# Formation DIU EIL 2021 Bloc 1

## Fonctions

### Exercice 1. Binôme de Newton

```
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 chaine
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 occurence
40             chaine = chaine + " + "
41     print(chaine)

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

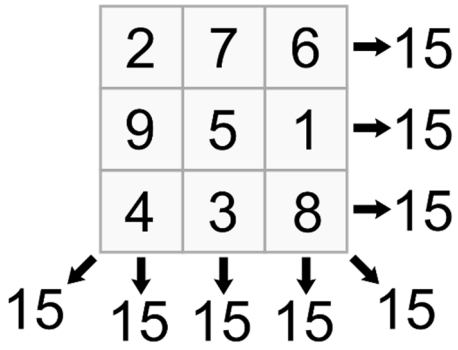
1. Expliquer les commentaires et les pseudo-instructions assert.
2. Modifier le code précédent afin de générer une erreur pour tout appel de Newton(n), avec n supérieur à 10.
3. Écrire une fonction qui génère le triangle de Pascal.
4. Proposer une fonction, basée sur le triangle de Pascal, qui calcule le binôme de Newton.

## Exercice 2. Carré magique

Un carré magique de dimension *n* est une matrice carrée d'ordre n contenant des entiers naturels, telle que :

- Tous les entiers de 1 à n × *n* sont présents dans la matrice,
- Les sommes des entiers de chaque ligne sont toutes égales à une même valeur *S*,
- La somme de chaque colonne est également *S*,
- Les sommes des deux diagonales de la matrice sont également *S*.

Par exemple, la matrice 3 × 3 suivante est un carré magique de dimension 3 :




1. Écrire une fonction valideCarreMagique(C)

## Exercice 3. Fonctions statistiques

1. Générer un tableau de 200 nombres aléatoires compris entre 0 et 2000.
2. Écrire une fonction qui donne le maximum d'une liste de nombres.
3. Écrire une fonction qui retourne la moyenne des nombres d'une liste de nombres.
4. Écrire une fonction qui retourne tous les nombres inférieurs à la moyenne d'une liste de nombres.
5. Écrire une fonction qui retourne les carrés d'une liste de nombres.
6. La variance d'une liste de nombres est égale à la différence entre la moyenne des carrés des éléments de la liste et le carré de la moyenne des éléments de la liste. Écrire une fonction qui renvoie la variance de la liste.
7. Écrire une fonction qui renvoie l'écart-type d'une liste de nombres.
8. Reprendre le calcul de la moyenne, la variance et l'écart type d'une liste de nombres en utilisant le module prédéfini numpy.