Cet exercice est basé sur l’automate qui gère les feux de signalisation d’un carrefour. Le comportement réel est loin de celui propose ici, mais le contexte permet de comprendre rapidement les questions. On ne tient pas compte des différentes temporisations du système réel par exemple.

Les variables feu\_vert, feu\_orange, feu\_rouge, pieton\_vert et pieton\_rouge sont de type bool.

|  |  |
| --- | --- |
| L’organigramme ci-contre représente de manière schématique le branchement conditionnel qui impose que le feu piéton soit au rouge si le feu est vert ou orange. Ecrire ce branchement conditionnel en Python en utilisant un if |  |
| Dans le même esprit, écrire la condition qui impose que le feu piéton soit rouge si le feu n’est pas rouge et que le feu piéton soit vert sinon (voir organigramme ci-contre). |  |

Écrire la boucle while qui speciffe que tant que le feu pieton est au vert, le feu tricolore doit rester au rouge.

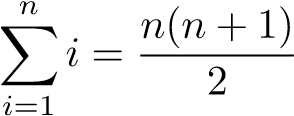
Écrire la fonction passage\_feu\_vert(fv, fo, fr) fonction admet en entrée trois variables booléennes (feu\_vert, feu\_orange nouvelle valeur de ces trois variables.

Sur le même principe, écrire les fonctions passage\_feu\_orange, passage\_pieton\_vert, passage\_pieton\_rouge.

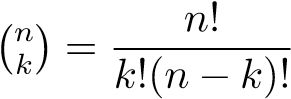
Proposer une fonction changement\_feux qui permette de passer si le feu était vert, etc. On exécutera cette fonction manuellement pour chaque changement d’état du feu.



Écrire une fonction somme\_n\_premiers\_entiers qui calcule la somme *Sn* des *n* premiers entiers naturels non nuls en utilisant une boucle for. Cette fonction doit admettre en entree l’entier *n* et renvoyer la somme *Sn*. Tester sur plusieurs valeurs de n pour valider le resultat :

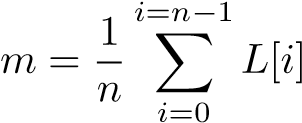


Écrire une fonction factorielle\_n qui calcule la valeur *n*! avec une boucle while. Cette fonction doit admettre en entrée l’entier *n* et renvoyer *n*!.

écrire une fonction qui fait appel à la précédente pour calculer le coefficient binomial . La fonction admettra en entrée *n* et *k*.



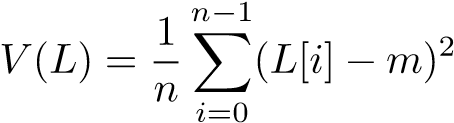
On considère une liste *L* de nombres ; dans les formules qui suivront on notera *n* la longueur de *L*. La moyenne d’une liste *L* de nombres est donnée par :



Écrire une fonction somme(L) qui prend en argument une liste *L* et qui retourne la somme de ces éléments (sans utiliser la fonction native sum(L)).

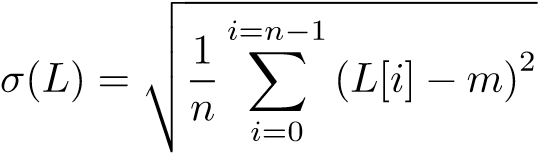
Écrire une fonction moyenne(L) qui prend en argument une liste *L* et qui retourne sa moyenne et qui utilise la fonction somme de nie précédemment.

La variance d’une liste *L* de nombres est donnée par :



Écrire une fonction variance(L) qui prend en argument une liste *L* et qui retourne sa variance.

L’Écart-type est défini comme étant la racine carrée de la variance :



Écrire une fonction ecart\_type(L) qui prend en argument une liste *L* et qui retourne l’écart-type des éléments de cette liste.

Tester les fonctions sur des listes que vous aurez créées.



Écrire une fonction est\_pair(x) qui renvoie un booléen True si x est un entier pair et False sinon. On admet que x est nécessairement un entier positif.

Écrire une fonction est\_premier(x) qui renvoie un booléen True si x est un nombre premier et False sinon. On admet que x est nécessairement un entier positif.

Écrire une fonction qui affiche la liste des 50 premiers nombres premiers.