3.4 Correction de quelques exercices

Vous pouvez remarquer que, souvent, nous reprenons des algorithmes des chapitres précédents que nous retraitons sous forme de fonctions.

Exercice 1 :

Ecrire un programme qui affiche tous les nombres impairs entre 0 et n, dans l'ordre croissant, le nombre n étant donné par l'opérateur

La résolution de cette question ne pose pas de difficultés :

```
#-*-coding:Latin-1 -*


#algo Fonction nombres pairs

def Pair(n):
    for i in range(0,n+1,2):
        print(i)
```

Et nous obtenons :

```
>>> Pair(12)
0
2
4
6
8
10
12
>>> Pair(15)
0
2
4
6
8
10
12
14
>>>
```

Question : Comment afficher les nombres impairs ?

Exercice 2 :

Ecrire la fonction qui prend en entrée la température en degrés Celsius et la convertit en degré Farheneit.
La formule de conversion est donnée par : $C° = (F° - 32) \times \frac{5}{9}$

Cet exercice ne pose pas de difficultés.
Tout d'abord, en transformant un peu la formule $C° = (F° - 32) \times \frac{5}{9}$, nous avons $F° = \frac{9}{5}C° + 32$, d'où les fonctions :

```
 #Transformation de degré
    #Celsius en degré Farenheit
def degre(d):
    return 9/5*d+32


degre(25)
77.0
degre(180)
356.0
degre(100)
212.0
degre(0)
32.0
```

```
 #Transformation de degré
    #Farenheit en degré Celsius
def conversion(t):
    return (t-32)*5/9

conversion(-56)
-48.888888888888886
conversion(56)
13.333333333333334
```

Et je ne résiste pas :

```
conversion(degre(25))
25.0
degre(conversion(25))
25.0
```

Exercice 3 :

Ecrire une fonction qui nous donne $n!$ (sans utiliser la récursivité)

Là, non plus, ce n'est pas extraordinaire.....

```
def factorielle(n):
    fact = 1
    for i in range(2, n+1):
        fact = fact * i
    return fact

factorielle(5)
120
factorielle(12)
479001600
factorielle(30)
265252859812191058636308480000000
factorielle(50)
30414093201713378043612608166064768844377641568960512000000000000
```

Exercice 4 :

Calculez et affichez à l'écran 2^k avec k variant de 0 à n inclus.

```
#-*-coding:Latin-1 -*
#algo Puissance_de_2


def puissance_Deux(k):
    return 2**k

n=int(input("Jusque quelle valeur souhaitez vous la puissance de 2?"))
for i in range(0,n+1):
    print(" Pour n=",i, "la puissance de 2 est ", puissance_Deux(i))
```

Le résultat est donné par :

```
Jusque quelle valeur souhaitez vous la puissance de 2?25
 Pour n= 0 la puissance de 2 est  1
 Pour n= 1 la puissance de 2 est  2
 Pour n= 2 la puissance de 2 est  4
 Pour n= 3 la puissance de 2 est  8
 Pour n= 4 la puissance de 2 est  16
 Pour n= 5 la puissance de 2 est  32
 Pour n= 6 la puissance de 2 est  64
 Pour n= 7 la puissance de 2 est  128
 Pour n= 8 la puissance de 2 est  256
 Pour n= 9 la puissance de 2 est  512
 Pour n= 10 la puissance de 2 est  1024
 Pour n= 11 la puissance de 2 est  2048
 Pour n= 12 la puissance de 2 est  4096
 Pour n= 13 la puissance de 2 est  8192
 Pour n= 14 la puissance de 2 est  16384
 Pour n= 15 la puissance de 2 est  32768
 Pour n= 16 la puissance de 2 est  65536
 Pour n= 17 la puissance de 2 est  131072
 Pour n= 18 la puissance de 2 est  262144
 Pour n= 19 la puissance de 2 est  524288
 Pour n= 20 la puissance de 2 est  1048576
 Pour n= 21 la puissance de 2 est  2097152
 Pour n= 22 la puissance de 2 est  4194304
 Pour n= 23 la puissance de 2 est  8388608
 Pour n= 24 la puissance de 2 est  16777216
 Pour n= 25 la puissance de 2 est  33554432
```

Exercice 5 :

Créez une fonction qui calcule la distance euclidienne dans $\mathbb{R}^3$ entre 2 points $A\left(x_A, y_A, z_A\right)$ et $B\left(x_B, y_B, z_B\right)$

Cette distance $d\left(A, B\right)$ est donnée par : $d\left(A, B\right) = \sqrt{\left(x_A - x_B\right)^2 + \left(y_A - y_B\right)^2 + \left(z_A - z_C\right)^2}$ D'où

```
#-*-coding:Latin-1 -*
#algo Distance Euclidienne

import math

def distance(xa,ya,za,xb,yb,zb):
    return math.sqrt((xa-xb)**2+(ya-yb)**2+(za-zb)**2)

xa=float(input("donner l'abscisse de A "))
ya=float(input("donner l'ordonnée de A "))
za=float(input("donner la cote de A "))
xb=float(input("donner l'abscisse de B "))
yb=float(input("donner l'ordonnée de B "))
zb=float(input("donner la cote de B "))
print("La distance entre les points A et B est ", distance(xa,ya,za,xb,yb,zb))
```

Exercice 6 :

Écrire une fonction suite qui, étant donnés deux arguments n et a renvoie la valeur de u_n, où la suite $(u_k)_{k\in\mathbb{N}}$ est la suite définie par :

$$\begin{cases} u_0 = 1 \\ u_{n+1} = \dfrac{1}{2}\left(u_n + \dfrac{a}{u_n}\right) \end{cases}$$

Enonce qui ne pose pas de difficultés majeures... Toujours faire attention à l'indentation

```
#-*-coding:Latin-1 -*



#algo Fonction racine carrée
def suite(n,a) :
    u = 1
    for k in range(n+1) :
        u=(u+a/u)/2
    return u
```

Nous obtenons comme résultats :

```
>>> suite(38,5)
2.23606797749979
>>> suite(100,5)
2.23606797749979
>>> suite(100,9)
3.0
>>>
```

Exercice 7 :

Ecrire une fonction récursive $U(n)$ qui retourne U_n avec n un entier positif passé en paramètre :

$$\begin{cases} U_0 = 5 \\ U_n = \sqrt{1+U_{n-1}} \end{cases}$$

```
#-*-coding:Latin-1 -*
#algo calcul_de_suite

import math
def U(n):
  if (n==0):
    return 1
  else:
    r=math.sqrt(1+U(n-1))
    return r
```

Il est possible de ré-écrire les lignes :

```
r=math.sqrt(1+U(n-1))
return r
```

En

```
return math.sqrt(1+U(n-1))
```

Exercice 8 :

En supposant que la fonction puissance ne soit pas native, coder un programme, utilisant une fonction récursive pour calculer la fonction puissance entière.

```
#-*-coding:Latin-1 -*
#algo calcul_de_puissance

def puissance( x, n):
  x=float(x)
  if (n==0):
    r=1
    return r
```

```
  else:
    r=x*puissance(x,n-1)
    return r
```

Quelques exemples de résultats :

```
>>> puissance(2,3)
8.0
>>> puissance(32,6)
1073741824.0
>>> puissance(2,10)
1024.0
>>> puissance(1.4242,2)
2.02834564
>>> puissance(1.4142,2)
1.9999616399999998
>>> puissance (100,0)
1
>>>
```

Exercice 9 :

Ecrire une fonction récursive Binomial(n,p) permettant de calculer le coefficient binomial C_n^p, où n et p sont des entiers naturels passés en paramètres. On rappelle que :

$$C_n^p = \frac{n!}{p!\,(n-p)!} \text{ et que } C_n^p = C_{n-1}^p + C_{n-1}^{p-1}$$

```
#-*-coding:Latin-1 -*
#algo calcul_coefficients_Binomiaux


def Binomial(n,p):
  if (n==0) or (p==n):
    return 1
  elif 0<p<n:
    return Binomial(n-1,p-1)+ Binomial(n-1,p)
  else:
    return 0
```

Nous obtenons comme résultat :

```
>>> Binomial(0,0)
1
>>> Binomial(100,100)
1
>>> Binomial(12,5)
330
>>>
```

Le calcul de

```
>>> Binomial(230,26)
```

prend un temps faramineux....à éviter!!*(je n'ai pas hésité à tuer le programme en fermant la fenêtre)*

Exercice 10 :

Ecrire une fonction récursive sommation(n) qui calcule la somme des inverses des carrés des n premiers entiers naturels non nuls, n est passé en paramètre : $S = \sum_{k=1}^{n} \frac{1}{k^2}$

```
#-*-coding:Latin-1 -*
#algo calcul_Sommation_Recursive


def Sommation(n):
  if (n==0):
    print("Division par zéro impossible")
  elif n==1:
    return 1
  else:
    return 1/(n**2)+Sommation(n-1)
```