# MATHEMATIQUES
## Python : les fonctions

Dans un programme, il est possible d'écrire de petits sous programmes intermédiaires appelés fonctions.
Une fonction porte un nom et utilise zéro, une ou plusieurs variables appelées paramètres.

La syntaxe :
```
def nom_de_la_fonctio(paramètre1,paramètre2,etc):
    instruction(s)
    return resulat
```

L'instruction "return" permet de renvoyer une valeur de type entier, décimal ou chaîne de caractères qui peut être utilisée dans un autre programme ou une autre fonction. Elle interrompt le programme dès qu'elle s'est exécutée.

Exemple :
```
1 def somme_carres(a,b):
2     somme=a**2+b**2
3     return somme
```

Le nom de cette fonction est somme_carres. Elle a deux paramètres a et b. Si on tape l'instruction somme_carres(10,2) dans la console Python, on obtient l'affichage .....

## Exercice 1

**1.** On a écrit le script de la fonction exerice1.
```
def exercice1(a) :
    a=a**2-a+1
    return a
```
Combien de paramètres cette fonction possède-t-elle ? Le(s)quel(s) ?
Que renvoie l'instruction `>>> exercice1(2)` ? et `>>> exercice1(5)` ?

**2.** On a programmé une fonction $f$ :
```
def f(a,b,c) :
    return (a**2+b**2-c**2)
```

**a.** Que renvoie `>>> f(3,4,5)` ? et `>>> f(5,4,3)` ?

**b.** Que renvoie $f(a,b,c)$ lorsque $a$, $b$ et $c$ sont les longueurs d'un triangle rectangle ?

**c.** Compléter le programme de la fonction trirec, en faisant appel à la fonction $f$, afin que la fonction trirec renvoie si le triangle $ABC$ de côtés $a$, $b$, $c$ est ou non un triangle rectangle (or est l'écriture en Python du connecteur ou).
```
def f(a,b,c) :
    return (a**2+b**2-c**2)


def trirec(a,b,c):
    if ..... or ..... or ..... :
        return ("...........................")
    else :return ("........................")
```

**3.** On souhaite créer une fonction diviseur qui compte le nombre de diviseurs d'un nombre entier. Pour cela on dispose d'un algorithme en langage naturel.

**Fonction** Diviseur(n)
**1.** *compteur* $\longleftarrow$ 10
**2.** Pour Diviseur allant de 1 à $n$
**3.** Si $n$ est divisible par Diviseur alors
**4.** *compteur* $\longleftarrow$ *compteur* $+ 1$
**5.** Fin Si
**6.** Fin Pour
**7.** Retourner *compteur*

**a.** Ecrire le script correspondant à l'algorithme en langage Python ($a\%b$ donne le reste de la division euclidienne de $a$ par $b$).

**b.** Quelles sont les valeurs retournées par la fonction diviseur lorsque $n = 13$, $n = 15$, $n = 36$ ?

## Exercice 2

On appelle *nombre parfait* un nombre qui est égal à la somme de tous ses diviseurs sauf lui-même. Par exemple 6 est un nombre parfait car ses diviseurs sont 1, 2, 3 rt 6 et $6 = 1 + 2 + 3$.
On considère l'algorithme écrit en langage naturel. Il donne tous les nombres parfaits entre 1 et 10 000.

**1.** Pour $n$ allant de 1 à 10 000
**2.** $somme \longleftarrow 0$
**3.** Pour $i$ allant de 1 à $n - 1$
**4.** Si reste de la division euclidienne de $n$ par $i$ est égal à 0 alors
**5.** $somme \longleftarrow somme + i$
**6.** Si *somme* est égal à $n$ alors
**7.** Afficher "$n$ est parfait"
**8.** Fin Pour

Ecrire le script correspondant en langage Python.

## Exercice 3

Afin de conjecturer les variations de la fonction $f$ sur l'intervalle $[a\ ;\ b]$, on propose le script de la fonction variation.

```
def variation(f,a,b):
    pas=(b-a)/100
    x,n=a,0
    while x<b:
        if f(x+pas)>=f(x):
            n=n+1
        x=x+pas
    return n
```

**1.** Que fait la boucle ?
**2.** Combien de fois la boucle est-elle parcourue ?
**3.** Si la fonction $f$ est croissante sur $[a\ ;\ b]$, quelle est la valeur renvoyée par la fonction variation ?
**4.** Supposons que pour une fonction particulière, la fonction variation renvoie 77. Que peut-on en déduire pour la fonction $f$ ?
**5.** Que penser de l'affirmation : « la fonction variation renvoie 0 donc la fonction $f$ est décroissante sur $[a\ ;\ b]$ » ?
**6.** Traduire le script de la fonction variation en un algorithme en langage naturel.

## Exercice 4

Soit $h$ un réel strictement positif. On considère un rectangle dont les dimensions sont $h$ et $\dfrac{1}{h}$.

On veut déterminer $h$ pour que le périmètre de ce rectangle soit le plus petit possible puis déterminer cette valeur minimale du périmètre. Pour cela, on propose le script suivant :

```
1 def perimetre(h):
2     return(2*h+2/h)
3
4
5 def minperimetre(a,b,n):
6     pas=(b-a)/n
7     x=a
8     val_min=a
9     val_min_perimetre=perimetre(a)
10    for i in range(1,n+1):
11        x=x+pas
12        y=perimetre(x)
13        if y<val_min_perimetre:
14            val_min_perimetre=y
15            val_min=x
16    print('valeur de h:',round(val_min,2),'valeur du perimetre:',round(val_min_perimetre,2))
```

**1.** Que fait le script à la ligne 16 ?

**2.** En saisissant dans la console l'instruction min(0,5,100), on reçoit un message d'erreur. Pourquoi ?

**3.** Quel sera l'affichage si on saisit min(0.1,10,100) dans la console ?

## Exercice 5

Pouvez-vous trouver un (ou des) entier(s) $n$ tels que :

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}>10$$

## Exercice 6

On souhaite déterminer une valeur approchée du minimum de la fonction $f$ définie sur [10 ; 100] par : $f(x)=x+\dfrac{900}{x}$.

Pour cela on utilise la méthode suivante :

Prendre de manière aléatoire $n$ valeurs différentes dans l'intervalle [10 ; 100[ en utilisant la fonction uniform(10,100) qui renvoie un nombre réel aléatoire entre 10 et 100, puis calculer leurs images par la fonction $f$ et les comparer pour en extraire la plus petite. Compléter le script suivant de façon que si on saisit dans la console minimum(longueur,10000) on obtient une réponse au problème de la forme$(x\,;\,f(x))$.

```
from random import uniform

def longueur(x):
    return(........)


def minimum(.....):
    min=f(.....)
    for i in range (....):
        x=uniform(........)
        if ............ :
            min=......
            val=....
    return val,min
```

## Exercice 7

**1.** On veut simuler cent lancers d'une pièce de monnaie équilibrée. on a commencé à écrire le script ci-contre qui compte le nombre de fois où l'on a obtenu Pile au cours des cent lancers. Compléter ce script.

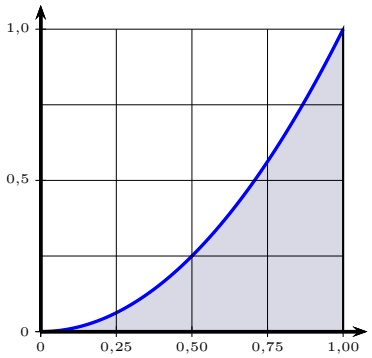
```
1 from random import randint
2 nbrepile=0
3 for k in range(.............):
4     p=randint(0,1)
5     if p==1:
6         nbrepile=.........
```

**2.** Modifier le script pour qu'il calcule la fréquence d'apparition de Pile au cours des cent lancers.

**3.** Améliorer le script en écrivant une fonction frequence_pile qui calcule la fréquence d'apparition de Pile pour un nombre de lancers choisi par l'utilisateur.

## Exercice 8

On considère la courbe de la fonction carré tracée ci-contre sur l'intervalle [0 ; 1] dans un repère orthonormé.
On a écrit le script d'une fonction aire qui permet de calculer une valeur approchée de l'aire grisée.




```
1 from random import *
2 from math import*
3 def aire(n):
4     d=0
5     for i in range(n):
6         x=random()
7         y=random()
8         if y<=x**2:
9             d=d+1
10    return(d/n)
```

**1.** Décrire ce que font les instructions des lignes 6 et 7.

**2.** Décrire ce que fait l'instruction conditionnelle des lignes 8 et 9 et ce qui est compté avec la variable *d*.

**3.** A partir de la console, on obtient l'affichage `>>> aire(1000000)` `0.334344` . Que peut-on en déduire ?