
Algorithmique et Programmation 1 – TD - TP 6

FONCTIONS

Exercice 1 - Triangle de Pascal (retour sur le TP5)

Soit le programme suivant vu en cours, permettant d'afficher le binôme de Newton pour un entier entré par l'utilisateur :

```
1 def factorielle(n):
2     """Int --> Int
3     Fonction retournant la factorielle de n"""
4     assert type(n) is int, "Factorielle d'un entier"
5     fact = 1
6     for i in range(1, n+1) :
7         fact = fact * i
8     return(fact)

10 def coefficient_binomial(n, k):
11     """Int x Int --> Int, avec k <= n
12     Fonction retournant le coefficient binomial de k et n"""
13     assert type(n) is int, "n est un entier"
14     assert type(k) is int, "k est un entier"
15     #Pour k in [0, n], le coefficient binomial est un entier
16     coeff = factorielle(n)/(factorielle(k) * factorielle(n-k))
17     return(coeff)

19 def newton(n):
20     """Int --> None, n != 0
21     (a+b)^n = Somme(k=0 à n) (n!//k!(n-k)!a^{n-k}b^k)."""
22     assert type(n) is int, "n est un entier"
23     assert not(n == 0), "n différent de 0"
24     #Initialisation de la chaîne
25     chaine = "(a + b)^" + str(n) + " = "
26     for k in range(n+1) :
27         c = coefficient_binomial(n, k)
28         p = n - k
29         if not(c == 1) : #on écrit pas le facteur 1
30             chaine = chaine + str(c)
31         if not(p == 0) : #si un facteur = 0, on écrit pas les autres facteurs
32             chaine = chaine + "a"
33             if not(p == 1) :
34                 chaine = chaine + "^" + str(p)
35         if not(k == 0):
36             chaine = chaine + "b"
37             if not(k == 1) :
38                 chaine = chaine + "^" + str(k)
39         if k < n: #On continue avec + sauf pour la dernière occurrence
40             chaine = chaine + " + "
41     print(chaine)

43 n = int(input("Calculer le binôme de newton pour n = "))
44 newton(n)
```

1. Récupérer le fichier `newton.py` sur Moodle, et le tester.

Le **triangle de Pascal** est une présentation des coefficients binomiaux dans un triangle. La construction du triangle est régie par la relation de Pascal, pour tout entiers n et k tels que $0 < k < n$:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Par exemple, pour $n = 8$:

Afficher le triangle de Pascal jusqu'à $n = 8$

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
[1, 6, 15, 20, 15, 6, 1]
[1, 7, 21, 35, 35, 21, 7, 1]
[1, 8, 28, 56, 70, 56, 28, 8, 1]
```

2. Construire, dans une liste de listes, le triangle de Pascal jusqu'à un entier n entré par l'utilisateur en utilisant les fonctions écrites pour calculer le binôme de Newton : dans le triangle de Pascal, une même ligne contient tous les coefficients intervenant dans le développement d'une puissance de la somme de deux termes. Affichez ensuite ce triangle.
3. Construire, dans une liste de listes, le triangle de Pascal jusqu'à un entier n entré par l'utilisateur en utilisant la formule de Pascal¹ :

On en déduit une méthode de construction du triangle de Pascal, qui consiste, sous forme pyramidale, à placer 1 au sommet de la pyramide, puis 1 et 1 en dessous, de part et d'autre. Les extrémités des lignes sont toujours des 1, et les autres nombres sont la somme des deux nombres directement au-dessus. Sous forme triangulaire, i étant l'indice de ligne et j l'indice de colonne :

- *placer dans la colonne 0 des 1 à chaque ligne, et des 1 à chaque entrée de la diagonale,*
- *en partant du haut et en descendant, compléter le triangle en ajoutant deux coefficients adjacents d'une ligne, pour produire le coefficient de la ligne inférieure, en dessous du coefficient de droite.*

Utilisez ensuite la procédure définie dans la question précédente pour afficher ce triangle.

1. Source : https://fr.wikipedia.org/wiki/Triangle_de_Pascal

Exercice 2 - Fonctions mystères

Soit les 7 fonctions mystères suivantes, que vous pouvez télécharger dans le fichier `mysteres.py` sous Moodle.

Ces fonctions prennent en paramètre soit un entier, soit une chaîne de caractères et renvoient, soit un entier, soit une chaîne de caractères. A l'aide de tests, déterminer leur signature et compléter la chaîne de documentation de chacune d'entre elle.

```
def mystere0(s) :
    if len(s) == 0 :
        return ""
    else :
        return mystere0(s[1:len(s)]) + s[0]

def mystere1(n) :
    a = n // 3600
    c = n % 3600
    b = c // 60
    c = c % 60
    return (str(a) + ":" + str(b) + ":" + str(c))

def mystere2(n) :
    u = 1
    for x in range(1, n, 1) :
        u = u * x
    return u

def mystere3(n) :
    m = 0
    while (m+1)**2 < n :
        m = m + 1
    return m

def mystere4(n):
    if (n==0):
        return 2
    else:
        return mystere5(n-1) * mystere5(n-1)

def mystere5(n):
    a = 1
    s = 0
    t = -1
    while (a <= n) :
        a = a + 1
        s = s + t + 2
        t = t + 2
    return s

def mystere6(n):
    b, m = 2, n
    r = 1
    while m > 0:
        if m % 2 == 1:
            r = r * b
        b = b * b
        m = m // 2
    return r
```

Exercice 3 - Une maison avec turtle

```
1 import turtle

3 # Code de configuration de la fenetre
4 picture_width = 800
5 picture_height = 600
6 turtle.setup (width=picture_width, height=picture_height)
7 # Fin de code de configuration de la fenetre

9 # Ici commence le code pour dessiner

11 turtle.right(90)
12 turtle.forward(100)
13 turtle.left(90)
14 turtle.forward(100)

17 # Code d'attente pour fermer la fenetre
18 turtle.hideturtle ()
19 turtle.exitonclick ()
```

1. Récupérer le fichier `dessine.py` sur Moodle, le lire attentivement et le tester.
2. Modifiez le code pour finir de dessiner un carré.
3. Modifiez le code pour ajouter un triangle au-dessus du carré afin d'obtenir le schéma d'une 'maison'.
4. Ajoutez les lignes suivantes à votre programme (avant les instructions gérant le code de fermeture de la fenêtre), observez le résultat :

```
    turtle.up()
    turtle.goto(150,150)
    turtle.down()
```

5. Utilisez ce que vous avez pu déduire des questions précédentes pour dessiner deux maisons l'une à côté de l'autre.

Exercice 4 - Des dessins à répétition

On s'intéresse dans cet exercice au programme se trouvant dans le fichier `creneaux.py` :

```
1  import turtle

3  # Code de configuration de la fenetre
4  picture_width  = 800
5  picture_height = 600
6  turtle.setup (width=picture_width, height=picture_height)
7  # Fin de code de configuration de la fenetre

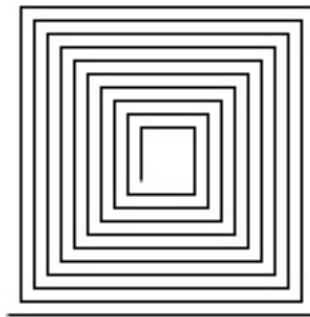
9  # Ici commence le code pour dessiner

11 # On deplace la tortue sur le bord droit de la fenetre
12 turtle.up()
13 turtle.goto(-400,0)
14 turtle.down()

16 for i in range(0,2,1) :
17     turtle.forward(40)
18     turtle.left(90)
19     turtle.forward(40)
20     turtle.right(90)
21     turtle.forward(40)
22     turtle.right(90)
23     turtle.forward(40)
24     turtle.left(90)

27 # Code d'attente pour fermer la fenetre
28 turtle.hideturtle ()
29 turtle.exitonclick ()
```

1. Le programme actuel dessine deux créneaux de largeur et hauteur égale. Pouvez-vous le modifier pour lui en faire dessiner 8, qui soient de plus en plus haut et de moins en moins larges au fur et à mesure qu'ils sont dessinés ?
2. A partir de ce que vous avez vu, créez un programme pour faire une spirale carré. Voilà un exemple de ce que vous pouvez obtenir :



Exercice 5 - Oursins

Ecrivez une fonction `dessine_oursin(nb_epines, taille_epines)` qui prend en argument un entier `taille_epines` et qui dessine un oursin à `nb_epines` épines dont les longueurs sont tirées au hasard dans l'intervalle $[0, \text{taille_epines}]$.

