PYTHON : LES INSTRUCTIONS ET BOUCLES CONDITIONNELLES

EXERCICE 1 (Fonction simple)

Programmer la fonction $x \mapsto x^2 - 3x + 1$. Évaluer cette fonction en 0, en 1 et en 0,5.

EXERCICE 2 (Trinôme)

On considère l'équation du second degré (on suppose $a \neq 0$) : $ax^2 + bx + c = 0$.

1) Programmer une fonction qui renvoie le nombre de racines réelles distinctes en fonction de a, b et c.
2) Tester avec des exemples pour les trois cas possibles.
3) Tester avec les trinômes $x^2 + 1.4x + 0.49$ et avec $x^2 + 0.2x + 0.01$, commenter.

EXERCICE 3 (Pythagore)

Programmer une fonction qui renvoie si un triangle est rectangle en fonction des longueurs de ses trois côtés.

Si le triangle est rectangle, le programme renvoie **True**, et sinon **False**.

Tester avec un triangle rectangle (en donnant les côtés dans des ordres différents) et avec un triangle non rectangle.

EXERCICE 4 (Trouver l'erreur)

Corriger le programme suivant :

```
def compter(n):
  i = 1
  while i <= n:
    print(i)
  print("fini !")
```

EXERCICE 5

1) Écrire un programme qui affiche à l'écran les nombres de 1 jusqu'à 1000 (inclus).
2) Écrire un programme qui affiche à l'écran tous les nombres pairs compris entre 1 et 1000 (inclus).

EXERCICE 6 (Devinette)

Pour demander à l'utilisateur de saisir un nombre entier au clavier et l'enregistrer sous la variable `x`, on utilise les commandes

```
x = int(input("Entrer un nombre entier"))
```

La fonction **randrange**(`<debut>`,`<fin>`) de la bibliothèque **random** permet de générer un nombre entier aléatoire dans l'intervalle ⟦`<debut>`,`<fin>-1`⟧.

On veut créer un programme qui

- génère un nombre entier aléatoire entre −10 000 et 10 000,
- demande à l'utilisateur de deviner cet entier,
  - si l'utilisateur a trouvé, c'est terminé,
  - sinon, lui indique si son choix est trop grand ou trop petit et lui demande un nouveau nombre,
- recommence jusqu'à ce que l'utilisateur ait trouvé le bon nombre.

1) Écrire le programme et le tester.
2) Compléter ensuite ce programme pour qu'il renvoie à la fin le nombre d'essais dont a eu besoin l'utilisateur pour trouver le résultat.
3) Avec la commande **break**, prévoir l'arrêt du jeu si le joueur rentre 0 (il ne faut pas que la fonction puisse faire chercher le nombre 0).

Complément : Faire un programme sur le même modèle, mais pour lequel c'est l'ordinateur qui cherche un nombre que l'on rentre. Il renvoie le nombre d'essais nécessaires pour trouver ce nombre.
Le nombre entier à chercher est *quelconque* (il n'est pas dans un intervalle prédéfini).

EXERCICE 7

Programmer une fonction qui détermine le maximum de trois nombres (on peut imbriquer des branchements **if**).
Tester avec les couples $(1,2,3)$, $(3,1,2)$ $(-5,-2,-3)$ et $(1,4.2,4.2)$.