TP4

Création d'une fonction.

Python dispose d'une instruction « def » qui permet de créer une fonction. Cette instruction évite de réécrire systématiquement la fonction dans un programme. On pourra dès lors l'utiliser à tout moment dans le programme en appelant cette fonction.

Exemples :

Taper les lignes suivantes dans l'éditeur de programme et tester sur des exemples. Comprendre ce qui se passe !

a)

```
def carrée(x):    # On définit la fonction carrée.
    y=x**2
    return(y)        # La fonction s'arrête ici: l'instruction "return" indique cela.
x=eval(input("Donner x "))
print(carrée(x))
```

Autre façon :

```
def carrée(x):    # On définit la fonction carrée.
    y=x**2           # La fonction s'arrête ici: l'indentation à la ligne suivante indique cela.
x=eval(input("Donner x "))
print(carrée(x))
```

Cette fonction ne sert en fait à rien …

Autre façon :

```
def carrée(x):    # On définit la fonction carrée.
    print(x**2)          # La fonction affichera directement la valeur x^2 sans avoir besoin de le repréciser.
x=eval(input("Donner x "))
carrée(x)
```

Ici on obtient un affichage directement mais il est impossible de stocker l'information obtenue dans une autre variable. Une fonction sans return est en fait appelée une procédure.

Attention : on « sort » de la fonction dès la rencontre d'un return dans l'exécution du code.

Exemple :

```
def fonction(x) :
    y = x**2
    return y
    return 'patate'
```

Le return 'patate' ne sera jamais lu lors de l'appel de cette fonction.

b) Fonction avec conditions

```
def fonction_affine_par_morceaux(x):
    if x>5:
        y=2*x+1
    elif x>-1 and x<=5:
        y=-x+16
    else:
        y=x+18
    return(y)
x=eval(input("Entrer x "))
print(fonction_affine_par_morceaux(x))
```

c) Fonction qui appelle une fonction.

```
def carrée(x):
    y=x**2
    return(y)

def polynôme(x):
    y=carrée(x)+2*x+6
    return(y)
x=eval(input("Entrer x "))
print(polynôme(x))
```

d) Fonction utilisant plusieurs arguments (variables) :

```
def multivariables(x,y,z):
    t=x**2+y**3-z**4
    return(t)
x=eval(input("Entrer x "))
y=eval(input("Entrer y "))
z=eval(input("Entrer z "))
print(multivariables(x,y,z))
```

e) Fonction appelant la fonction elle-même : Fonction récursive.

1)

```
def factorielle(n):
    if n<=1:
        y=1
    else:
        y=n*factorielle(n-1)
    return(y)
n=eval(input("Entrer un entier natruel n "))
print(factorielle(n))
```

2) Suite de Lucas

```
def Lucas(n):                       # La suite de Lucas est définie
    if n==0:
        y=1                         # U0 = 1  et
    elif n==1:
        y=3                         # U1 = 3  et
    else:
        y=Lucas(n-1)+Lucas(n-2)     #  par Un = Un-1 + Un-2
    return(y)
n=eval(input("Entrer n "))
print(Lucas(n))
```

3) Calcul de la puissance d'un réel.

```
def puissance(x,n):
    if n==0:
        if x==0:
            print("impossible")
        else:
            return 1
    elif n%2==0:
        y=puissance(x,n//2)**2
    else:
        y=x*puissance(x,n-1)
    return(y)
x=eval(input("Entrer le réel x "))
n=eval(input("Entrer la puissance entière positive n "))
print(puissance(x,n))
```

4) Calcul d'une somme.

```
def somme(n):   # On va calculer la somme des inverses.
    if n==1:
        y=1
    else:
        y=1/n+somme(n-1)
    return(y)
n=eval(input("Entrer l'entier naturel non nul  n "))
print(somme(n))
```

5) Nombre de poignées de mains échangées entre n personnes :

```
def poignée(n):
    if n==1:
        y=0
    else:
        y=n-1+poignée(n-1)
    return(y)
n=eval(input("Combien de personnes? "))
print(poignée(n))
```

<u>Exercices</u> :

<u>Exercice 1</u> :
Ecrire un script (utilisant une fonction) qui permet de donner les images par f des réels compris dans l'intervalle I avec un pas de a dans chaque cas suivant.
a) $f(x) = x^2 + 2x - 1$ I = [- 2 ; 3] a = 0.5
b) $f(x) = \ln(x^2 + 1)$ I = [- 3 ; 3] a = 0.25
c) $f(x) = \ln(x + \sqrt{1 + x^2})$ I = [- 4 ; 4] a = 0.5

<u>Exercice 2</u> :
Ecrire une fonction triple et une autre fonction cube qui calculent respectivement le triple et le cube de leur argument. (Le tester pour voir s'il fait bien ce que vous voulez.)

<u>Exercice 3</u> :
Écrire un code qui vous donne la table de multiplication par 7.
A priori vu le cahier des charges du titre, on ne reçoit rien et on ne renvoie rien. Nous ne devons que produire un affichage. Il nous faut donc une procédure sans argument.

<u>Exercice 4</u> :
Produire les résultats des tables de multiplication de j à k. On décompose le travail à effectuer en parties :
— partie 1 : table(n) Afficher les résultats de la table de multiplication avec n
— partie 2 : table(j,k) Appeler les table(n) pour $j \leq n \leq k$ (on utilisera des fonctions imbriquées ou plutôt ici des procédures imbriquées).

<u>Exercice 5</u> : Voici une fonction, que fait-elle ? Expliquer votre démarche.

```
def test(x,y) :
    somme = 0
    for i in range(y) :
        somme = somme + x
    return somme
```

<u>Exercice 6</u> : Ecrire une fonction max3(a, b, c) qui renvoie le plus grand des trois nombres.

<u>Exercice 7</u> : Calculer, en utilisant des fonctions:

a) $\prod_{k=2}^{n} ln\left(1 - \frac{1}{k^2}\right)$, n ≥ 2 b) $A = \sum_{1 \leq i,j \leq n} \min(i;j)$, $B = \sum_{1 \leq i,j \leq n} \max(i;j)$, $C = \sum_{1 \leq i,j \leq n} |i - j|$.

<u>Exercice 8</u> **:**
Ecrire une fonction qui parcourt une liste de nombres L et qui dit si le nombre n est dans la liste L ou non en renvoyant un booléen.

<u>Exercice 9</u> :
La suite de Fibonacci est définie par $U_0 = U_1 = 1$ et pour tout $n \geq 0$, $U_{n+2} = U_{n+1} + U_n$.
Ecrire un script qui permet de calculer un terme de la suite de Fibonacci, puis un script qui permet de lister les n premiers termes de la suite de Fibonacci.

Exercice 10 :

a) Ecrire un script qui permet de calculer le coefficient binomial $\binom{n}{k}$ avec 0 ≤ k ≤ n :

1) Sans fonction. Utiliser l'instruction factorial de python (from math import*)
2) En créeant une fonction coef_bino qui utilise factorial.
3) En créant 2 fonctions : L'une retourne n ! – nommée facto(n) et l'autre utilisant cette fonction facto(n).

b) Je rappelle que l'on a $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$ si 1 ≤ k ≤ n. Ecrire un script qui permet de calculer $\binom{n}{k}$ avec une fonction récursive .

c) On a la relation de Pascal : $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$ avec 0 ≤ k ≤ n-1.

Ecrire un nouveau script qui permet de calculer $\binom{n}{k}$ de proche en proche :

1) sans fonction.
2) avec une fonction récursive.

Exercice 11 :

Th :

On considère la récurrence double

$aU_{n+2} + bU_{n+1} + cU_n = 0$, U_0 et U_1 donnés, a, b et c sont trois réels (a non nul).

On associe à cette relation le polynôme dit caractéristique $P(r) = ar^2 + br + c$

Si $\Delta = b^2 - 4ac > 0$ alors les suites (U_n) sont de la forme $U_n = \alpha r_1^n + \beta r_2^n$ où r_1 et r_2 sont les racines de P et α et β sont déterminés par U_0 et U_1.

Si $\Delta = b^2 - 4ac = 0$ alors les suites (U_n) sont de la forme $U_n = (\alpha n + \beta) r_1^n$ où r_1 est la racine de P et α et β sont déterminés par U_0 et U_1.

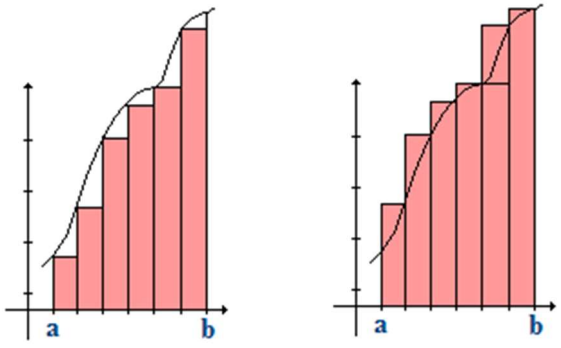
1) Ecrire une fonction qui permet de calculer des termes de la suite.
2) Ecrire un programme qui donne l'expression de U_n.

Exercice 12 **: Calcul approchée d'une aire.**

1) Méthode des rectangles.

On considère une fonction continue (intégrable) croissante et positive sur un intervalle [a ; b].

On encadre « l'aire sous la courbe » (pour x dans [a ; b]) par la somme des aires des rectangles sous la courbe et la somme des aires des rectangles au-dessus de la courbe.



Pour cela :

- On découpe l'intervalle [a, b] en n intervalles (n ≥ 1), pour former n rectangles.

On construit donc n +1 points $x_k = a + \frac{k(b-a)}{n}$, k =0 ,1, 2, …, n

- On calcule l'aire du k^{ème} rectangle sous la courbe qui est $\frac{(b-a)}{n} \times f\left(a + \frac{(k-1)\times(b-a)}{n}\right)$

- On calcule l'aire du k^{ème} rectangle au-dessus de la courbe qui est $\frac{(b-a)}{n} \times f\left(a + \frac{k\times(b-a)}{n}\right)$.

- On somme l'aire des rectangles dans les deux cas.

Ces deux valeurs donnent un encadrement de l'aire sous la courbe.

Pour tout n ≥ 1, on a $\sum_{k=1}^{n} \frac{(b-a)}{n} \times f(a + \frac{(k-1)\times(b-a)}{n}) \leq \int_a^b f(x)dx \leq \sum_{k=1}^{n} \frac{(b-a)}{n} \times f(a + \frac{k\times(b-a)}{n})$.

Si f est de classe C^1 sur [a, b], alors ces deux sommes tendent vers $\int_a^b f(x)dx$ lorsque n tend vers +∞.

Remarque :

On peut appliquer la même méthode pour des fonctions C^1 changeant de variations ou de signes mais alors on n'a plus l'encadrement précédent. On a cependant

$\lim_{n\to+\infty} \sum_{k=1}^{n} \frac{(b-a)}{n} \times f(a + \frac{(k-1)\times(b-a)}{n}) = \lim_{n\to+\infty} \sum_{k=1}^{n} \frac{(b-a)}{n} \times f(a + \frac{k\times(b-a)}{n}) = \int_a^b f(x)dx.$

1) Ecrire un script qui permet de donner un encadrement de « l'aire sous la courbe » pour a ≤ x ≤ b ,en utilisant la méthode des rectangles dans les cas suivants.

a) f(x) = sin(x) a= 0 et b = $\frac{\pi}{2}$. b) f(x) = ln (x) a = 1 et b = e.

c) f(x) = exp($-x^2$) a = 0 et b = 1 d) f(x) = $\sqrt{1-x^2}$ a = -1 et b = 0

2) On peut montrer que si f est de classe C^1 alors l'erreur commise en valeur absolue est $\frac{(b-a)^2}{2n} \times |\sup(f')|$.

Compléter le script en indiquant l'erreur commise.

3) Aspect math :

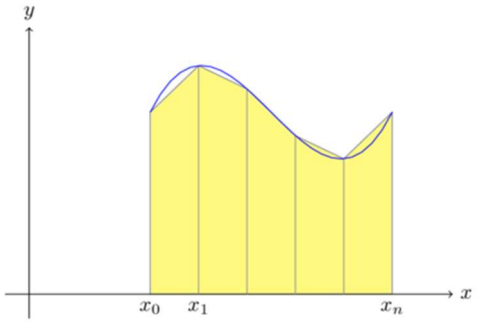
a) Calculer les valeurs exactes dans les cas a) et b).

b) Dans le cas d) , comment peut-on, sans calcul, (juste avec un argument géométrique) donner la valeur de l'aire ?

2) Méthode des trapèzes :

On considère une fonction de classe C^2 et positive sur un intervalle [a ; b].

On calcule une valeur approchée de « l'aire sous la courbe » (pour x dans [a ; b]) en sommant les aires des trapèzes comme sur la figure ci-après.



Pour cela :

- On découpe l'intervalle [a, b] en n intervalles (n ≥ 1), pour former n trapèzes.

On construit donc n +1 points $x_k = a + \frac{k(b-a)}{n}$, k =0 ,1, 2, …, n

- On calcule l'aire du $k^{ème}$ trapèze proche de la courbe qui est $\frac{(b-a)\times\left(f\left(a+\frac{(k-1)\times(b-a)}{n}\right)+f\left(a+\frac{k\times(b-a)}{n}\right)\right)}{2n}$

- On somme l'aire des trapèzes.

Cette valeur donne une valeur approchée de l'aire sous la courbe.

Remarque :

Si f est de classe C^2 sur [a ; b] alors la somme $\sum_{k=1}^{n} \frac{(b-a)\times\left(f\left(a+\frac{(k-1)\times(b-a)}{n}\right)+f\left(a+\frac{k\times(b-a)}{n}\right)\right)}{2n}$ tend vers $\int_a^b f(x)dx$.

4) Ecrire un script qui permet de calculer l'aire sous la courbe pour les fonctions de la question 1) avec la méthode des trapèzes.

Remarque : On peut montrer que si f est de classe C^2 alors l'erreur commise en valeur absolue est $\frac{(b-a)^3}{12n^2} \times |\sup(f")|$.

Exercice 13 :
Que fait ce script ?

```
def nb_chiffres(n):
    b=n
    m=0
    while b>=1:
        m=m+1
        b=n//10**m
    return(m)
n=eval(input("Entrer un entier naturel n "))
print(nb_chiffres(n))
```

On se donne un entier naturel non nul n.
Ecrire un script qui donne le nombre de chiffres qui composent n en utilisant une fonction récursive.
On pourra utiliser la division euclidienne de n par 10 (le reste donne le dernier chiffre…) .

Exercice 14 :
Ecrire des scripts, utilisant des fonctions récursives, qui ressortent U_n pour un n choisi dans les cas suivants :
a) $U_n = \frac{U_{n-1}+1}{U_{n-1}+2}$ et $U_0 = 1$ b) $U_{n+1} = \sqrt{U_n + 1}$ et $U_0 = 1$ c) $U_{n+1} = U_n^2 - 2$ et $U_0 = 1$.

Exercice 15 **: Babylone**
On se donne un réel a strictement positif et différent de 1. On considère la suite définie par récurrence ci-après:
$U_{n+1} = \frac{1}{2}\left(U_n + \frac{a}{U_n}\right)$ et $U_0 = 1$
Ecrire un script utilisant une fonction récursive (contenant les paramètres a et n) et qui retourne U_n.
Aspect math : On admet que la suite U converge. Quelle est sa limite ?

Exercice 16 :
Roger et Monique sont sur un bateau avec un ordi sur lequel python est installé et décident de faire une fonction F() qui renvoie au hasard l'un de leur nom, parce que c'est cool de voir son nom apparaitre une fois sur deux quand on s'ennuie.
1) Ecrire la fonction F() (on pourra utiliser la librairie random avec from random import * puis la commande randint(a,b)).
2) Ecrire une fonction G(n) qui répète n fois la fonction F et renvoie le nom le plus souvent sorti lors des n tirages.

Exercice 17 :
a) Affecter à la variable phrase la phrase : « Aujourd hui en info je fais un peu de python pour être au point le jour du concours et je m amuse beaucoup».
b) Combien de caractères contient cette phrase ? (ne pas compter bêtement, soyez malins …)
Quelle commande permet de trouver rapidement cette information ?
c) En tapant cette phrase, on remarque qu'on a tapé pleins de 'e'. Combien y en a-t-il ?
d) Monique qui n'aime pas les 'e' se posent alors la question sur le nombre de 'a' de la phrase phrase. Comment retrouver rapidement la réponse à cette question ?
e) Roger décide alors d'écrire une fonction NOMBRE(phrase, lettre) qui détermine le nombre de fois que lettre apparait dans phrase pour gagner du temps. Ecrire la fonction pour Roger qui est allé faire une sieste.
f) Tester la fonction dans l'interpréteur pour trouver le nombre de 'm' dans la phrase phrase.