Algo

Utilisateur entre une équation ex: "2x² + 4x +4"
consigne: Résoudre l'équation grâce à un module python

Type d'écriture d'équation:
Cas positif et négatif
a,b ou c peut être négatif

+	-
2x ² + 4x + 4	-2x ²+ 4x- 4

1- Comment récupérer A, B et C dans une chaîne de caractère?

"2x² + 4x +4"

1-2 Récupérer C

Nous allons commencer par récupérer C car il est plus facile à récupérer

Nous remarquons que **C** est en dernière position.

equation[-1] ⇒ "C"

On a juste a utiliser `int()` pour changer la chaine de caractère en integer

Le signe de C est important dans le calcul de Δ

$4x^2 + 4x - 2$

EX signe termes négatifs: $\Delta = b^2 - 4ac = 4^2 - 4 * 4*(-2)$

$= 48$

$\neq 4^2 - 4 * 4* 2 = -16$

Δ< 0 peut faussé le calcul

Δ< 0 ⇒ pas de solution

Pour savoir si C est 0> ou 0 < on regarde l'élément d'avant

X[-2] ⇒ "+" ou "-"

Cas positif

SI le caractère d'avant est "+" alors on a juste a récupérer C et on le transforme en int

```
if x[-2]=="+":
    c = int(x[-1])
    return c
```

Cas négatif

SI le caractère d'avant est "-" alors on récupère C on le transforme en int et on utilise `-abs()` pour le indique à Python que c'est un entier négatif

```
>>> -abs(5)
...
-5
```

Cette portion de code nous montre comment on fait pour détecter le cas négatif

```
if x[-2]=="-":
    c = int(x[-1])
    s=-abs(c)
    return s
```

1-3 Récupérer B

Pour récupérer B nous allons procéder de la même façons que pour C mais nous allons effacer les caractères après B afin d'avoir B à la fin de la chaîne.

Le but est d'obtenir ceci:

"2x²4x-4" ⇒ "2x²+4"

cela nous permet d'avoir B et son signe en position [-1] et [-2]

pour effacer le dernier caractère nous allons utiliser [:-1]
et faire une boucle pour effacer les 3 derniers caractères

```
for i in range(3):
    x = x[:-1]
```

puis on applique la fonction qui permet de récupérer C car elle permet de récupérer le dernier caractère et de vérifier le signe

```
def recup_c(x):
    if x[-2]=="+":
        c = int(x[-1])
        return c

    if x[-2]=="-":
        c = int(x[-1])
        x=-abs(c)
        return  x
```

1-4 Récupérer A

Nous pouvons voir que a est en position 0 quand il est positif

Cas positif

equation=”2x ² + 4x + 4”

equation[0] ⇒”2“

Cas négatif
A est est en position 1 s’il est négatif

équation=”-2x ² + 4x + 4”

[0] ⇒ “-”

équation[1] ⇒ 2

pour résoudre ce problème on va regarder si en position [0] on a un “-” si c’est le cas on va récupérer le caractère suivant le changer en integer et changer son signe avec `-abs()`

s’il est positif il y aura rien avec A comme dans l'équation suivante: “2x ² + 4x + 4” où il y a rien avant le 2 alors on le récupère et on le change en integer

Si l’utilisateur entre un “+” avant A alors il va simplement récupérer la valeur d’après et le changer en int

2- Utiliser delta sur avec A,B,C

Maintenant que nous avons recuperer A,B et C on peut maintenant appliquer Delta

2-1 Comment calculer delta

Résolution d'une équation du second degré

Calcul de Δ : $\Delta = b^2 - 4ac$

Déduction de l'ensemble des solutions de l'équation :

En fonction de la valeur de Δ , nous avons 3 cas distincts :

Si $\Delta > 0$:
L'équation admet **deux solutions dans R** :

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

Si $\Delta = 0$:
L'équation admet **une unique solution dans R** :

$$x = \frac{-b}{2a}$$

Si $\Delta < 0$:

Pas de solutions

www.mathsbook.fr

2-2 version python du calcul de delta

```
def CalcRacine(a,b,c):
    delta = (b**2)-4*a*c

    if delta >0:

        racine1=-b+sqrt(delta)/2*a
        recine2=-b-sqrt(delta)/2*a
        return racine1, recine2

    if delta==0:
        racine=-b/2*a
        return racine

    if delta<0:
        return "pas de solution"
```

Le programme est terminé et est capable de résoudre des équations écrites sous les formes suivantes:

```
"2x²-9x-5"
```

```
"2x**2-9x-5"
```

```
"2x^2-9x-5"
```

et même avec des espaces

```
"2x² - 9x - 5"
```

```
"2x * *2 - 9x-5"
```

```
"2x^2-9x-5"
```

pour cela on utilise le module `replace` présent dans le built-in de python qui permet de remplacer un caractère par un autre

```
>>> a="2x² - 9x - 5"
>>> a.replace(" ","")
'2x²-9x-5'
```

ici on remplace un espace `" "` par `""`

La suppression d’espace est primordial pour le bon fonctionnement de notre programme car un espace mal placé viendrais rendre inutilisable notre programme