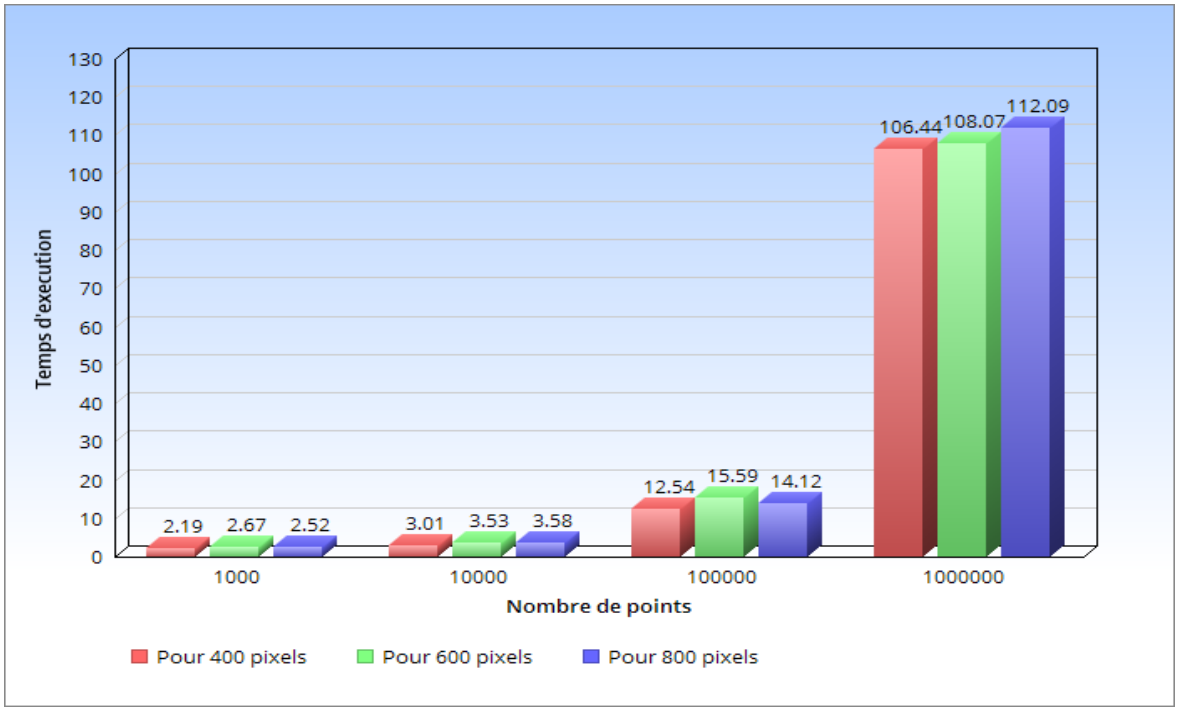
**Tous les programmes sont exécutés dans un pc avec 4GB de RAM et un processeur i5 sur un WSL.**

## Respect des consignes:

**. Le programme simulator.py donne une approximation de pi et approximate_pi.py crée un gif comme demandé avec au milieu l'approximation de pi qui contient des segments en gris ou noir selon le chiffre qu'on veut afficher (un peu comme le principe de l'afficheur 7 segments)**

**.Le programme convert a été utilisé pour créer le gif puis le module simulator pour créer "un générateur" de points aléatoires. Le simulateur a reçu un 10 par pylint mais pour le simulateur on a réussi à atteindre 9.82 faute de la multitude des variables locales et lignes utilisés en la fonction qui trace les chiffres de l'approximation de pi.**

**.On a recapitulé les performances du programme dans un graphe, pour cela on fixe la précision à 5(elle n'a pas beaucoup d'effet sur le temps d'exécution) :**




**On remarque que le temps d'exécution est quasi linéaire par rapport au nombre de points.**

**.On a décidé de ne pas créer une liste de points mais plutôt d'utiliser un générateur de points ce qui nous a permis d'avoir pour un test de 800 pixels, 1 million points et 4 dans la précision un maximum de 110 mégaoctets de mémoire(ceci a été regardé en utilisant le logiciel htop).**

**.Les images PPM 800*800 créées avaient chacune 1.8 mégaoctets, pour le gif c'est 0.9 mégaoctets.**

**.-Une exception a été lancé pour indiquer le format des arguments (faut donner exactement 3). Une autre au cas où l'utilisateur donne quelque chose qui n'est pas un entier. Puis on attrape finalement un ValueError si les arguments entrés sont négatifs**

**-Les f-string ont été utilisés dans tout cas de besoin.**

**-Le subprocess a été utilisé pour générer notre gif.**