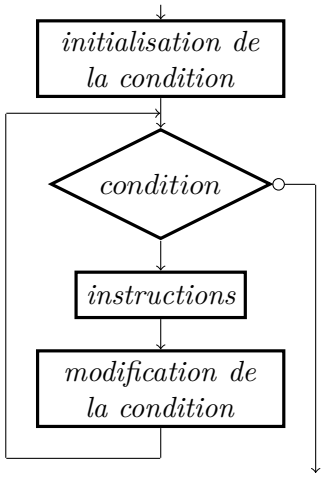
Dans ce TP, nous élargirons les boucles POUR aux boucles TANT QUE.

Les boucles `POUR` sont pratiques mais très limitées. Comme vous l'avez vu dans l'exercice de la mini-bataille navale de Vladimir, si l'utilisateur a coulé le croiseur, la boucle `POUR` oblige le programme boucler le nombre de fois initialement prévu. Afin de palier à ce problème, nous allons utiliser la boucle `TANT QUE`.

I Structure de boucle

On souhaite répéter une même (ou similaire) instruction jusqu'à ce qu'une condition ne soit plus vérifiée. Pour ce faire, on peut utiliser la structure de boucle TANT QUE.

initialisation de la condition
TANT-QUE (*condition*) **FAIRE**
    les instructions sont effectuées
    modification de la condition
FIN-TANT-QUE
Le reste du programme continue.



Sous python, la structure de boucle TANT-QUE se présente ainsi

```
initialisation de la condition
while (condition) :
    bloc d instructions
    modification de la condition
instructions suivantes  # instructions a l'exterieur de la boucle
```

Rq : pour ceux qui connaissent, les boucles `repeat` n'existent pas sous `python`.

Exo 1 : À l'aide d'une boucle `TANT-QUE`, et avant de dérouler le 3è algorithme pour $N = 3$, proposer un algorithme, son organigramme et son programme `python` qui affichent

1. dix fois la phrase "je suis le meilleur / la meilleure",
2. tous les entiers entre 0 et 10,
3. tous les carrés entre 0 et une valeur entière N donnée par l'utilisateur.

II Traduction d'une boucle POUR en TANT-QUE

Comme on vient de le voir, nous avons pu refaire l'exo 1 du TP3 à l'aide de boucles `TANT-QUE`.

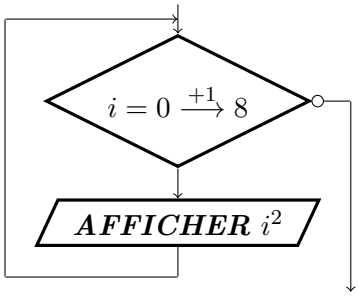
Propriété 1

Toute boucle POUR peut être traduite en boucle TANT-QUE.

⚠ la réciproque n'est pas vraie, à moins d'utiliser un artifice.

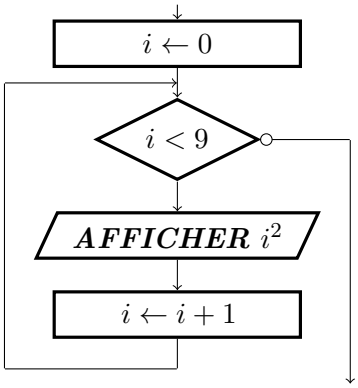
expl :

```
for i in range(9):
    print(i**2)
```



devient :

```
i=0
while i<9:
    print(i**2)
    i=i+1
```



Exo 2 :

1. Donner les organigrammes des programmes suivants et les traduire à l'aide de boucles `TANT-QUE`.

```
for i in range(13):
    print(i)
```

```
for i in range(-2,13):
    print(i)
```

```
for i in range(3,12,4):
    print(i)
```

2. Pour le 3ème algorithme, dérouler l'algorithme lorsqu'il s'agit d'une boucle `POUR` puis lorsqu'il s'agit d'une boucle `TANT-QUE`. Dorénavant, on déroulera par itération de boucle (comme au BAC).

Exo 3 : Proposer un programme `python` qui génère deux chiffres aléatoires et demande à l'utilisateur la valeur du produit tant que la réponse est fausse. Comment renvoyer le nombre d'essais avant réussite ?

Exo 4 : On souhaite jouer au jeu du "juste prix". Le maître du jeu (ou l'ordinateur) choisi un entier entre 0 et 100 et le joueur doit trouver cet entier avec le moins de coup possible. Selon sa réponse, le maître du jeu dit si le nombre recherché est plus petit, plus grand ou égal à sa proposition.

1. proposer un organigramme qui réponde au problème,
2. proposer une implémentation `python`.

Travail à faire pour la prochaine séance

Exo 5 :

À l'aide de la boucle `TANT-QUE` proposer un organigrame puis un programme python qui affiche la table de multiplication de 12 sous le format ci-contre :

$$\begin{aligned} 1 \times 12 &= 12 \\ 2 \times 12 &= 24 \\ &\vdots \\ 12 \times 12 &= 144 \end{aligned}$$

Rq : pour l'affichage, on pourra utiliser `print("{}×12={}".format(xx,xx))` où xx est à compléter

Exo 6 : Un établissement scolaire souhaite vérifier si la quantité de CO_2 de l'air ne dépasse pas la valeur limite de 5 000 ppm (partie par million). Proposer un programme `python` qui lit des valeurs ; si ces valeurs restent en deçà de la limite rien ne se passe, sinon le programme s'arrête et lance un message d'alerte.

Exo 7 : Dans une boîte de Pétri, un biologiste met en culture 100 bactéries. Il remarque que toutes les heures, elles augmentent de 40%. Il souhaite savoir au bout de combien d'heure il y en aura plus d'un million. On définit la suite (u_n) de sorte que u_n représente le nombre de bactéries à la n$^{\text{ième}}$ heure.

1. Montrer que u_n est une suite géométrique dont vous préciserez la relation de récurrence et le premier terme.
2. Compléter l'algorithme et l'organigramme suivants qui demandent à l'utilisateur le nombre maximal de bactéries voulues et prévient quand celui-ci sera atteint.
3. Au bout de combien d'heure y aura-t-il plus d'un million de bactéries ?

Variables
I, N : entiers
U : réel

Algorithme
$U \leftarrow \ldots$
$I \leftarrow \ldots$
SAISIR N
TANT-QUE ..... **FAIRE**
  $U \leftarrow \ldots$
  $I \leftarrow \ldots$
FIN-POUR
AFFICHER ...
Fin Algorithme