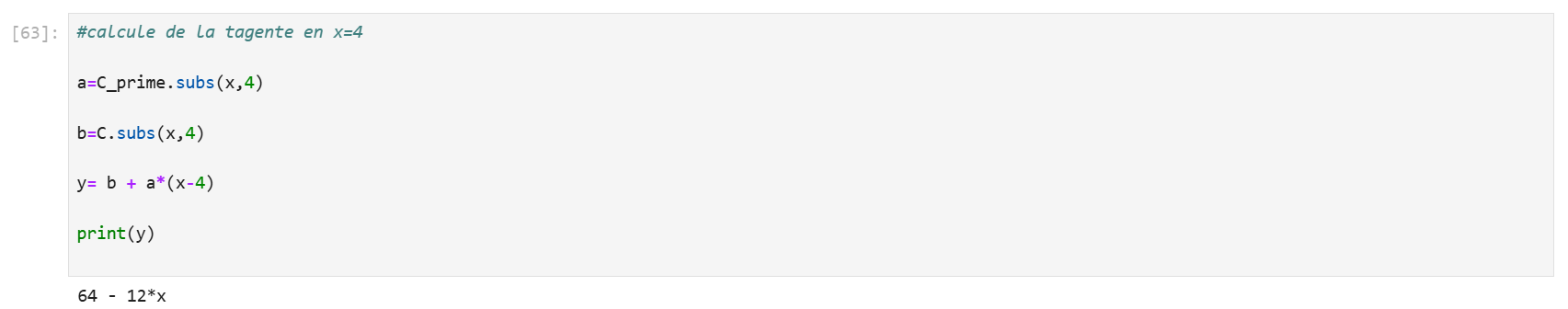


On constante bien que ***C’*(*x)= 3x*2 −24*x* +36**

# 1.1.7.2 calcule et représentation de la tangente en x = 4

Pour analyser localement le comportement de la fonction *C*(*x)* au voisinage de *x* =4, nous allons déterminer l'équation de la tangente à la courbe représentative de C en ce point. La tangente donne une approximation affine de la fonction au point considéré, et permet également de comparer visuellement la position relative de la courbe par rapport à la droite. Cette démarche implique de calculer la **dérivée** *C’*(*x),*puis d’évaluer à *x* =4 la pente de la tangente ainsi que la valeur de la fonction C(4). Une fois l’équation obtenue, nous tracerons à l’aide de *matplotlib* à la fois la courbe de *C*(*x),* et sa tangente en *x* =4.

Pour chercher la tangente nous aurons besoin d’utiliser encor une fois *numpy*

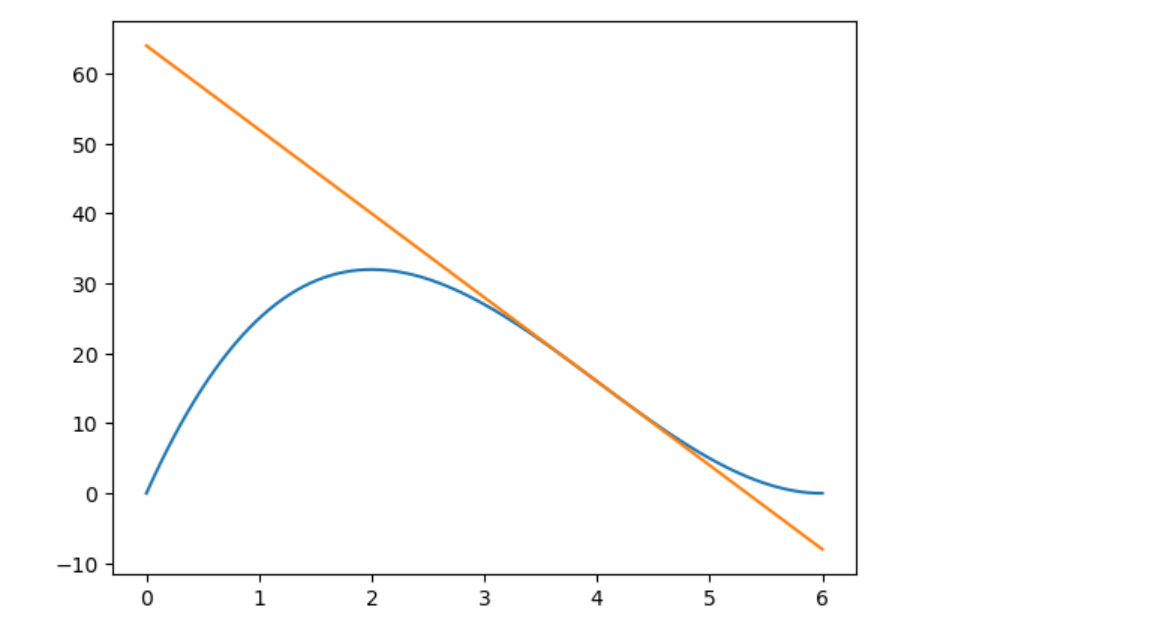


L’équation de la tangente en 4 est bel et bien ***-12x+64***

Essayons de tracé la tangente en 4 on a :



Et l’exécution nous donne



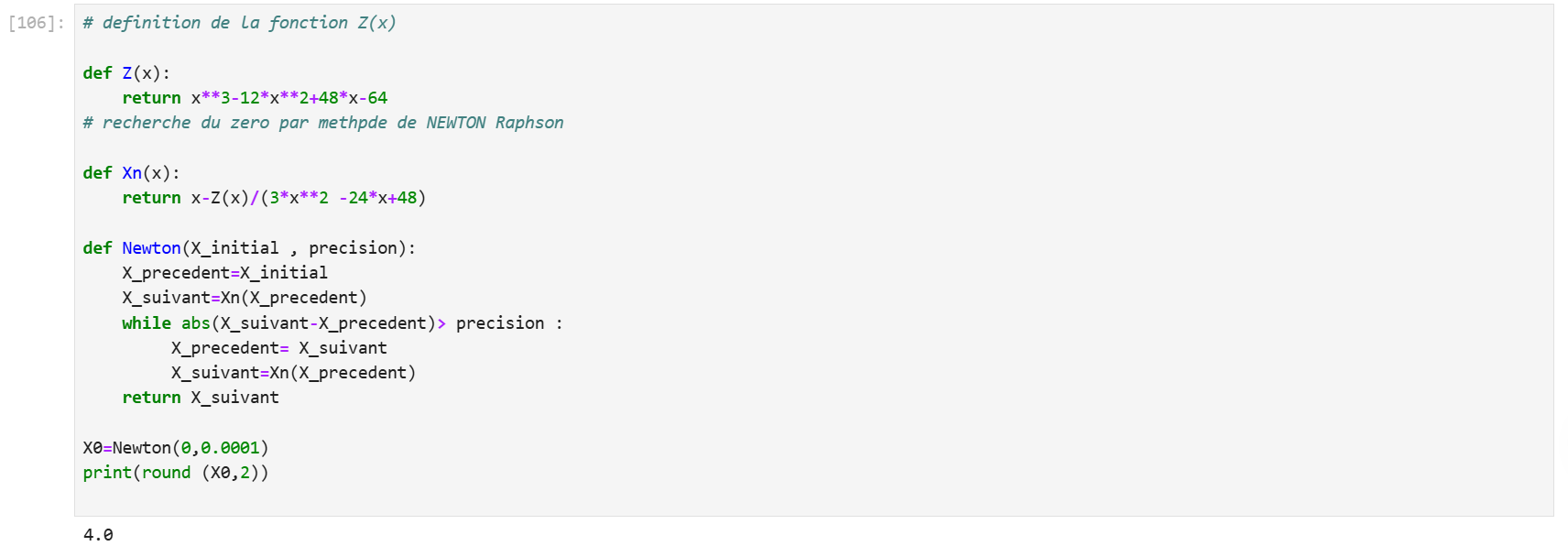
# 1.1.7.3 études de la position relatif de la courbe et de la tangente

Pour connaitre la position relation de la coure et de la tenante il nous faut étudier le signe de la fonction *C*(*x)* −*T(x)* avec *T(x)* l’équation de la tangente en 4.

*C*(*x)* −*T(x) = x3 +36x – 64*

Par la méthode de newton Raphson essayons de trouver une solution pour l’équation *C*(*x)* −*T(x) =*0 c est à dire *x3 +36x – 64 = 0.* Avecl’implémentation du code en python en retrouve facilement le racine de cette équation .

Premièrement nous allons définir une fonction *Z(x) = x3 +36x – 64 ,* nous allons implémenter cette fonction en python puis après appliquer la méthode de Newton raphson pour trouver la solution de l’équation Z(x) =0 ; ça s’explique comme suite

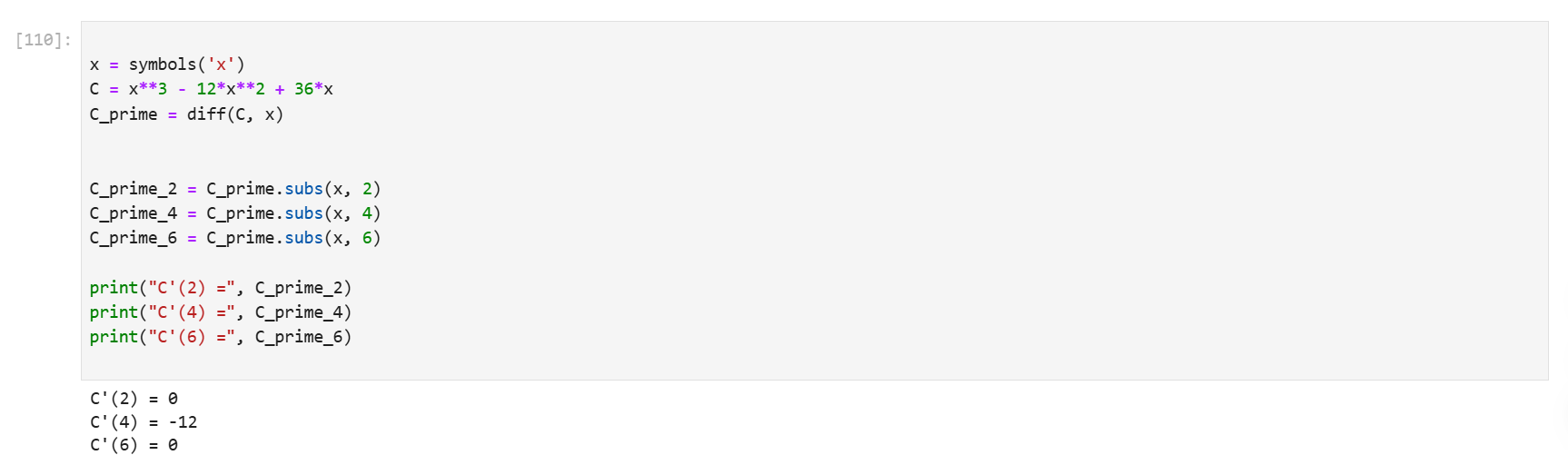


On constante que pour x=4 la fonction Z(x) s’annule alors on a le tableau récapitulatif comme suite

|  |  |  |
| --- | --- | --- |
| X | 0 4 6 | |
| Z(x) | Négative | Positive |
| Position de T et C | C au-dessous de la tangente | C au-dessus de la tangente |

# 1.1.8-Étude de la rapidité de diminution du médicament dans le sang entre 2h et 6h

L’affirmation du médecin est « La concentration du produit actif dans le sang diminue **plus vite** entre 2h et 4h qu'entre 4h et 6h » nous allons vérifier la viridité de cette affirmation pour ce fais on analyse la vitesse de diminution c est a dire la dériver de la fonction C

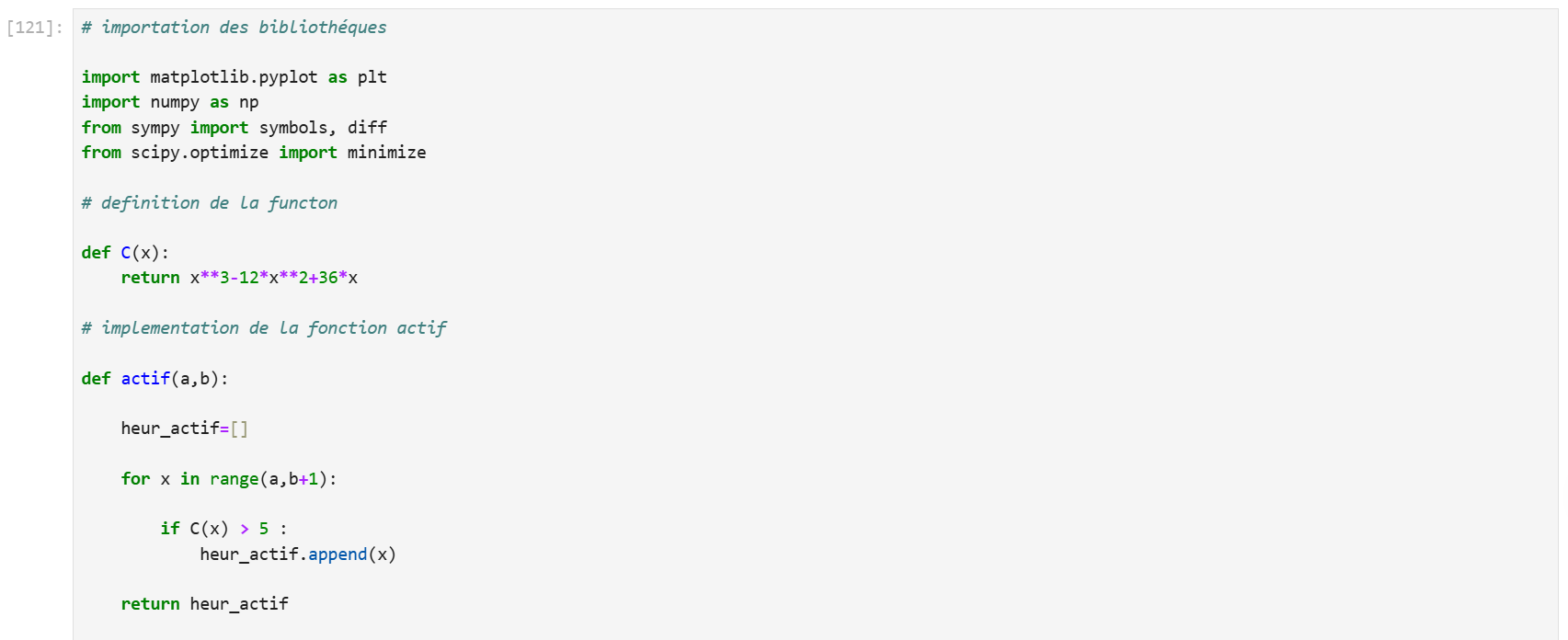


Entre 2h et 4h, la dérivée passe de 0 à -12 → la concentration diminue fortement

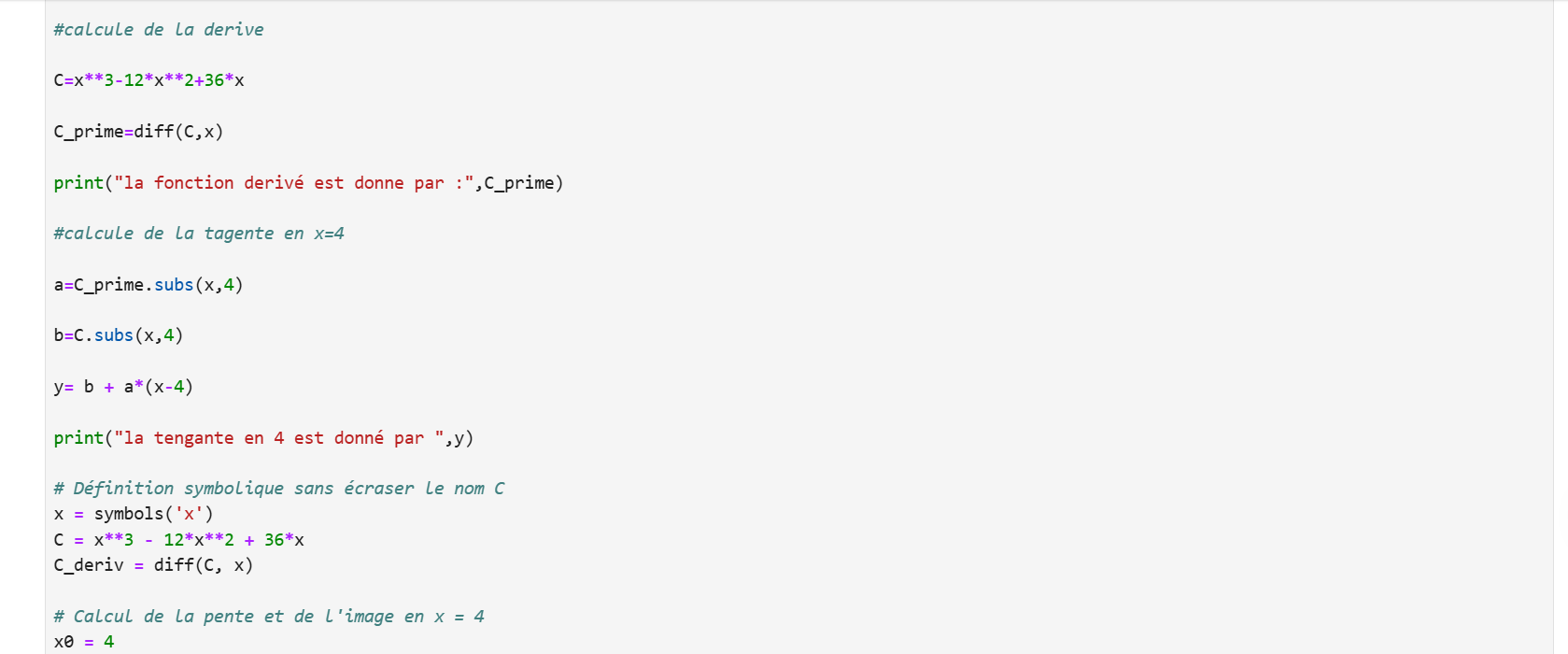
Entre 4h et 6h, la dérivée passe de -12 à 0 → la concentration diminue moins vite

Alors on constante que le médecin a bel et bien raison

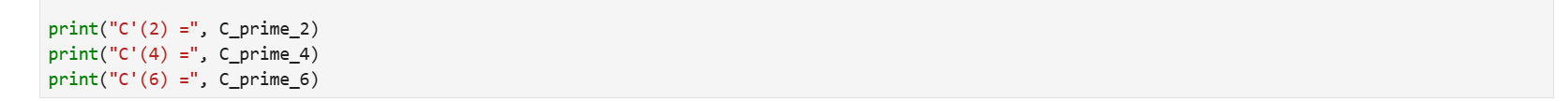
# 1.2 implémentation du code complet

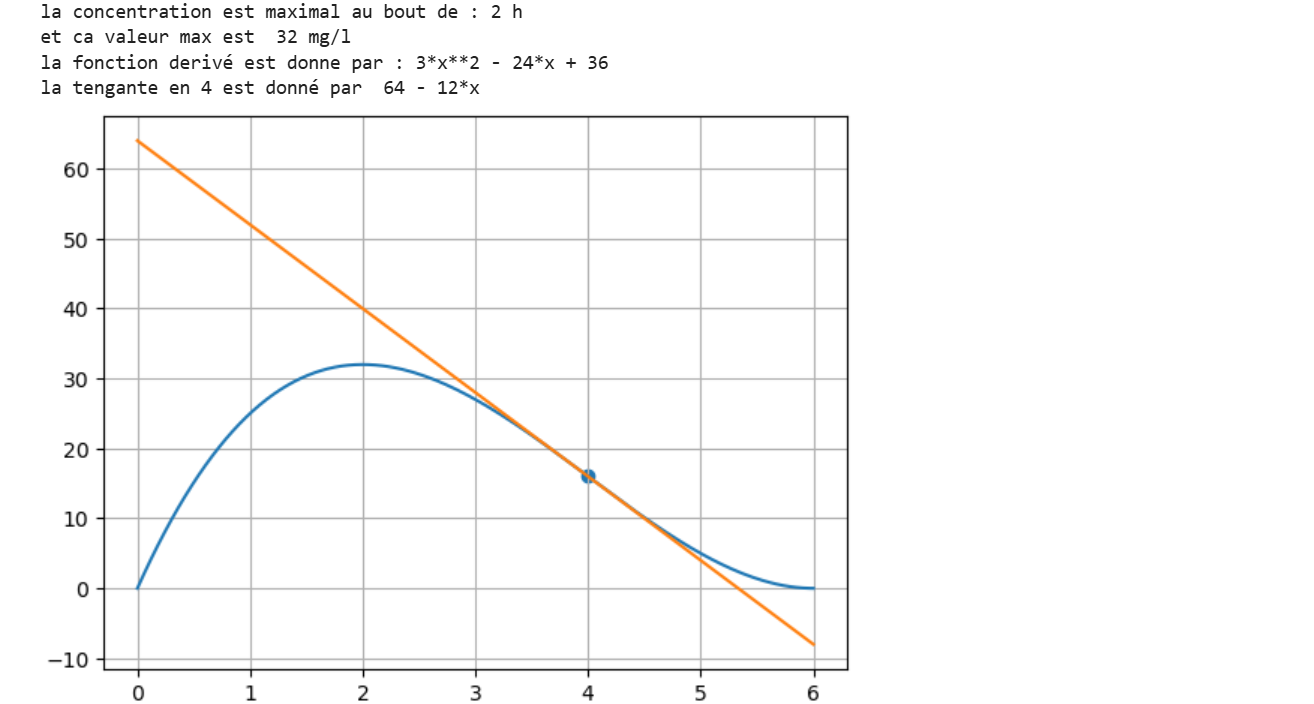
******

******

******

******

******

******

# 1.3 Recherche de la limite des fonctions

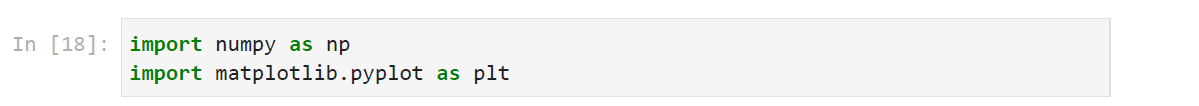
# 1.3.1 utilisation de numpy et matplotlib

Grace au Numpy matplotlib et sympy du python il est aisé de trouver la limite des fonctions même les plus complexe.

Déterminons la limite des fonction suivantes :

* en

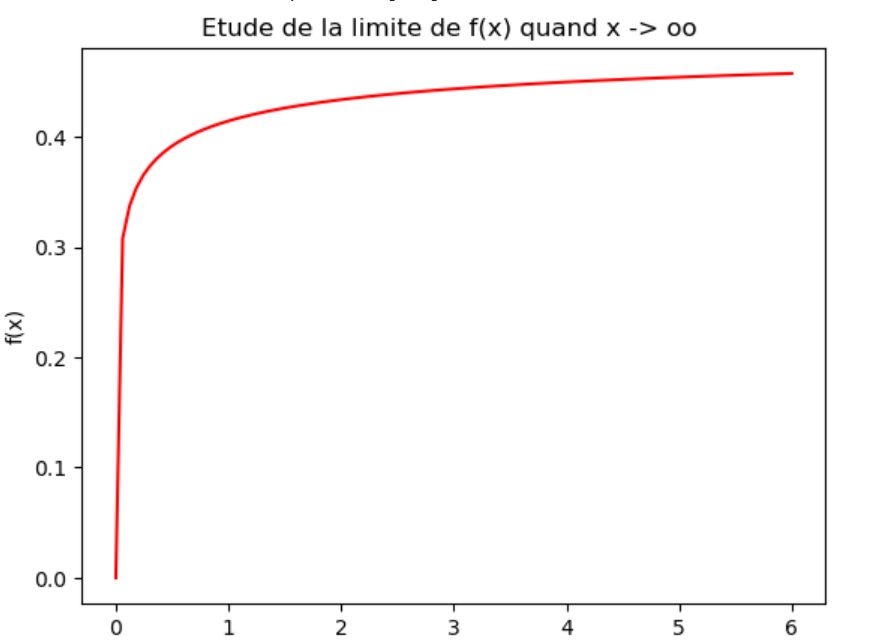
Premièrement on import *numpy* et *matplotlib*



Apres avoir définit la fonction on trace la courbe représentatif sur [0,6]



Apres l’exécution on obtient



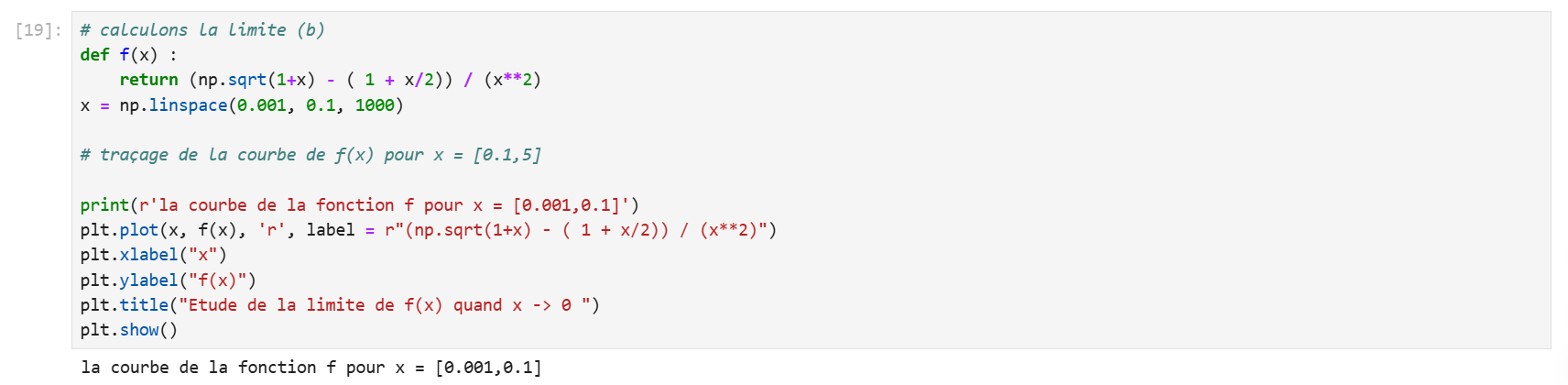
Pour trouver la limite de f en nous allons calculer l’image d’une valeur trés grand qui tende vers l’infinis et essayer de déduire la limite

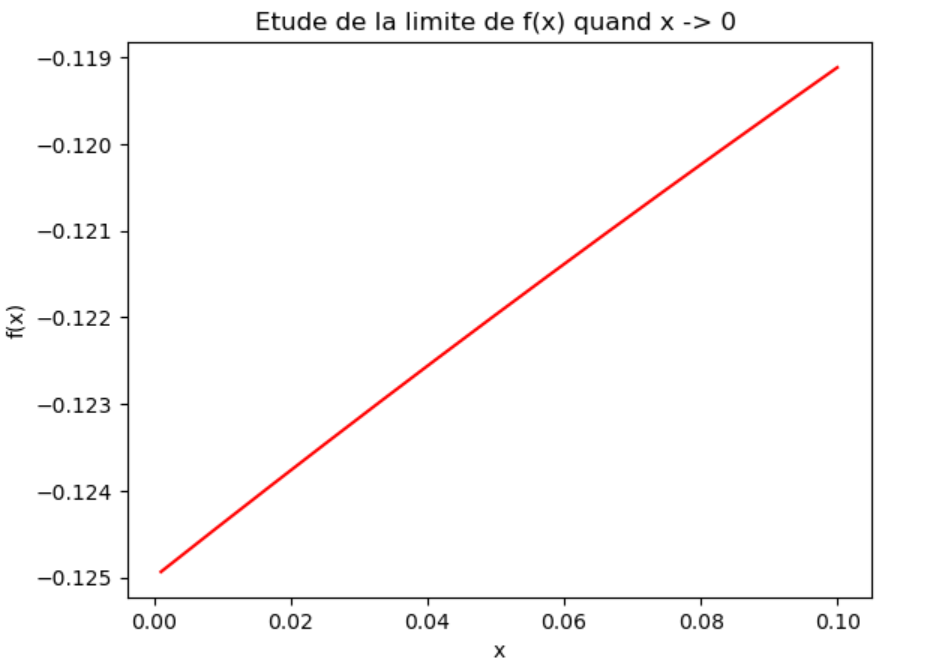


On remarque pour une très grande valeur de x , f(x) tend vers 0.5 alors **la limite de f(x) quand x tend ver est egale à 1/2**

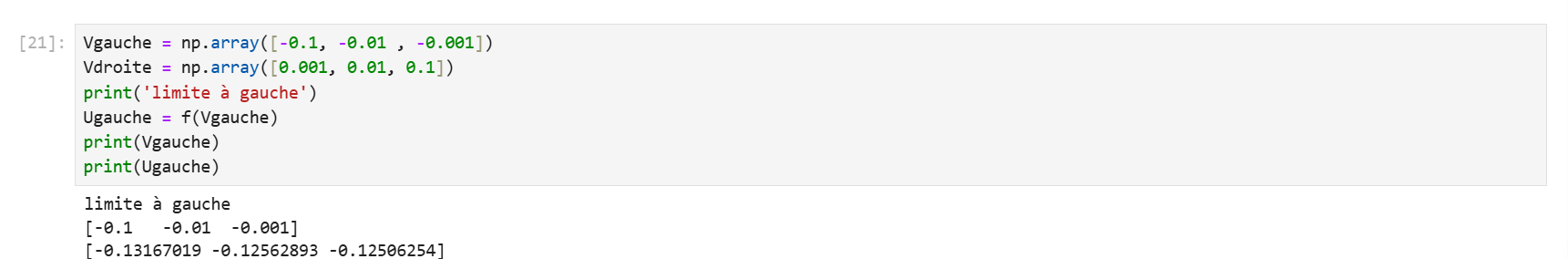
* en 0

Pour calculer cette limite traçons la courbe de la fonction au voisinage de 0 pour une meilleure visualisation. En choisissant un intervalle très proche de 0 comme [0.001 , 0.1] on :



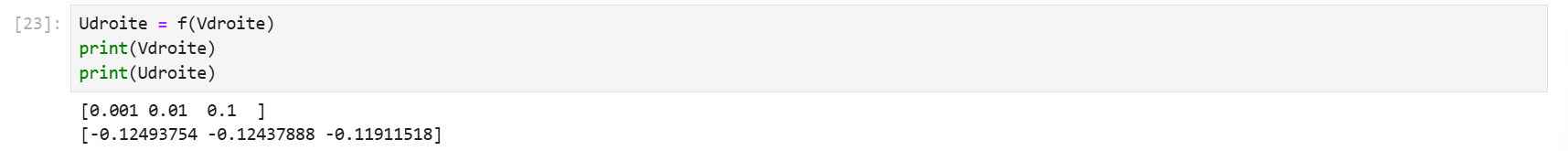


On calcule la limite a gauche et et droite ca se fait comme suite



On constante que la limite a gauche donne 12506

Essayons de le calculer à droite on a :

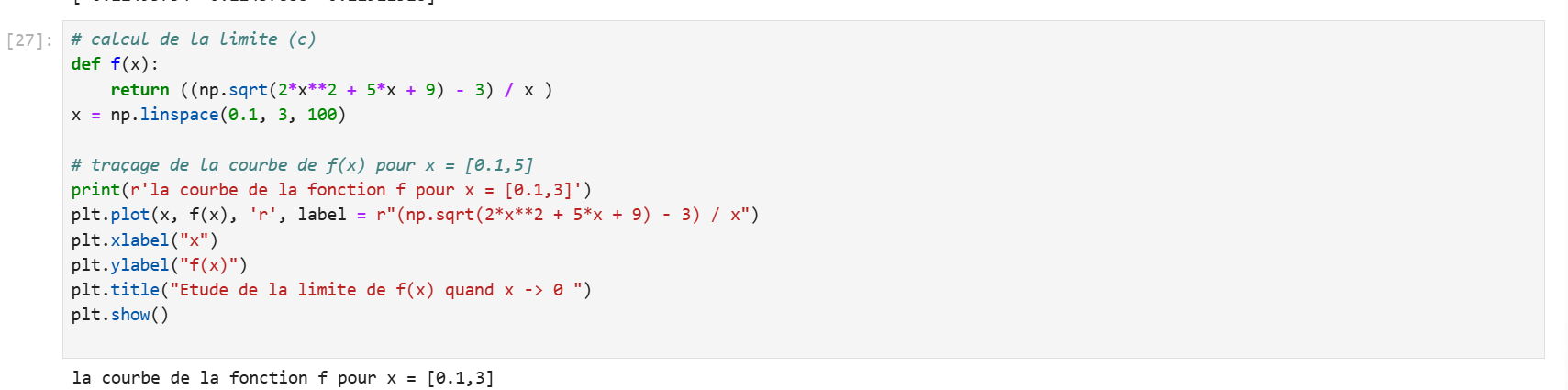


La limite à droite aussi tend vers 12493

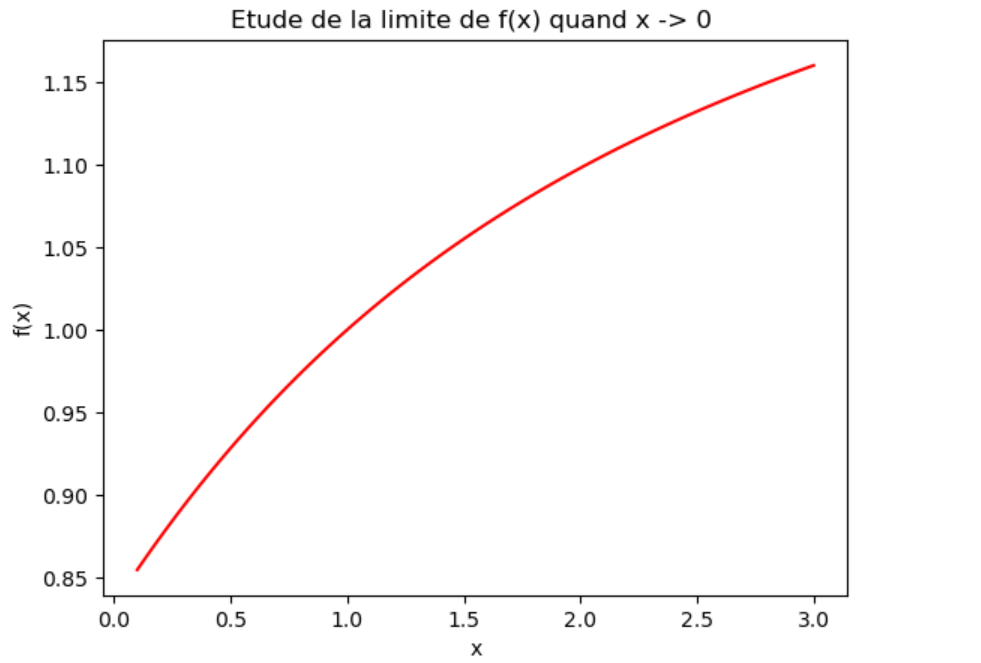
Alors avec une approximation on peut conclure que la limite quand x tend vers 0 d’est égale à 125 qui est égale a

* en 0

La définition et le traçage de la courbe se fais comme suite

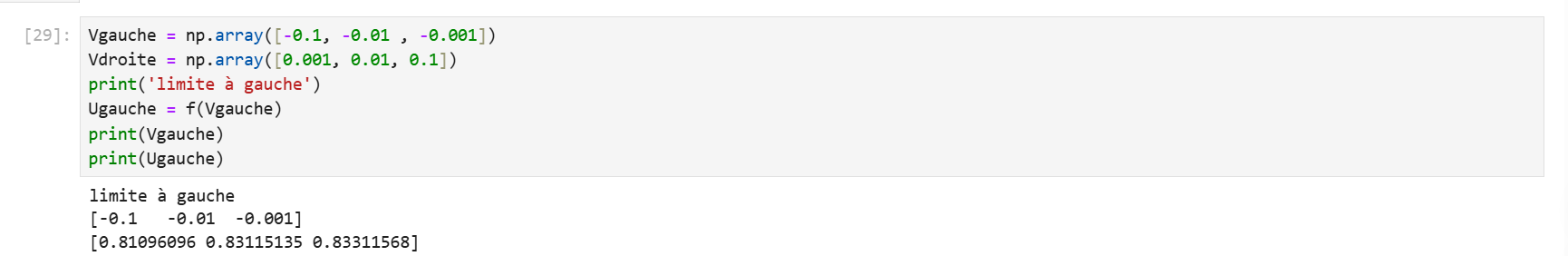


On a



Ensuite on calcule la limite a gauche et a droite

La limite a gauche nous donne :



A gauche la fonction tend vers 0.8333

La limite à droite nous donne



A droite la fonction tend vers 0.8333

* Comme la limite a gauche est égale à la limite à droite qui est égale a 0.8333 alors **la limite de en 0 nous donne 0.8333=**

# 1.3.2 Utilisation de sympy

On arrive a calculer les limite grâce au nombre dériver c’est à dire soit

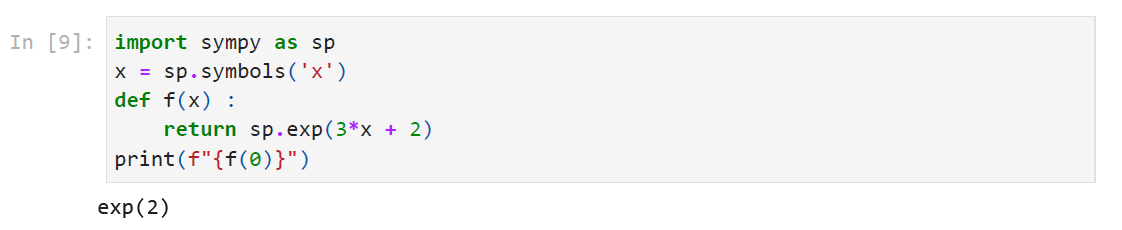
EE

x*f(x)*

f étant continue et dérivable en un point x0 , pour évaluer la limite

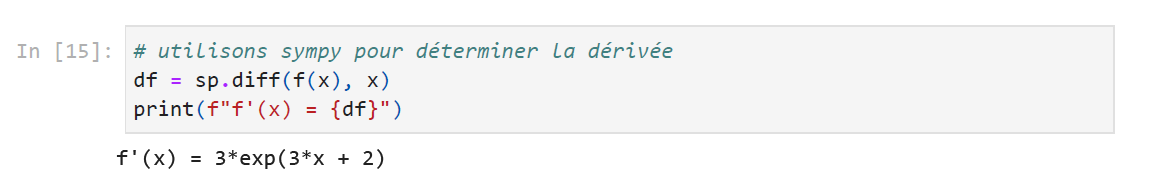
Quand x tend vers *x0* de il suffit juste de calculer la dérive de f en ce point c’est à dire  **=**

En python ca se calcule comme suite :



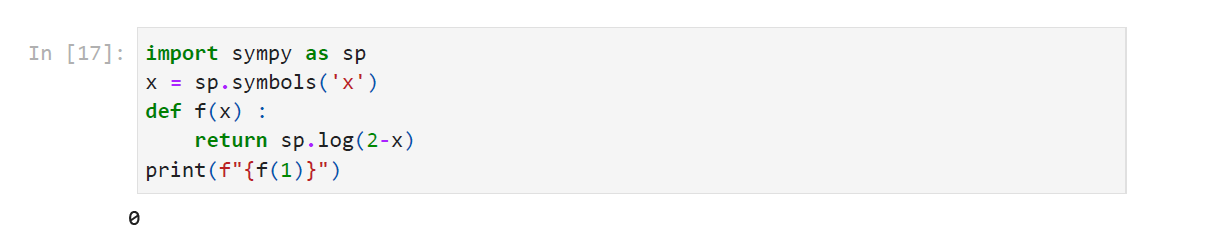
Si on pose une fonction *f(x)=*  alors  *=*

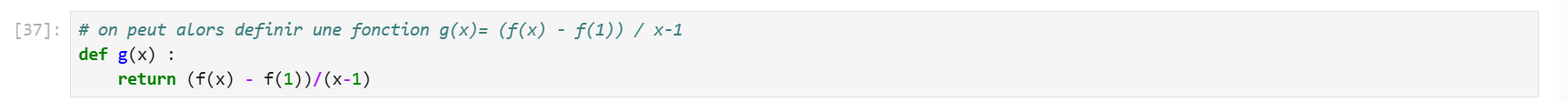
Avec sympy on calcule la derivé de f



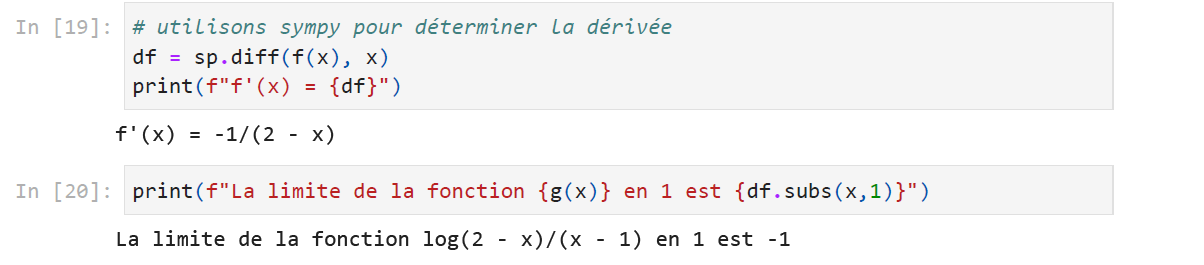
Alors

En suivants le même raisonnement on a :

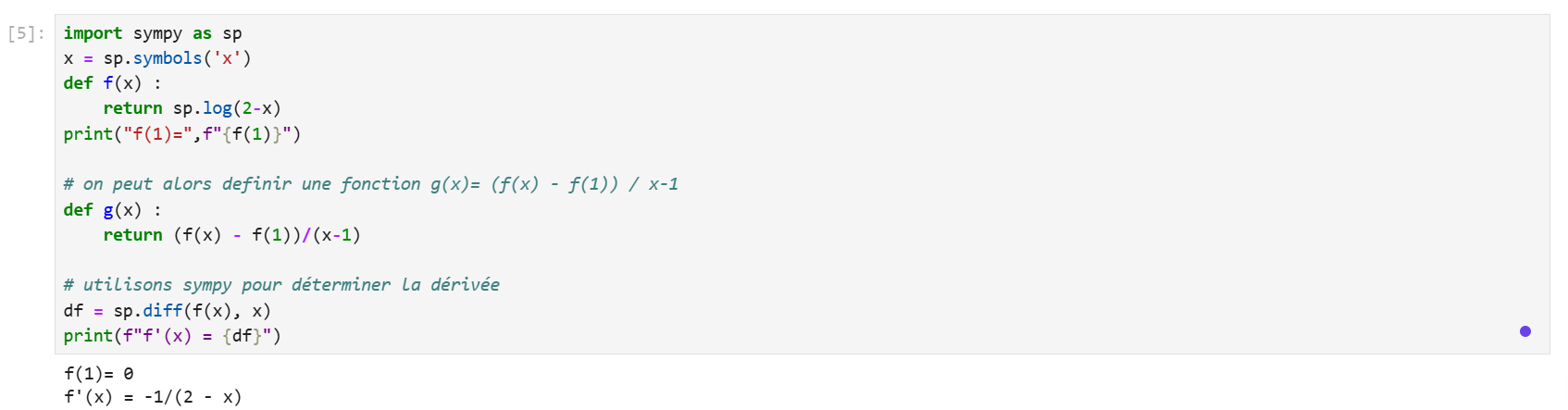




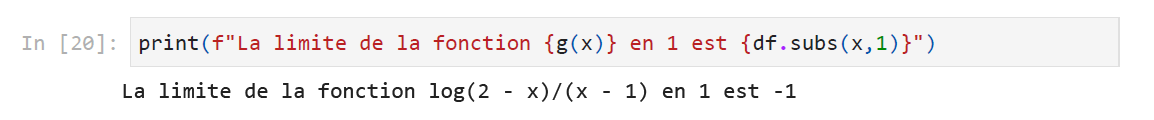
Déterminer la limite de la fonction g(x) en 1 revient à déterminer la dérivée de f(x) en 1



En suivant le même raisonnement on a Même raisonnement on a

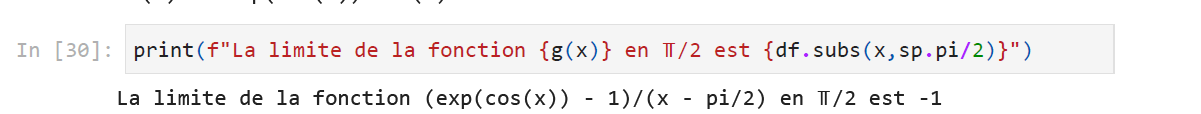


Alors on a





Et on a



# CONCLUSION

En conclusion, ce rapport a démontré l'efficacité et la polyvalence des outils numériques Python, notamment NumPy, SymPy, SciPy et Matplotlib, pour aborder une variété de problèmes mathématiques et scientifiques. À travers des exercices pratiques, nous avons appliqué ces bibliothèques pour modéliser la concentration d'un médicament dans le sang, analyser ses propriétés (dérivées, limites), et visualiser son comportement. La résolution d'équations, l'étude des limites et la représentation graphique ont mis en évidence la capacité de ces outils à simplifier des calculs complexes et à fournir des interprétations claires des résultats. En somme, ces bibliothèques constituent un ensemble d'outils indispensables pour toute analyse numérique et scientifique, offrant des solutions robustes pour la recherche, l'ingénierie et l'enseignement.