Structure d'un programme :

1- Importation des bibliothèques

2 – Définition de toutes les constantes de l'énoncé : T, Veq, V0 et les demi-étendues associées D_Veq, D_V0, N le nombre de tirage souhaité pour un calcul Monté-Carlo…

ainsi dans la suite du programme : aucune valeur numérique !!! Il faut appeler les constantes par leur nom

Par exemple :

on n'écrit pas : v_MC=np.random.uniform(9.1,10.1,10000)

on écrit vv=np.random.uniform(V0-D_V0,V0+D_V0,N) ou vv=V0 + D_V0* np.random.uniform(-1,1,N)

3- Donner une instruction : calcul de C... une régression linéaire… calcul Monté Carlo...

Ex : Si on veut calculer C avec la célèbre formule (hélas fausse dans ce DM) : C = Cper*Veq/(V0)

- Avec un **array** Veq, c'est simple ! (Mais certainement moins élégant)

```
C = Cper*Veq/(V0)                # génère directement un array avec C
```

- Avec une **liste** Veq, on fait une boucle

```
C=[ ] # on initialise une liste vide, en dehors de la boucle bien sûr puis on fait la boucle
for i in range(len(Veq)): #len(Veq) correspond au nombre d'éléments dans la liste Veq
    C.append(Cper*Veq[i]/(V0)) #on ajoute à la liste C le terme Ci calculé à chaque itération
```

4- Demander au programme d'afficher quelque chose : 1 graphe et/ou une valeur...

- Pour un graphe, il faut le titre du graphe et les titres des axes :

```
plt.plot(t,C,'b*')
plt.title('C = f(t)')
plt.xlabel('t (s)')
plt.ylabel('C (mol/L)')
plt.grid() #parce que c'est plus joli
plt.show()
```

- Pour afficher un résultat : commande print

```
print("l'incertitude type est :", u_LNC)
print("k=",-p[0],"s^-1")
```