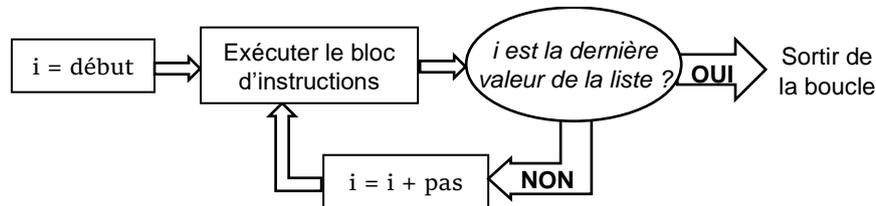


Il est souvent utile de confier aux machines des tâches répétitives, tels que des calculs, puisque ces dernières sont plus rapides (et moins facilement ennuyées) que les humains pour accomplir ces opérations.

Partie I : Découverte

A. Répétition prédéfinie d'instructions (boucle bornée)

Pour répéter des instructions, on peut utiliser une boucle POUR, qui permet de répéter un bloc d'instructions un certain nombre de fois déterminé par un paramètre entier (un compteur) prenant des valeurs entre début inclus et fin exclus, en évoluant de pas en pas :



Chaque boucle est appelée **itération**.

Les arguments début et pas sont optionnels dans les deux cas suivants :

- on peut écrire de début à fin et omettre pas ; par défaut, pas = 1
- on peut écrire à fin et omettre début et pas ; par défaut, début = 0 et pas = 1

Il est tout à fait possible d'imbriquer plusieurs boucles l'une dans l'autre.

I-A.1. Dans les algorithmes suivants, quelle variable est le compteur ? Qu'affiche l'algorithme ?

• **Algorithme A**

POUR i allant de 10 à 0 exclus par pas de -2
AFFICHER i^2
FIN POUR

• **Algorithme B**

POUR x allant de 2 à 10 exclus
AFFICHER x
FIN POUR

• **Algorithme C**

POUR s allant à 10 exclus
AFFICHER s
FIN POUR

I-A.2. Que font les algorithmes suivants ? Quel commentaire peut-on alors faire sur la boucle POUR ?

• **Algorithme D**

u = 0
POUR k allant de 1 à 11 exclus
u = 2*u+1
FIN POUR
AFFICHER u

• **Algorithme E**

u = 0
POUR k allant de 1 à 11 exclus
u = 2*u+1
AFFICHER u
FIN POUR

• **Algorithme F**

POUR k allant de 1 à 11 exclus
u = 0
u = 2*u+1
FIN POUR
AFFICHER u

I-A.3. On considère la suite définie pour tout $n \in \mathbb{N}$, par $S_n = \sum_{k=1}^n \frac{1}{k} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1} + \frac{1}{n}$.

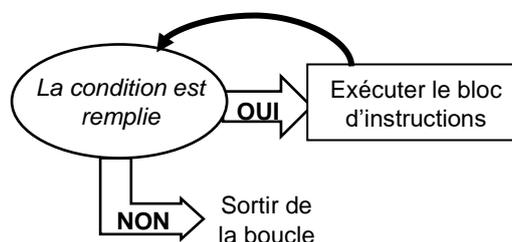
Ecrire un algorithme qui, étant donné un rang n , calcule et affiche la valeur de S_n .

Note : Le symbole $\sum_{k=a}^b$ signifie « la somme pour k allant de a à b inclus »

I-A.4. Ecrire un algorithme qui affiche 5 fois tous les entiers de 1 à 10 les uns après les autres.

B. Répétition d'instructions sous condition (boucle non-bornée)

Dans le cas où on cherche à répéter un bloc d'instruction jusqu'à ce qu'une certaine condition ne soit plus remplie, on utilise une boucle TANT QUE.



Cette boucle ne s'arrêtera QUE quand la condition ne sera plus remplie. Il faut donc s'assurer que cette condition soit non-réalisée à un moment ou un autre si on ne veut pas être bloqué dans une boucle infinie (dans laquelle le programme ne s'arrêterait jamais)...

I.B.1. Que fait l'algorithme suivant ?

```
somme = 0
i = 0
TANT QUE somme inférieure à 100
    Ajouter 1 à i
    Ajouter 12 à somme
FIN TANT QUE
AFFICHER somme, i
```

I.B.2. Quel est le problème dans chacun des algorithmes suivants ?

• **Algorithme A**

```
somme = 0
i = 0
TANT QUE somme supérieure à 100
    Ajouter 1 à i
    Ajouter 12 à somme
FIN TANT QUE
AFFICHER somme, i
```

• **Algorithme B**

```
somme = 0
i = 0
TANT QUE somme inférieure à 100
    Ajouter 1 à i
FIN TANT QUE
AFFICHER somme, i
```

I.B.3. Ecrire un algorithme qui, étant donné un entier naturel N , affiche la plus petite puissance de 2 supérieure à N .

Partie II : A vos ordinateurs !

La syntaxe Python (=CQFR !)

Répétitions prédéfinies POUR	Répétitions sous condition TANT QUE
<p>Commande for</p> <pre>for i in range(début,fin,pas): Bloc d'instructions</pre> <p>Rappel: les arguments <i>début</i> et <i>pas</i> sont optionnels</p>	<p>Commande while</p> <pre>while Condition: Bloc d'instructions</pre>

A. Pour débiter ensemble

II-A.1. Pour chacun des algorithmes suivants, écrire le code correspondant (les entrées sont à demander à l'utilisateur avec input) puis noter le résultat du test.

<p>• Algorithme A</p> <pre>POUR i allant de (10,0,-2) AFFICHER i² FIN POUR</pre>	<p>• Algorithme B</p> <pre>somme = 0 i = 0 TANT QUE somme inférieure à 100 Ajouter 1 à i Ajouter 12 à somme FIN TANT QUE AFFICHER somme, i</pre>
<p style="text-align: center;">Tests</p> <p>Les nombres affichés sont :</p> <p>.....</p> <p>.....</p>	<p style="text-align: center;">Tests</p> <p>Les nombres affichés sont :</p> <p>.....</p> <p>.....</p>

II-A.2. Écrire le code qui réalise la tâche demandée (dont l'algorithme est donné dans le sujet), sans oublier les commentaires, et le tester.

a. On considère la suite définie pour tout $n \in \mathbb{N}$, par $S_n = \sum_{k=1}^n \frac{1}{k} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1} + \frac{1}{n}$.

Étant donné un rang n , afficher la valeur de S_n .

b. Afficher 5 fois tous les entiers de 1 à 10 les uns après les autres.

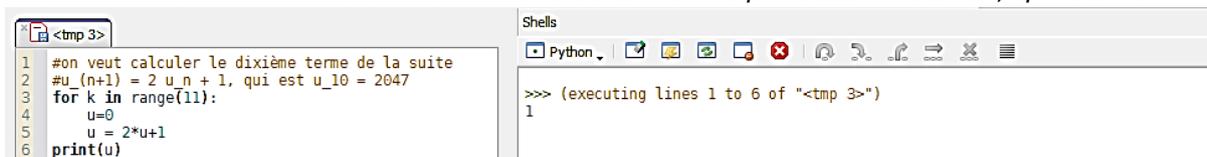
c. Étant donné un entier naturel N , afficher la plus petite puissance de 2 supérieure à N .

Vérifions que nous avons compris : Exercice flash !

Débogage

Lorsque le compilateur ne repère pas d'erreur mais que le code ne fait pas ce que vous voulez, il est nécessaire de revoir la logique du code, ligne par ligne. Pour cela, il est utile d'utiliser **des commandes d'affichage** print pour vérifier, au fur et à mesure, ce que fait le code et ainsi trouver où est le problème.

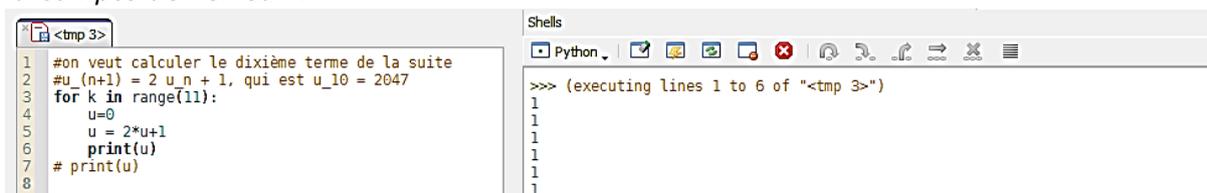
Exemple : On cherche à calculer le terme 10 de la suite définie par $u_{n+1} = 2u_n + 1$, qui est $u_{10} = 2047$.



```
1 #on veut calculer le dixième terme de la suite
2 #u_(n+1) = 2 u_n + 1, qui est u_10 = 2047
3 for k in range(11):
4     u=0
5     u = 2*u+1
6 print(u)
```

```
>>> (executing lines 1 to 6 of "<tmp 3>")
1
```

Le code nous donne 1... il y a un problème ! En affichant les valeurs de u au fur et à mesure de la boucle, on se rend compte de l'erreur :



```
1 #on veut calculer le dixième terme de la suite
2 #u_(n+1) = 2 u_n + 1, qui est u_10 = 2047
3 for k in range(11):
4     u=0
5     u = 2*u+1
6     print(u)
7 # print(u)
8
```

```
>>> (executing lines 1 to 6 of "<tmp 3>")
1
1
1
1
1
1
1
1
1
1
1
```

On a initialisé $u=0$ à l'intérieur de la boucle ! Donc u ne change jamais...

B. Exercices pour s'entraîner

Pour chacun des exercices suivants, écrire le code qui réalise la tâche demandée et le tester. Ne pas hésiter à écrire l'algorithme avant de coder, et penser aux commentaires !!!

II.B.1. En mathématique, *factorielle*(n) est le nombre entier définie par *factorielle*(1)=1 et, pour tout $n > 1$, *factorielle*(n) = $1 \times 2 \times \dots \times (n-1) \times n$.

Étant donné un entier naturel n demandé à l'utilisateur, vérifier que $n > 1$ puis afficher le résultat $1 \times 2 \times \dots \times (n-1) \times n$ si la condition est vérifiée. Proposer un jeu de tests en commentaire.

II.B.2. Étant donné un entier N demandé à l'utilisateur, vérifier que $N \leq 10$ puis afficher les unes à la suite des autres les 5 premiers termes des tables de multiplication de 1, puis de 2, puis de 3..., puis de N , en affichant quelle table est affiché à chaque fois.

Exemple : pour $N=2$ on affiche

Table de 1 :

1
2
3
4
5

Table de 2 :

2
4
6
8
10

II.B.3. Les suites de Syracuse sont définies de la manière suivante :

u_0 est un entier naturel non nul et, pour tout $n \in \mathbb{N}$, le terme suivant est :

$$u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair (division euclidienne)} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

Étant donné un entier u_0 demandé à l'utilisateur, afficher le rang n du premier terme égal à 1 de la suite de Syracuse associée.

Pour aller un peu plus loin...

Ecrire un code pour résoudre le problème suivant :

Edgar veut placer un capital initial de 1000 euros. On lui propose deux placements :

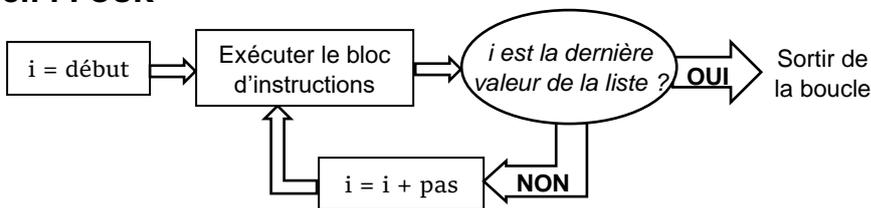
- Le premier propose d'augmenter le capital de 4% d'une année sur l'autre.
- Le second propose une augmentation annuelle de 5% du capital initial de 1000 euros.

A partir de combien d'années de placement est-il plus rentable de choisir le premier placement ?

Ce qu'il faut retenir

❖ **Répétition prédéfinie d'instruction : POUR**

POUR i allant de début à fin pas
 Bloc d'instruction
 FIN POUR



Chaque boucle est appelée *itération*.

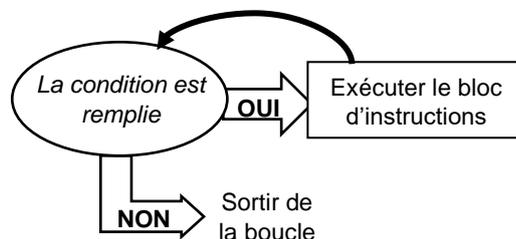
Le paramètre entier i prenant des valeurs entre début inclus et fin exclus, en évoluant de pas en pas :

Les arguments début et pas sont optionnels dans les deux cas suivants :

- on peut écrire de début à fin et omettre pas ; par défaut, pas = 1
- on peut écrire à fin et omettre début et pas ; par défaut, début = 0 et pas = 1

❖ **Répétition d'instruction sous condition : TANT QUE**

TANT QUE condition
 Bloc d'instruction
 FIN TANT QUE



Cette boucle ne s'arrêtera QUE quand la condition ne sera plus remplie

❖ **Répétitions d'instructions : Syntaxe python** (Voir sujet)