ALGO 3 – NOTION DE FONCTION

I Définir et appeler une fonction

En programmation, les **fonctions** sont semblables aux fonctions mathématiques. À une valeur x, la fonction va retourner une valeur y.

EN MATHÉMATIQUES.

- **Définir** une fonction.

$$\forall x \in \underbrace{\ldots}_{\text{ens. de def}}, \quad f(x) = \underbrace{\ldots}_{\text{expression}}$$

- **Valuation** d'une fonction.

$$f(\underbrace{\ldots}_{\text{pt où on évalue}}) = \underbrace{\ldots}_{\text{résultat}}$$

EN INFORMATIQUE.

- **Définir** une fonction.

```
def f(x):
    return(...)
```

- **Valuation/appel** d'une fonction.

```
Entrée [1]: f(...)

Out [1]: ...
```

La différence principale entre une fonction mathématiques et une fonction en informatique est l'absence d'ensemble de définition en informatique. Décortiquons maintenant la syntaxe nécessaire à la bonne définition d'une fonction en Python en prenant appui sur les deux lignes de code ci-dessus.

- La commande `def` permet d'indiquer à Python que l'on souhaite **définir une fonction**.
- La lettre `f` désigne le **nom de la fonction** et sera utilisée pour la valuation de la fonction. En mathématiques, on a l'habitude d'appeler très souvent nos fonctions f. En informatique, on essayera de donner des noms plus descriptifs à nos fonctions, comme par exemple cube si l'on code la fonction $x \mapsto x^3$.
- La variable `x` désigne l'**argument de la fonction**. Une fonction peut prendre un argument des objets de types différents (`int`, `float`, `list`, `str`) mais peut aussi avoir plusieurs arguments voir aucun (ce qui n'est pas possible en mathématiques)
- Enfin, la commande `return` permet d'indiquer l'**expression de la fonction**. Des instructions intermédiaires peuvent être effectués au sein de la fonction. Les variables définies au cours d'une fonction n'ont pas d'existence en dehors : on parle de variables locales.

En résumé, une fonction en informatique est une séquence d'instructions, dépendant de paramètres d'entrée, appelés arguments, et renvoyant un résultat. La synthaxe est la suivante.

```
def nomfonction(argument1, argument2, ...):
    instruction
    return(result)
# ici, on est hors de la définition de la fonction.
```

Une fois la fonction définie, on l'appelle à l'aide de la syntaxe :

```
Entrée [2]: nom_fonction(exp1, exp2,..., expn)
```

où `exp1, exp2,..., expn` sont les **expressions** auxquelles on veut évaluer la fonction.

```
Out [2]: .... #Python renvoie alors la valuation de la fonction
         .... #aux expressions correspondantes
```

II Exercices

Exercice 1 On considère la fonction (mathématique) $f : \mathbb{R} \to \mathbb{R}$ définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = x^2$$

Définir en Python cette fonction et calculer son évaluation en 2.

Entrée [3]:
```
def carre(x):
    return (x**2)
```

Entrée [4]:
```
carre(2)
```

Out [4]:
```
4
```

Entrée [5]:
```
carre(2+3)
```

Out [5]:
```
25
```

Exercice 2 On considère la fonction (mathématique) $f : \mathbb{R}^3 \to \mathbb{R}^3$ définie par

$$\forall (x, y, z) \in \mathbb{R}^3, \qquad f(x, y, z) = (x + y + z, 2x - y, xy)$$

Définir en Python cette fonction et calculer son évaluation au point $(1, 2, 3)$.

Entrée [6]:
```
#Manière compacte
def exemple(x,y,z):
    return(x+y+z, 2*x-y, x*y)

#Manière non compacte
def exemple(x,y,z):
    a=x+y+z
    b=2*x-y
    c=x*y
    return (a,b,c)
```

Entrée [7]:
```
exemple(1,2,3)
```

Out [7]:
```
(6,0,2)
```

Exercice 3 Définir en Python une fonction, qui prend en argument deux nombres réels et qui renvoie le produit de ces deux nombres réels.

Entrée [8]:
```
def multiplie(a,b):
    return(a*b)
```

Entrée [9]:
```
multiplie(2,4)
```

Out [9]:
```
8
```

Exercice 4 Définir en Python une fonction, qui ne prend aucun argument, et qui renvoie le message "Bonjour".

Entrée [10]:
```
def message():
    return ('Bonjour')
```

Entrée [11]:
```
message() #Même sans argument, l'appel comporte des parenthèses
```

Out [11]:
```
Bonjour
```

Exercice 5 Définir en Python une fonction, qui prend en argument un prénom, et qui renvoie le message "Bonjour [Prénom]".

Entrée [12]:
```
def bonjour(prenom):
    return('Bonjour', prenom)
```

Entrée [13]:
```
bonjour('Alice')
```

Out [13]:
```
Bonjour Alice
```

Exercice 6 On considère la fonction (mathématique) $f : \mathbb{R} \to \mathbb{R}$ définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = x^3 + \frac{1}{2}$$

Définir en Python cette fonction. Afficher son évaluation en 2.

Entrée [14]:
```
def f(x):
    return(x**3+1/2)
```

Entrée [15]:
```
f(2)
```

Out [15]:
```
8.5
```

Exercice 7 On considère la fonction (mathématique) $f : \mathbb{R}^2 \to \mathbb{R}^3$ définie par

$$\forall (x, y) \in \mathbb{R}^2, \qquad f(x, y) = (x + y, x - 2, xy + 1)$$

Définir en Python cette fonction. Afficher son évaluation en $(1, 2)$.

Entrée [16]:
```
def plusieursvariables(x,y):
    return(x+y, x-2, x*y+1)
```

Entrée [17]:
```
plusieursvariables(1,2)
```

Out [17]:
```
(3,-1,3)
```

Exercice 8 Écrire une fonction, qui prend en argument deux nombres et qui renvoie la moyenne de ces deux nombres. Afficher son évaluation au point $(11, 16)$.

Entrée [18]:
```
def moyenne(a,b):
    return((a+b)/2)
```

Entrée [19]:
```
moyenne(11,16)
```

Out [19]:
```
13.5
```

Exercice 9 On considère la fonction (mathématique) $f : \mathbb{R} \to \mathbb{R}$ définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = \begin{cases} 2x-1 & \text{si } x \leqslant 2 \\ \ln(x) & \text{si } x > 2 \end{cases}$$

Définir en Python cette fonction et calculer son évaluation en -4 et en e.

Entrée [20]:
```
import numpy as np

def parmorceaux(x):
    if x<=2:
        return (2*x-1)
    else :
        return(np.log(x))
```

Entrée [21]:
```
parmorceaux(-4)
```

Out [21]:
```
-9
```

Entrée [22]:
```
parmorceaux(np.e)
```

Out [22]:
```
1.0
```

Exercice 10 Écrire une fonction, qui prend en argument deux nombres et qui renvoie le plus grand de ces deux nombres. Afficher son évaluation au point $(101, 100)$.

Entrée [23]:
```
def plusgrand(a,b):
    if a>= b:
        return(a)
    else:
        return(b)
```

Entrée [24]:
```
plusgrand(101,100)
```

Out [24]:
```
101
```

Exercice 11 Écrire une fonction, qui prend en argument un nombre réel et renvoie sa valeur absolue, de deux manières différentes.

1. En utilisant la fonction `abs` de la bbliothèque `numpy`.
2. Sans utiliser la fonction `abs` de la bbliothèque `numpy`.

Entrée [25]:
```
#Première manière
import numpy as np
def valeurabsolue(x):
    return(np.abs(x))
```

Entrée [26]:
```
#Deuxième manière
def valeurabsolue(x):
    if x>= 0:
        return(x)
    else:
        return(-x)
```

Exercice 12 Écrire une fonction, qui prend en argument trois nombres réels a, b et c et qui renvoie le(s) racines du polynôme $x \mapsto ax^2 + bx + c$ (et qui renvoie un message 'Pas de racine' s'il en a pas). Afficher la valuation de cette fonction au point $(2, -6, 4)$.

Entrée [27]:
```
def racinepoly(a,b,c):
    Delta = b ** 2 - 4 * a * c
    if Delta > 0 :
        x1= (-b - np.sqrt(Delta)) / (2 * a)
        x2= (-b + np.sqrt(Delta)) / (2 * a)
        return(x1, x2)
    elif Delta == 0 :
        x0= (-b) / (2 * a)
        return(x0)
    else :
        return("Le polynome n'admet pas de racine réelle")
```

Entrée [28]:
```
racinepoly(2,-6,4)
```

Out [28]:
```
(1.0, 2.0)
```

Exercice 13

1. Définir une fonction f, qui prend en argument un nombre réel x et qui renvoie la valeur de $1 + \ln(x)$.

Entrée [29]:
```
def fonction(x):
    return (1+np.log(x))
```

2. On considère la suite définie par $u_0 = 2$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = f(u_n)$ où f est la fonction définie à la question précédente. Écrire un programme qui calcule et affiche les 5 premiers termes de la suite.

Entrée [30]:
```
u = 2 #valeur de u0
print(u)
u = fonction(u) #valeur de u1
print(u)
u = fonction(u) #valeur de u2
print(u)
u = fonction(u) #valeur de u3
print(u)
u = fonction(u) #valeur de u4
print(u)
```

Out [30]:
```
2
1.6931471805599454
1.5265890341390445
1.423035857164402
1.35279251707865
```

Exercice 14 Écrire une fonction, appelée `affichetablede7`, qui ne prend pas d'argument et qui affiche la table de multiplication de 7.

Entrée [31]:
```
def affichetablede7():
    return('7x1=7', '7x2=14', '7x3=21','7x4=28',
           '7x5=35','7x6=42','7x7=49','7x8=56','7x9=63','7x10=70')
```

Exercice 15 Écrire une fonction, qui prend en argument un entier `n` et qui affiche la table de multiplication de cet entier `n`.

Entrée [32]:
```
def affichetable(n):
    return(n, 'x1=', n*1 , n, 'x2=', n*2) #etc.
```

Exercice 16 Écrire une fonction, qui prend en argument un nombre réel (représentant une somme d'argent en euros) et une chaine de caractère (représentant une devise, soit le dollar, soit la livre, soit le yens) et qui renvoie la somme d'argent dans la devise souhaitée. *On donne les conversions suivantes : 1 euro = 1,15 dollars ; 1 euro = 0,81 livre et 1 euro = 130 yens).*

Entrée [33]:
```
def conversion(x,devise):
    if devise=='dollar':
        return(x*1.15)
    elif devise=='livre':
        return(x*0.81)
    else :
        return(x*130)
```

Entrée [34]:
```
conversion(10,'dollar')
```

Out [34]:
```
11.5
```