ALGORITHMIQUE ET PROGRAMMATION
EXERCICES : **CORRECTION**

Exercice 1 Signe d'images

```
from math import * #pour pouvoir rentrer un nombre comme une racine carrée ou le nombre pi

def f(x):
    return 3*x-7

x = eval(input("Entrez un nombre:"))
if f(x)>=0:
    signe = "positif"
else:
    signe = "negatif"
print("f(",x,") est ",signe)
```

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

```
def f(x):
    return -x**4+3*x-5

def maximum_balayage(f,a,pas,precision): #f=fonction, a est le nombre auquel on commence à balayer la courbe, pas est le pas avec lequel on balaye, precision est le nombre de décimales que l'on affichera pour le résultat
    x1, x2 = a, a + pas
    while f(x1)<f(x2):
        x1, x2 = x2, x2 + pas
    return (x1,round(f(x1),precision))

print(maximum_balayage(f,-10,10**(-5),8))
```

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

```
from math import sqrt
def f(x):
    return x**4-3*(x**2)+2*x+3

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

def longueur_arc(f,a,b,nb_segments): #f=fonction, a et b forment l'intervalle sur lequel on mesure l'arc de courbe, nb_segments est le nombre de segments avec lequel on balaye la courbe
    long = 0
    pas = (b-a)/nb_segments
    x1, x2 = a, a + pas
    while x2<b:
        long = long + distance(x1,x2,f(x1),f(x2))
        x1, x2 = x2, x2 + pas
    return long

print(longueur_arc(f,-1,1,1000000))
```

→ 5.397108072189678

Exercice 4 Équation réduite [*]

```
def equation_reduite(A,B):
    xA,yA=A
    xB,yB=B
    if A==B:
        aff = "Une droite est formée par deux points DISTINCTS, ce qui n'est pas le cas ici."
    elif xA==xB:
        aff = "Droite verticale d'équation x="+str(xA)
    elif yA==yB:
        aff = "Droite horizontale d'équation y="+str(yA)
    else: #donc xA différent de xB, et yA différent de yB
        m = (yB-yA)/(xB-xA) #coeff. directeur
        p = yA-m*xA
        aff = "Droite d'équation réduite y={}x+{}".format(m,p)
    return aff

A, B = (4,31), (7,-43)
print(equation_reduite(A,B))
```

→ Droite d'équation réduite y=-24.666666666666668x+129.66666666666669

Exercice 5 Tester si un nombre est premier [*]

```
def premier(nb):
    if nb<=1:
        return "Le nombre testé doit être supérieur ou égal à 2."
    compteur = 2
    reponse = True
    while compteur<nb and reponse:
        if nb%compteur==0:
            reponse = False
        else:
            compteur = compteur + 1
    return str(nb)+" est premier." if reponse else str(nb)+" n'est pas premier (divisible par "+str(compteur)+")."

print(premier(529))
```

→ 529 n'est pas premier (divisible par 23).

Exercice 6 Diviser pour mieux régner

1.

```
def diviseur(n):
    compteur = 0
    for div in range(1,n+1):
        if n%div==0:
            compteur += 1
    return compteur
```

2.

12 admet 6 diviseurs	36 admet 9 diviseurs
13 admet 2 diviseurs	60 admet 12 diviseurs
15 admet 4 diviseurs	90 admet 12 diviseurs

3. a)
```
def max_div(nb): #retourne l'entier inférieur à nb qui a le plus de diviseurs
    maxi = 1
    for i in range(1,nb+1):
        if diviseur(i)>maxi:
            maxi = diviseur(i)
            entier = i
    return "L'entier inférieur à {} qui a le plus de diviseurs est {} avec {} diviseurs.".format(nb,entier,maxi)

print(max_div(1000))
```

→ L'entier inférieur à 1000 qui a le plus de diviseurs est 840 avec 32 diviseurs.

b)
```
def aff_diviseurs(n): #affiche tous les diviseurs du nombre n
    compteur = 0
    for div in range(1,n+1):
        if n%div==0:
            print(div)
    return "liste des diviseurs ci-dessus"

def max_div(nb): #retourne l'entier inférieur à nb qui a le plus de diviseurs
    maxi = 1
    for i in range(1,nb+1):
        if diviseur(i)>maxi:
            maxi, entier = diviseur(i), i
                return "L'entier inférieur à {} qui a le plus de diviseurs est {} avec {} diviseurs.".format(nb,entier,maxi),aff_diviseurs(entier)
#remarque : il est tout à fait possible qu'il y ait plusieurs nombres "gagnants", seul le dernier est retenu ici.

print(max_div(1000))
```

4.
```
def compteur_diviseurs(nb): #classifie les entiers inférieurs à nb en fonction du nombre de diviseurs
    nb1, nb2, nb3, nb4, nb5, nb6, nb7, nb8, nb9, nb10 = 0,0,0,0,0,0,0,0,0,0
    for i in range(1,nb+1):
        if diviseur(i)==1:
            nb1 += 1
        elif diviseur(i)==2:
            nb2 += 1
        elif diviseur(i)==3:
            nb3 += 1
        elif diviseur(i)==4:
            nb4 += 1
        elif diviseur(i)==5:
            nb5 += 1
        elif diviseur(i)==6:
            nb6 += 1
        elif diviseur(i)==7:
            nb7 += 1
        elif diviseur(i)==8:
            nb8 += 1
        elif diviseur(i)==9:
            nb9 += 1
        elif diviseur(i)==10:
            nb10 += 1
    return nb1, nb2, nb3, nb4, nb5, nb6, nb7, nb8, nb9, nb10

print(compteur_diviseurs(1000))
```

→ (1, 168, 11, 292, 3, 110, 2, 180, 8, 22)

Nb de diviseurs	1	2	3	4	5	6	7	8	9	10
Effectif	**1**	**168**	**11**	**292**	**3**	**110**	**2**	**180**	**8**	**22**