ALGORITHMIQUE ET PROGRAMMATION
EXERCICES

Exercice 1 Signe d'images

Soit f la fonction affine définie par $f(x)=3x-7$.
Écrire un programme qui affiche le signe de l'image d'un nombre x entré par l'utilisateur.
Affichage souhaité du type : f(-4) est positif.

Exercice 2 Approximation d'un extremum par balayage [*]

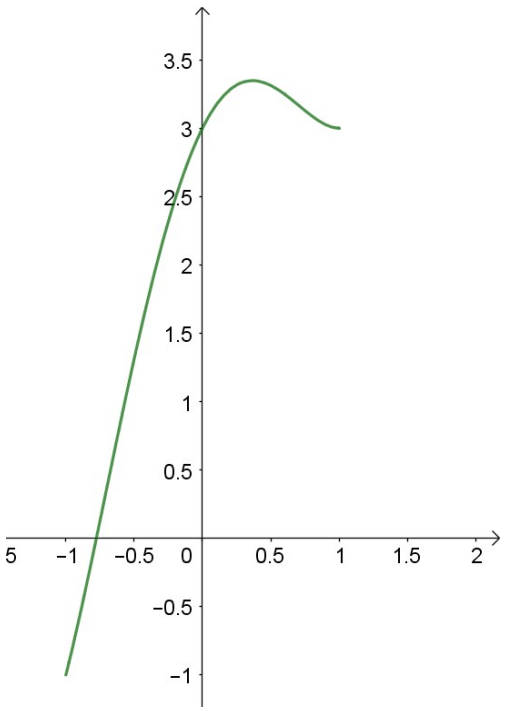
On considère la fonction f définie sur $\mathbb{R}$ par $f(x)=-x^4+3x-5$.
On admet que f est strictement croissante puis strictement décroissante, et admet donc un maximum noté α.

Puisque f est strictement croissante puis strictement décroissante, on se propose de « balayer » la courbe avec deux points A et B : au début, l'ordonnée de A est strictement inférieure à celle de B puisque f croît. À partir du moment où on aura l'ordonnée de A qui sera strictement supérieure à celle de B, cela voudra dire que f change de sens de variation, autrement dit qu'on a trouvé une valeur approchée du maximum.
C'est ce qu'on appelle *la méthode par balayage*.

Programmer cette méthode, puis tester le programme en donnant une valeur approchée à 10^{-8} près du maximum α.

Exercice 3 Longueur d'un arc de courbe [*]

Écrire un algorithme qui permette de calculer la longueur de la courbe représentative de la fonction f définie sur $[-1;1]$ par $f(x)=x^4-3x^2+2x+3$.



On pourra utiliser les deux fonctions suivantes :

```
def f(x) :
        return(x**4-3*(x**2)+2*x+3)
```

```
from math import sqrt
def distance(x1, x2, y1, y2) :
        return( sqrt( (x1-x2)**2+(y1-y2)**2 ) )
```

Exercice 4 Équation réduite [*]

Écrire un algorithme qui affiche l'équation réduite d'une droite (AB), où A et B sont deux points dont on connaît les coordonnées.

Exercice 5 Tester si un nombre est premier [*]

On souhaite créer un programme qui détermine si un nombre donné est premier.
Voici la fonction que l'on souhaite utiliser :

Compléter l'algorithme et le tester avec 7 ; 29 ; 91 ; 529.

```
def premier(nb):
    compteur = 1
    reponse = True
    while … :
        if … :
            reponse = False
        compteur = compteur + 1
    return …
```

Exercice 6 Diviser pour mieux régner

On souhaite créer une fonction `diviseur` qui compte le nombre de diviseurs d'un nombre entier.
Pour cela, on a écrit un algorithme en langage naturel :

```
fonction diviseur(n):
        compteur ← 0
        Pour div allant de 1 à n
                Si n est divisible par div alors
                        compteur ← compteur + 1
                Fin Si
        Fin Pour
        Retourner compteur
```

1. Écrire le script correspondant en langage Python.

2. Avec le programme, compléter les phrases suivantes :

12 admet ..... diviseurs	36 admet ..... diviseurs
13 admet ..... diviseurs	60 admet ..... diviseurs
15 admet ..... diviseurs	90 admet ..... diviseurs

3. a) Modifier le programme pour qu'il retourne l'entier inférieur à 1 000 qui a le plus de diviseurs.

b) Modifier alors le programme pour qu'il affiche ces diviseurs.

4. Modifier le programme afin de pouvoir compléter le tableau suivant, qui donne le nombre d'entiers inférieurs à 1 000 qui ont *x* diviseurs.

Nombre de diviseurs	1	2	3	4	5	6	7	8	9	10
Effectif										

Si ce sujet vous intéresse, vous pouvez lire un de mes articles sur mon site :
www.mathemathieu.fr/art/articles-maths/33-theorie-probabiliste-nombres-thm-fondateurs

[*] : explicitement au programme