P4

LES BOUCLES

I Avant-propos : range

La fonction **range** permet de créer rapidement des listes de valeurs *entières* consécutives ou régulièrement espacées.

Dans ce qui suit, n et m sont des entiers relatifs.

- **range(n,m) :** liste des entiers de n à $m-1$.
- **range(n) :** liste des entiers de 0 à $n-1$.
- **range(n,m,pas)** : liste des entiers n, $n+pas$, $n+2pas$, etc., la dernière valeur excluant toujours m. Si le pas est négatif, la liste sera décroissante avec $n > m$.

Exemples

1. Définir la liste [3,4,5,6,7] avec range.

Entrée [1]:

2. Définir la liste [0,1,2,3,...,100] avec range.

Entrée [2]:

3. Définir la liste [4,6,8,10,12,14,16] avec range.

Entrée [3]:

4. Définir la liste [0,3,6,9,12] avec range.

Entrée [4]:

5. Définir la liste [5,4,3,2,1] avec range.

Entrée [5]:

Remarques :

- range(...) n'est pas du type list mais d'un type spécifique, appelé également range. On ne peut pas lui appliquer les opérations vues sur les listes (concaténation par exemple).
- range ne fonctionne qu'avec des paramètres entiers.
- Un range peut être vide, par exemple `range(1,1)` ou encore `range(5,4)`.

II Boucle itérative for

La **boucle for** permet de réaliser une itération sur les éléments d'une *séquence* (liste, range, chaîne de caractères, etc.) selon leur ordre d'apparition dans la séquence.

Pour tout élément d'une séquence
    Réaliser les instructions ...
Fin

```
for element in sequence:
    bloc d'instructions
#cette ligne ne fait pas partie du bloc
```

Exemples :

Entrée [6]:
```
for element in [3,7,1]:  # [3,7,1] est une liste
    print("L'élément est", element)
```

Out:
```
L'élément est 3
L'élément est 7
L'élément est 1
```

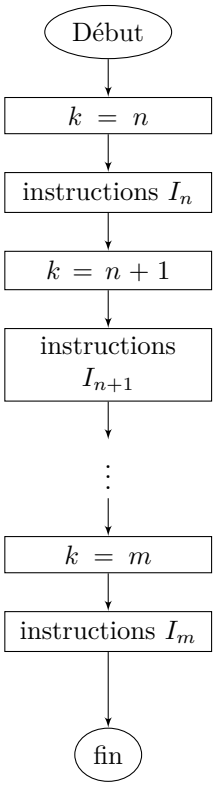
Entrée [7]:
```
for caractere in 'info': # 'info' est une chaîne de caractères
    print("L'élément est", caractere)
```

Out:
```
L'élément est i
L'élément est n
L'élément est f
L'élément est o
```

Le cas le plus classique est l'itération sur une liste d'entiers consécutifs. C'est ici que la fonction range intervient.

Pour k variant de n à m,
    Réaliser les instructions I_k
Fin

```
for k in range(n,m+1):
    instructions Ik
```



Exemple : Écrire le programme permettant le résultat donné.

Entrée [8]:
```

```

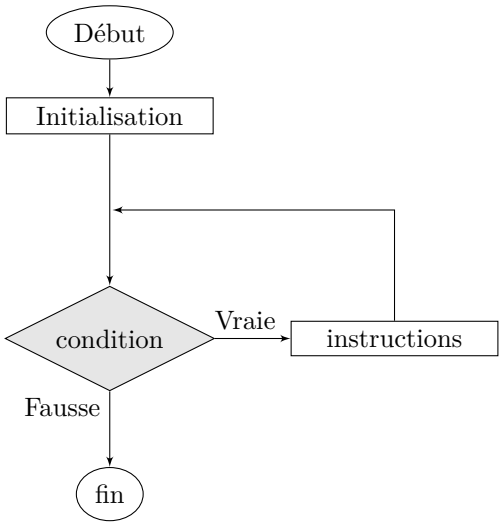
Out:
```
1
4
9
16
25
36
```

Remarque : si la sequence est vide, la boucle est ignorée sans erreur.

III Boucle conditionnelle while

La **boucle while** permet de réaliser un bloc d'instruction *tant qu'*une certaine condition est vraie.

Initialisation
Tant que la condition est vraie,
    Réaliser les instructions
Fin



La syntaxe python pour écrire ce type de boucle est la suivante :

```
initialisation
while condition:
    bloc_instructions
```

Remarques importantes :

- La *condition* est un booléen.
- Une boucle for est une boucle toujours finie (le nombre maximal de répétitions est égal au nombre d'éléments dans la séquence). Avec les boucles while, il y a un risque de créer une **boucle infinie**. Pour éviter les problèmes, on respectera les règles suivantes :

> La condition doit :
>
> **être initialisée :** on doit pouvoir dire dès le départ si elle est vraie ou fausse. Les variables de la condition doivent donc être définies avant le while.
>
> **évoluer à chaque étape :** au moins une des variables de la condition doit changer de valeur à chaque tour de boucle.
>
> **devenir fausse à un moment.**

Exemples à ne pas faire :

Entrée [9]:
```
while i > 0:
    print(i)
```

Out:
```
----> 1 while i > 0:
      2     print(i)

NameError: name 'i' is not
defined
```

Entrée [10]:
```
i = 5
while i > 0:
    print(i)
```

Out:
```
5
5
5
... # boucle infinie
```

Entrée [11]:
```
i = 5
while i > 0:
    print(i)
    i = i + 1
```

Out:
```
5
6
7
... # boucle infinie
```

Remarque : en T.P., vous pourrez repérer les boucles infinies en remarquant que l'exécution de votre cellule est très longue : tant qu'il est indiqué `Entrée [*]`, l'exécution n'est pas terminée. Dans ce cas, vous aurez la possibilité d'interrompre l'exécution avec le bouton "stop" ■.

Voici un exemple qui fonctionne :

Entrée [12]:
```
i = 5            # variable de la condition initialisée
while i > 0:
    print(i)
    i = i - 1    # la valeur de la variable évolue, dans le bon sens !
print("C'est fini !")
```

Out:
```
5
4
3
2
1
C'est fini !
```

Exemple : Déterminer le premier entier $n \in \mathbb{N}$ tel que $2^n \leqslant 10000$.

Entrée [13]:
```

```