# Python - Les boucles

Cours de ECG-2

Lycée Gerville Réache

Chapitre 2

# Sommaire I

# I. Compétences attendues

- Savoir programmer en `Python` : fonctions d'entrée et de sortie, boucles `if`, `for`, `while`.

# II. Liste des commandes Python exigibles aux concours

- Comparaison et connecteurs logiques pour les booléens : `==`, `>`, `<`, `>=`, `<=`, `!=`, `True`, `False`, `and`, `or`, `not`

# II. Liste des commandes Python exigibles aux concours

- Comparaison et connecteurs logiques pour les booléens : `==`, `>`, `<`, `>=`, `<=`, `!=`, `True`, `False`, `and`, `or`, `not`
- Programmation : `range`, `for`, `while`,

# III. Les boucles `for`

**Définition 1 :**

La commande range énumère des entiers en progression arithmétique.

- `range(n)` énumère les entiers successifs de 0 à $n-1$.

**Définition 1 :**

La commande range énumère des entiers en progression arithmétique.

- `range(n)` énumère les entiers successifs de 0 à $n-1$.
- `range(n,m)` énumère les entiers successifs de $n$ à $m-1$ (avec $n < m$ ).

**Définition 1 :**

La commande range énumère des entiers en progression arithmétique.

- `range(n)` énumère les entiers successifs de 0 à $n-1$.
- `range(n,m)` énumère les entiers successifs de $n$ à $m-1$ (avec $n < m$ ).
- `range(n,m,r)` énumère les entiers en progression arithmétique de raison $r$ (entier positif ou négatif, appelé le pas), de $n$ inclu à $m$ exclu.

Remarques :

- La commande range ne provoque pas d'affichage.

Remarques :

- La commande range ne provoque pas d'affichage.
- On prendra garde à deux choses : l'énumération s'arrête avant l'entier pris comme extrémité à droite, et par défaut, elle commence à 0 et non à 1 .
  Par exemple :

Remarques :

- La commande range ne provoque pas d'affichage.
- On prendra garde à deux choses : l'énumération s'arrête avant l'entier pris comme extrémité à droite, et par défaut, elle commence à 0 et non à 1 .
  Par exemple :
  - `range(7)` énumère les entiers 0, 1, 2, 3, 4, 5, 6.

Remarques :

- La commande range ne provoque pas d'affichage.
- On prendra garde à deux choses : l'énumération s'arrête avant l'entier pris comme extrémité à droite, et par défaut, elle commence à 0 et non à 1 .
  Par exemple :
  - `range(7)` énumère les entiers 0, 1, 2, 3, 4, 5, 6.
  - `range(3,9)` énumère les entiers 3, 4, 5, 6, 7, 8.

Remarques :

- La commande range ne provoque pas d'affichage.
- On prendra garde à deux choses : l'énumération s'arrête avant l'entier pris comme extrémité à droite, et par défaut, elle commence à 0 et non à 1 .
  Par exemple :
  - `range(7)` énumère les entiers 0, 1, 2, 3, 4, 5, 6.
  - `range(3,9)` énumère les entiers 3, 4, 5, 6, 7, 8.
  - `range(-2,8,2)` énumère les entiers $-2$, 0, 2, 4, 6.

Remarques :

- La commande range ne provoque pas d'affichage.
- On prendra garde à deux choses : l'énumération s'arrête avant l'entier pris comme extrémité à droite, et par défaut, elle commence à 0 et non à 1 .
  Par exemple :
  - `range(7)` énumère les entiers 0, 1, 2, 3, 4, 5, 6.
  - `range(3,9)` énumère les entiers 3, 4, 5, 6, 7, 8.
  - `range(-2,8,2)` énumère les entiers −2, 0, 2, 4, 6.
  - `range(4,0,-1)` énumère les entiers 4, 3, 2, 1.

**Définition 2 :**

Pour répéter un bloc d'instructions un nombre déterminé de fois, on utilise une boucle **for**.
La syntaxe est la suivante :

Éditeur

```
1   for variable in range(début, fin,pas) :
2           instructions
```

La variable parcourt dans l'ordre les entiers de début inclus à fin exclu en suivant le pas régulier et, à chaque étape, les instructions sont effectuées.

Remarques :

- Comme pour les instructions conditionnelles, il faut faire attention aux deux points après le range et à l'indentation !

Remarques :

- Comme pour les instructions conditionnelles, il faut faire attention aux deux points après le range et à l'indentation !
- Pour le `range`, on utilisera lorsque c'est possible les expressions simplifiées `range(n,m)` ou `range(n)`.

**Exemple 1 :**

Taper dans l'éditeur de texte le script suivant puis l'exécuter. Que calcule-t-il ?

Éditeur

```
1  p = 1
2  for k in range(1,11) :
3      p = p*k
4  print(p)
```

Remarques :

- Comme pour les instructions conditionnelles, il faut faire attention aux deux points après le range et à l'indentation !
- Pour le `range`, on utilisera lorsque c'est possible les expressions simplifiées `range(n,m)` ou `range(n)`.

**Exemple 1 :**

Taper dans l'éditeur de texte le script suivant puis l'exécuter. Que calcule-t-il ?

Éditeur

```
1  p = 1
2  for k in range(1,11) :
3      p = p*k
4  print(p)
```

Console

```
3628800
```

Le script calcule 10!.

Remarques :

- Une spécificité du langage Python est qu'il est possible d'itérer sur de nombreux objets. On peut remplacer l'énumérateur range par une chaîne de caractères ou un tableau.
  - Par exemple, avec une chaîne de caractère :

Éditeur

```
1  for car in "ma chaine":
2    print(car)
```

On obtient l'affichage successif de chaque lettre de cette chaine de caractère.

Console

```
m a c h a i n e
```

Remarques :

- Une spécificité du langage Python est qu'il est possible d'itérer sur de nombreux objets. On peut remplacer l'énumérateur range par une chaîne de caractères ou un tableau.
  - Par exemple, avec une chaîne de caractère :

Éditeur
```
1  for car in "ma chaine":
2    print(car)
```

On obtient l'affichage successif de chaque lettre de cette chaine de caractère.

Console
```
m a c h a i n e
```

  - Si la variable T contient un tableau, la boucle

Éditeur
```
1  for variable in T :
2          instructions
```

répète les instructions indentées, une fois pour chaque élément du tableau.

### Exercice 1 – Suite récurrente simple :

Soit $(u_n)_{n\in\mathbb{N}}$ une suite définie par $u_0 = \dfrac{1}{2}$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = \dfrac{2u_n}{u_n + 1}$.

1. Définir une fonction `suite` qui, à un entier naturel $n$ en entrée, renvoie la valeur de $u_n$.

Éditeur

```
1  def ... :
2      u = ...
3      for k in ... :
4          ...
5      return ...
```

### Exercice 1 – Suite récurrente simple :

Soit $(u_n)_{n\in\mathbb{N}}$ une suite définie par $u_0 = \dfrac{1}{2}$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = \dfrac{2u_n}{u_n + 1}$.

1. Définir une fonction `suite` qui, à un entier naturel $n$ en entrée, renvoie la valeur de $u_n$.

Éditeur

```
1  def ... :
2      u = ...
3      for k in ... :
4          ...
5      return ...
```

2. De manière empirique, que diriez vous de la limite éventuelle de cette suite ?

**Méthode**

Si on connaît le rang, une boucle `for` est plus simple à mettre en œuvre qu'une boucle `while`.

### Exercice 1 – Suite récurrente simple :

1.

Éditeur

```
1  def suite(n) :
2      u = 1/2
3      for k in range(1,n+1) :
4          u = 2*u/(u+1)
5      return u
```

**Méthode**

Si on connaît le rang, une boucle `for` est plus simple à mettre en œuvre qu'une boucle `while`.

### Exercice 1 – Suite récurrente simple :

1. 

Éditeur

```
1  def suite(n) :
2      u = 1/2
3      for k in range(1,n+1) :
4          u = 2*u/(u+1)
5      return u
```

2. Il semblerait que $(u_n)_{n\in\mathbb{N}}$ converge vers 1.

### Exercice 2 – Produit simple :

On définit la suite $(u_n)_{n\in\mathbb{N}^*}$ par $\forall\, n \in \mathbb{N}^*,\ u_n = \prod_{k=1}^{n} \left(1 + \frac{1}{k}\right)$.

1. 1. Compléter le programme ci-dessous qui prend en argument $n$ et renvoie $u_n$. Que conjecturez-vous ?

Éditeur

```
1  n = int(input('Donner n : '))
2  P = ...
3  for i in ... :
4          ...
5  print(...)
```

### Exercice 2 – Produit simple :

On définit la suite $(u_n)_{n\in\mathbb{N}^*}$ par $\forall\, n \in \mathbb{N}^*,\ u_n = \prod_{k=1}^{n} \left(1 + \frac{1}{k}\right)$.

1. 1. Compléter le programme ci-dessous qui prend en argument $n$ et renvoie $u_n$. Que conjecturez-vous ?

Éditeur

```
1  n = int(input('Donner n : '))
2  P = ...
3  for i in ... :
4          ...
5  print(...)
```

2. Démontrer votre conjecture.

### Exercice 2 – Produit simple :

On définit la suite $(u_n)_{n\in\mathbb{N}^*}$ par $\forall\, n \in \mathbb{N}^*,\ u_n = \prod_{k=1}^{n} \left(1 + \frac{1}{k}\right)$.

1. 1. Compléter le programme ci-dessous qui prend en argument $n$ et renvoie $u_n$. Que conjecturez-vous ?

Éditeur

```
1  n = int(input('Donner n : '))
2  P = ...
3  for i in ... :
4          ...
5  print(...)
```

   2. Démontrer votre conjecture.

2. Mêmes questions avec la suite $(v_n)_{n\in\mathbb{N}^*}$ par $\forall\, n \in \mathbb{N}^*,\ v_n = \prod_{k=1}^{n} \left(1 - \frac{1}{k^2}\right)$.

**Attention**

Pensez à initialiser la variable P !

### Exercice 2 – Produit simple :

1. 1.

Éditeur

```
1  n = int(input('Donner n : '))
2  P = 1
3  for i in range(1,n+1) :
4          P=P*(1+1/i)
5
6  print(P)
```

Il semblerait que $\forall\, n \in \mathbb{N}^*,\ u_n = n+1$.

**Attention**

Pensez à initialiser la variable P !

### Exercice 2 – Produit simple :

1. 1.

```
Éditeur
1  n = int(input('Donner n : '))
2  P = 1
3  for i in range(1,n+1) :
4          P=P*(1+1/i)
5
6  print(P)
```

   Il semblerait que $\forall n \in \mathbb{N}^*$, $u_n = n+1$.

2. Après factorisation et télescopage, $\forall n \in \mathbb{N}^*$, on a :

$$u_n = \prod_{k=1}^{n} \left(1+\frac{1}{k}\right) = u_n = \prod_{k=1}^{n} \frac{k+1}{k} = \frac{n+1}{n} \times \frac{n}{n-1} \times \ldots \frac{3}{2} \times \frac{2}{1} = n+1.$$

### Exercice 2 – Produit simple :

2 1

Éditeur

```
1  n = int(input('Donner n : '))
2  P = 1
3  for i in range(1,n+1) :
4          P=P*(1-1/(i**2))
5
6  print(P)
```

Plus difficile à voir !

## Exercice 2 – Produit simple :

2. 1.

```
Éditeur
1  n = int(input('Donner n : '))
2  P = 1
3  for i in range(1,n+1) :
4          P=P*(1-1/(i**2))
5
6  print(P)
```

Plus difficile à voir !

2. Par télescopage toujours, $\forall n \in \mathbb{N}^*$, on a :

$$\prod_{k=2}^{n}\left(1-\frac{1}{k^2}\right)=\prod_{k=2}^{n}\left(\frac{(k-1)(k+1)}{k\times k}\right)=\prod_{k=2}^{n}\left(\frac{\frac{k+1}{k}}{\frac{k}{k-1}}\right)=\frac{\frac{n+1}{n}}{\frac{2}{1}}=\frac{n+1}{2n}.$$

### Exercice 3 – Suite récurrente double :

Soit $(F_n)_{n\in\mathbb{N}}$ une suite définie par $F_0 = 0$, $F_1 = 1$ et pour tout $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$.

1. Définir une fonction `Fib` qui, à un entier naturel $n \geqslant 2$ en entrée, renvoie la valeur de $F_n$.

Éditeur

```
1  def Fib... :
2      u = ...
3      v = ...
4      for k in ... :
5          ...
6      return w
```

## Exercice 3 – Suite récurrente double :

Soit $(F_n)_{n\in\mathbb{N}}$ une suite définie par $F_0 = 0$, $F_1 = 1$ et pour tout $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$.

1. Définir une fonction `Fib` qui, à un entier naturel $n \geqslant 2$ en entrée, renvoie la valeur de $F_n$.

Éditeur

```
def Fib... :
    u = ...
    v = ...
    for k in ... :
        ...
    return w
```

2. Modifier votre fonction afin qu'elle renvoie le quotient $\dfrac{F_{n+1}}{F_n}$ pour $n \geqslant 1$.

   Une conjecture sur sa limite éventuelle ?

**Attention**

Pensez à une variable auxiliaire pour conserver les termes de la suite !

### Exercice 3 – Suite récurrente double :

1.

Éditeur

```
1  def Fib(n):
2      u = 0
3      v = 1
4      for k in range(2,n+1):
5          w = u+v
6          u = v
7          v = w
8      return w
```

Une version plus élégante et plus « pythonique » qui fonctionne $\forall\, n \in \mathbb{N}$.

Éditeur

```
1  def Fib(n):
2      u,v = 0,1
3      for k in
       ↪  range(n):
4          u,v = v,u + v
5      return u
```

## Exercice 3 – Suite récurrente double :

2

Éditeur

```
1  def Fib_ratio(n):
2      u, v = 0, 1  # u_0, u_1
3
4      for k in range(n):
5          u,v = v,u + v    # u_n,u_{n+1}
6      return v / u
```

On peut montrer que $\lim\limits_{n\to+\infty} \dfrac{F_{n+1}}{F_n} = \varphi = \dfrac{1+\sqrt{5}}{2}$, le nombre d'or.

**Exercice 4 :**

**Méthode 1 :**

1. En vous aidant de l' exemple (1) , écrire une fonction `facto` qui prend un entier naturel $n$ en argument et renvoie $n!$.

**Exercice 4 :**

**Méthode 1 :**

1. En vous aidant de l' exemple (1) , écrire une fonction `facto` qui prend un entier naturel $n$ en argument et renvoie $n!$.
2. En utilisant la formule explicite des coefficients binomiaux, en déduire une fonction `binom1` qui prend en argument deux entiers $p$ et $n$ et renvoie $\binom{n}{p}$.

**Exercice 4 :**

**Méthode 1 :**

1. En vous aidant de l' exemple (1) , écrire une fonction `facto` qui prend un entier naturel $n$ en argument et renvoie $n!$.
2. En utilisant la formule explicite des coefficients binomiaux, en déduire une fonction `binom1` qui prend en argument deux entiers $p$ et $n$ et renvoie $\binom{n}{p}$.
3. Tester votre programme. Que pouvez vous conclure sur son efficacité ?

**Attention**

Pensez à initialiser la variable p !

### Exercice 4 :

#### Méthode 1 :

1.

Et une version récursive :

Éditeur

```
1  def facto(n):
2      p=1
3      for k in
       ↪  range(1,n+1):
4          p=p*k
5      return p
```

Éditeur

```
1  def facto(n):
2      if n == 0 or n
       ↪  == 1:
3          return 1
4      return n *
       ↪  facto(n -
       ↪  1)
```

## Exercice 4 :

### Méthode 1 :

2. On sait que pour $p, n \in \mathbb{N}^*$, $\binom{n}{p} = \begin{cases} \dfrac{n!}{p!(n-p)!} & \text{si } p \leqslant n \\ 0 & \text{sinon.} \end{cases}$

   On en déduit la fonction ci-dessous :

Éditeur

```
1  def binom1(n,p):
2    if p<n:
3      return
       ↪  facto(n)/(facto(p)*facto(n-p))
4    else :
5      return 0
```

## Exercice 4 :

### Méthode 1 :

2. On sait que pour $p, n \in \mathbb{N}^*$, $\binom{n}{p} = \begin{cases} \dfrac{n!}{p!(n-p)!} & \text{si } p \leqslant n \\ 0 & \text{sinon.} \end{cases}$

   On en déduit la fonction ci-dessous :

Éditeur

```
1  def binom1(n,p):
2    if p<n:
3      return
       ↪  facto(n)/(facto(p)*facto(n-p))
4    else :
5      return 0
```

3. Le programme est (trop) gourmand en calcul à travers la fonction `facto`.

**Exercice 4 :**

**Méthode 2 :**

1. Justifier que pour $p, n \in \mathbb{N}^*$ avec $p \leqslant n$ :
$$\binom{n}{p} = \prod_{i=1}^{p} \frac{n-p+i}{i}.$$

## Exercice 4 :

### Méthode 2 :

1. Justifier que pour $p, n \in \mathbb{N}^*$ avec $p \leqslant n$ :
$$\binom{n}{p} = \prod_{i=1}^{p} \frac{n-p+i}{i}.$$
2. Compléter le programme suivant permettant le calcul de $\binom{n}{p}$.

Éditeur

```
1  def binom1(n,p):
2    if p<n :
3      coef= ...
4      for i in range( ... ):
5        coef= ...
6      return coef
7    else :
8      return ...
```

**Exercice 4 :**

**Méthode 2 :**

1. Il suffit de simplifier numérateur et dénominateur :
Pour $p, n \in \mathbb{N}^*$ avec $p \leqslant n$, on a :

$$\begin{aligned}\binom{n}{p} &= \frac{n!}{p!(n-p)!} = \frac{(n-p)! \times (n-p+1) \times \ldots \times n}{p!(n-p)!} \\ &= \frac{(n-p+1) \times (n-p+2) \times \ldots \times (n-p+p)}{p!} \\ &= \frac{\prod_{i=1}^{p} n-p+i}{\prod_{i=1}^{p} i} \\ &= \prod_{i=1}^{p} \frac{n-p+i}{i}.\end{aligned}$$

Idée

Petite amélioration qui ne coûte rien :

$$\frac{n-p+i}{i} = \frac{n-p}{i} + 1.$$

### Exercice 4 :

**Méthode 2 :** ❷

Éditeur

```
1  def binom1(n,p):
2    if p<n :
3      coef= 1
4      for i in range(1,p+1):
5        coef= coef*((n-p)/i+1)
6      return coef
7    else :
8      return 0
```

Remarque : On pourrait aussi démontrer ce résultat par récurrence à partir de la formule, dite du capitaine :

$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1} = \frac{n}{k}\frac{n-1}{k-1}\binom{n-2}{k-2} = \cdots = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 1}.$$

Formule du capitaine qui donne l'envie d'une fonction récursive :

## Exercice 5 – Boucles imbriquées :

1. Compléter le programme Python ci-dessous qui calcule le nombre de triplets $(a, b, c) \in \mathbb{N}^3$ solutions de

$$a^2 + b^2 = c^2 \quad \text{et} \quad 1 \leqslant a \leqslant b \leqslant c \leqslant 50.$$

Éditeur

```
1  Compteur = 0
2
3  for c in ... :
4      for b in ... :
5          for a in ... :
6              if ... :
7                  Compteur+=1
8
9  print(Compteur)
```

### Exercice 5 – Boucles imbriquées :

1. Compléter le programme Python ci-dessous qui calcule le nombre de triplets $(a, b, c) \in \mathbb{N}^3$ solutions de

$$a^2 + b^2 = c^2 \quad \text{et} \quad 1 \leqslant a \leqslant b \leqslant c \leqslant 50.$$

Éditeur

```
1  Compteur = 0
2
3  for c in ... :
4      for b in ... :
5          for a in ... :
6              if ... :
7                  Compteur+=1
8
9  print(Compteur)
```

2. Même question en modifiant l'exposant 2 par 3, 4, 5 .... Que remarque-t-on ?

## Exercice 5 – Boucles imbriquées :

1.

Éditeur

```
1  Compteur = 0
2
3  for c in range(1,51) :
4      for b in range(1,c+1) :
5          for a in range(1,b+1) :
6              if a**2+b**2==c**2 :
7                  Compteur+=1
8
9  print(Compteur)
```

## Exercice 5 – Boucles imbriquées :

1. Éditeur

```
1  Compteur = 0
2
3  for c in range(1,51) :
4      for b in range(1,c+1) :
5          for a in range(1,b+1) :
6              if a**2+b**2==c**2 :
7                  Compteur+=1
8
9  print(Compteur)
```

2. Pour tout exposant supérieur ou égal à 3, il n'existe aucun triplet d'entier $(a, b, c)$ tel que $a^n + b^n = c^n$. C'est le grand théorème de Fermat ou théorème de Fermat-Wiles démontré par ce dernier en 1994.

**Exercice 6 – Sommes doubles :**

On considère les sommes doubles suivantes :

$$\mathrm{S}_n = \sum_{1\leqslant i,j\leqslant n} \frac{i}{2^j} \quad \text{et} \quad \mathrm{T}_n = \sum_{1\leqslant i\leqslant j\leqslant n} \frac{i^3}{j(j+1)}.$$

**Exercice 6 – Sommes doubles :**

On considère les sommes doubles suivantes :

$$\mathrm{S}_n = \sum_{1\leqslant i,j\leqslant n} \frac{i}{2^j} \quad \text{et} \quad \mathrm{T}_n = \sum_{1\leqslant i\leqslant j\leqslant n} \frac{i^3}{j(j+1)}.$$

1. 1. Écrire un programme qui, étant donné un entier $n \geqslant 1$, calcule et affiche $\mathrm{S}_n$.

**Exercice 6 – Sommes doubles :**

On considère les sommes doubles suivantes :

$$\mathrm{S}_n = \sum_{1\leqslant i,j\leqslant n} \frac{i}{2^j} \quad \text{et} \quad \mathrm{T}_n = \sum_{1\leqslant i\leqslant j\leqslant n} \frac{i^3}{j(j+1)}.$$

1. 1. Écrire un programme qui, étant donné un entier $n \geqslant 1$, calcule et affiche $\mathrm{S}_n$.
   2. Même question pour $\mathrm{T}_n$.

**Exercice 6 – Sommes doubles :**

On considère les sommes doubles suivantes :

$$\mathrm{S}_n = \sum_{1\leqslant i,j\leqslant n} \frac{i}{2^j} \quad \text{et} \quad \mathrm{T}_n = \sum_{1\leqslant i\leqslant j\leqslant n} \frac{i^3}{j(j+1)}.$$

1. 1. Écrire un programme qui, étant donné un entier $n \geqslant 1$, calcule et affiche $\mathrm{S}_n$.
   2. Même question pour $\mathrm{T}_n$.
2. Calculer à la main $\mathrm{S}_n$ et $\mathrm{T}_n$ en fonction de $n$.

### Exercice 6 – Sommes doubles :

1. 1.

Éditeur

```
1  n = int(input('Donner n : '))
2  S = 0
3  for i in range(1,n+1) :
4      for j in range(1,n+1) :
5          S = S+i/2**j
6  print(S)
```

## Exercice 6 – Sommes doubles :

1. 1.

Éditeur

```
1  n = int(input('Donner n : '))
2  S = 0
3  for i in range(1,n+1) :
4      for j in range(1,n+1) :
5          S = S+i/2**j
6  print(S)
```

2.

Éditeur

```
1  n = int(input('Donner n : '))
2  T = 0
3  for i in range(1,n+1) :
4      for j in range(i,n+1) :
5          T = T+i**3/(j*(j+1))
6  print(T)
```

**Exercice 6 – Sommes doubles :**

2. Pour $S_n$, on a :

$$S_n = \left(\sum_{i=1}^{n} i\right) \times \left(\sum_{j=1}^{n} \left(\frac{1}{2}\right)^j\right) = \frac{n(n+1)}{2} \times \left(\frac{1}{2}\,\frac{1-\left(\frac{1}{2}\right)^n}{1-\frac{1}{2}}\right) = \frac{n(n+1)}{2} \times \left(1-\left(\frac{1}{2}\right)^n\right)$$

Pour $T_n$, on a :

$$T_n = \sum_{j=1}^{n}\left(\sum_{i=1}^{j} \frac{i^3}{j(j+1)}\right) = \sum_{j=1}^{n} \frac{1}{j(j+1)}\left(\sum_{i=1}^{j} i^3\right) = \sum_{j=1}^{n} \frac{1}{j(j+1)} \times \frac{j^2(j+1)^2}{4}$$

$$= \sum_{j=1}^{n} \frac{j(j+1)}{4} = \frac{1}{4}\sum_{j=1}^{n} j^2 + \frac{1}{4}\sum_{j=1}^{n} j$$

$$= \frac{n(n+1)(2n+1)}{24} + \frac{n(n+1)}{8}.$$

**Exercice 7 – Suites imbriquées :**

On considère les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies par $a_0 = 1, b_0 = 2$ et les relations :

$$\forall\, n \in \mathbb{N},\ a_{n+1} = \frac{a_n^2}{a_n + b_n} \quad \text{et} \quad b_{n+1} = \frac{b_n^2}{a_n + b_n}.$$

Écrire une fonction `suites` ayant en entrée un entier naturel $n$ et qui renvoie les réels $a_n$, $b_n$.

**Attention**

Ne pas réutiliser les valeurs de $a$ ou $b$ sans réfléchir !

### Exercice 7 – Suites imbriquées :

Éditeur

```
1  def suites(n) :
2      a = 1
3      b = 2
4      for k in range(n) :
5          aaux = a
6          baux = b
7          a = aaux**2 / (aaux + baux)
8          b = baux**2 / (aaux + baux)
9      return a,b
```

# IV. Les boucles `while`

**Définition 3 :**

Pour répéter un bloc d'instructions tant qu'une condition est vérifiée, on utilise une boucle **`while`**.
La syntaxe est la suivante :

Éditeur

```
1   while condition :
2           instructions
```

Tant que la condition est vérifiée, les instructions sont effectuées. Dès que la condition n'est plus vérifiée, les instructions sont ignorées.

Remarques :

- Même remarque que pour les boucles **if** et **for** : attention aux deux points et à l'indentation !

**Exemple 2 :**

Taper dans l'éditeur de texte le script suivant puis l'exécuter. Que calcule-t-il ?

Éditeur

```
1  s = 0
2  i = 2
3  while i <=100 :
4      s = s + i
5      i = i+2
6  print(s)
```

Remarques :

- Même remarque que pour les boucles **if** et **for** : attention aux deux points et à l'indentation !
- Attention également à la condition : elle doit se révéler fausse à un moment donné, sinon la boucle est sans fin...

**Exemple 2 :**

Taper dans l'éditeur de texte le script suivant puis l'exécuter. Que calcule-t-il ?

Éditeur

```
1  s = 0
2  i = 2
3  while i <=100 :
4      s = s + i
5      i = i+2
6  print(s)
```

### Exercice 8 – Rang d'une suite dépassant un seuil :

Soit $(u_n)_{n\in\mathbb{N}}$ une suite définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = u_n^2 + 1$.

1. Montrer que la suite $(u_n)$ est croissante, puis que $\lim\limits_{n\to+\infty} u_n = +\infty$.

**Exercice 8 – Rang d'une suite dépassant un seuil :**

Soit $(u_n)_{n\in\mathbb{N}}$ une suite définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = u_n^2 + 1$.

1. Montrer que la suite $(u_n)$ est croissante, puis que $\lim\limits_{n\to+\infty} u_n = +\infty$.
2. Définir une fonction `rang` qui, à un réel $a$, renvoie le rang du premier terme de la suite $(u_n)_{n\in\mathbb{N}}$ qui est supérieur ou égal à $a$.

**Exercice 8 – Rang d'une suite dépassant un seuil :**

1. Pour tout $n \in \mathbb{N}$, on a :

$$u_{n+1} - u_n = u_n^2 - u_n + 1 = \mathrm{P}(u_n),$$

où $\mathrm{P} : x \longmapsto x^2 - x + 1$ est un polynôme de discriminant $< 0$. Il est donc de signe constant, du signe de son coefficient dominant 1 .
On a donc $u_{n+1} - u_n > 0$ pour tout $n \in \mathbb{N}$ et la suite est croissante.
Par théorème de limite monotone, on a alors deux cas possibles : soit $(u_n)_{n\in\mathbb{N}}$ converge vers une limite finie $\ell \in \mathbb{R}$, soit elle diverge vers $+\infty$.
Supposons qu'elle converge vers $\ell \in \mathbb{R}$. On a $u_{n+1} = u_n^2 + 1$ pour tout $n \in \mathbb{N}$.
D'où en passant à la limite, on obtient (puisque P est continue)

$$\ell = \ell^2 + 1 \iff \mathrm{P}(\ell) = 0.$$

Or ceci est impossible car P n'a pas de racine réelle.
Ainsi, on a bien que $\lim\limits_{n\to+\infty} u_n = +\infty$.

**Méthode**

Sans connaître le rang d'arrêt, une boucle `for` est impossible

### Exercice 8 – Rang d'une suite dépassant un seuil :

2

Éditeur

```
1  def rang(a) :
2      u = 1
3      n = 0
4      while u < a :
5          u = u**2 + 1
6          n = n+1
7      return n
```

**Exercice 9 :**

Soient $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ les suites définies par :

$$\forall\, n \in \mathbb{N}^*,\ u_n = \sum_{k=1}^{n} \frac{1}{k^2} \quad \text{et} \quad v_n = u_n + \frac{1}{n}.$$

1. Montrer que les suites $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont adjacentes. En déduire un encadrement de leur limite.

### Exercice 9 :

Soient $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ les suites définies par :

$$\forall\, n \in \mathbb{N}^*,\ u_n = \sum_{k=1}^{n} \frac{1}{k^2} \quad \text{et} \quad v_n = u_n + \frac{1}{n}.$$

1. Montrer que les suites $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont adjacentes. En déduire un encadrement de leur limite.
2. Écrire une fonction `approx` qui, à un réel strictement positif $\varepsilon$, associe une approximation de $\displaystyle\sum_{k=1}^{+\infty} \frac{1}{k^2}$ à $\varepsilon$ près.

**Exercice 9 :**

1. Montrons que $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont adjacentes :

Les suites $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont donc adjacentes.

**Exercice 9 :**

1. Montrons que $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont adjacentes :
   - Croissance de $(u_n)_{n\in\mathbb{N}}$.
     Pour tout $n \in \mathbb{N}$, on a :

$$u_{n+1} - u_n = \sum_{k=1}^{n+1} \frac{1}{k^2} - \sum_{k=1}^{n} \frac{1}{k^2} = \sum_{k=1}^{n} \frac{1}{k^2} + \frac{1}{(n+1)^2} - \sum_{k=1}^{n} \frac{1}{k^2} = \frac{1}{(n+1)^2} \geqslant 0.$$

Les suites $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont donc adjacentes.

**Exercice 9 :**

1. Montrons que $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont adjacentes :
   - Croissance de $(u_n)_{n\in\mathbb{N}}$.
     Pour tout $n \in \mathbb{N}$, on a :
     $$u_{n+1} - u_n = \sum_{k=1}^{n+1} \frac{1}{k^2} - \sum_{k=1}^{n} \frac{1}{k^2} = \sum_{k=1}^{n} \frac{1}{k^2} + \frac{1}{(n+1)^2} - \sum_{k=1}^{n} \frac{1}{k^2} = \frac{1}{(n+1)^2} \geqslant 0.$$
   - Décroissance de $(v_n)_{n\in\mathbb{N}}$. Pour tout $n \in \mathbb{N}$, on a :
     $$\begin{aligned} v_{n+1} - v_n &= \left(u_{n+1} + \frac{1}{n+1}\right) - \left(u_n + \frac{1}{n}\right) = \frac{1}{(n+1)^2} + \frac{1}{n+1} - \frac{1}{n} \\ &= \frac{n + n(n+1) - (n+1)^2}{n(n+1)^2} = \frac{-1}{n(n+1)^2} \leqslant 0. \end{aligned}$$

Les suites $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont donc adjacentes.

**À retenir**

$$\sum_{k=1}^{+\infty} \frac{1}{k^2} = \frac{\pi^2}{6}.$$

### Exercice 9 :

1. Montrons que $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont adjacentes :

   - Croissance de $(u_n)_{n\in\mathbb{N}}$.
     Pour tout $n \in \mathbb{N}$, on a :

$$u_{n+1} - u_n = \sum_{k=1}^{n+1} \frac{1}{k^2} - \sum_{k=1}^{n} \frac{1}{k^2} = \sum_{k=1}^{n} \frac{1}{k^2} + \frac{1}{(n+1)^2} - \sum_{k=1}^{n} \frac{1}{k^2} = \frac{1}{(n+1)^2} \geqslant 0.$$

   - Décroissance de $(v_n)_{n\in\mathbb{N}}$. Pour tout $n \in \mathbb{N}$, on a :

$$v_{n+1} - v_n = \left(u_{n+1} + \frac{1}{n+1}\right) - \left(u_n + \frac{1}{n}\right) = \frac{1}{(n+1)^2} + \frac{1}{n+1} - \frac{1}{n}$$

$$= \frac{n + n(n+1) - (n+1)^2}{n(n+1)^2} = \frac{-1}{n(n+1)^2} \leqslant 0.$$

   - Pour tout $n \in \mathbb{N}$, on a : $v_n - u_n = \left(u_n + \frac{1}{n}\right) - u_n = \frac{1}{n} \xrightarrow[n\to+\infty]{} 0.$

   Les suites $(u_n)_{n\in\mathbb{N}^*}$ et $(v_n)_{n\in\mathbb{N}^*}$ sont donc adjacentes.
   D'après le théorème des suites adjacentes, elles convergent vers une même limite $\ell$.
   De plus, on a pour tout $n \in \mathbb{N}^*$ :

$$u_n \leqslant \ell \leqslant v_n.$$

Idée

On calcule les termes de la suite dans la boucle.

### Exercice 9 :

2. On cherche le premier $n \in \mathbb{N}$ tel que $v_n - u_n = \dfrac{1}{n} \leqslant \varepsilon$.
   On utilise donc une boucle **while** :

Éditeur

```
1  def approx(eps) :
2      n = 1
3      u = 1
4      while 1/n > eps :
5          n = n+1
6          u = u+1/(n**2)
7      return(u)
```

**Exercice 10 :**

On reprend la suite de l' exercice (??) : $u_0 = \dfrac{1}{2}$ et $\forall\, n \in \mathbb{N},\ u_{n+1} = \dfrac{2u_n}{u_n + 1}$. On admet que $\lim\limits_{n \to +\infty} u_n = 1$.

1. Définir une fonction `valabs`, qui prend en entrée un flottant et qui renvoie sa valeur absolue.

**Exercice 10 :**

On reprend la suite de l' exercice (??) : $u_0 = \dfrac{1}{2}$ et $\forall\, n \in \mathbb{N},\ u_{n+1} = \dfrac{2u_n}{u_n + 1}$. On admet que $\lim\limits_{n \to +\infty} u_n = 1$.

1. Définir une fonction `valabs`, qui prend en entrée un flottant et qui renvoie sa valeur absolue.
2. À l'aide de la fonction `valabs`, écrire un programme permettant de déterminer le plus petit entier naturel $n$ pour lequel on a :

$$|u_n - 1| \leqslant 10^{-3}.$$

**Remarque**

Nous (re)verrons bientôt que la fonction valeur absolue est bien-sûr déjà définie dans Python, mais pas directement disponible. Elle nécessite d'importer le module `math`.

### Exercice 10 :

1. Voici une définition de la fonction `valabs` :

Éditeur

```
1  def valabs(x) :
2      if x < 0 :
3          return -x
4      else :
5          return x
```

**Remarque**

Nous (re)verrons bientôt que la fonction valeur absolue est bien-sûr déjà définie dans Python, mais pas directement disponible. Elle nécessite d'importer le module `math`.

### Exercice 10 :

1. Voici une définition de la fonction `valabs` :

Éditeur

```
1  def valabs(x) :
2      if x < 0 :
3          return -x
4      else :
5          return x
```

2. Cela nécessite l'utilisation d'une boucle **`while`**. On peut procéder comme suit :

Éditeur

```
1  u = 1/2
2  n = 0
3  while valabs(u-1) > 10**(-3) :
4      u = (2*u) / (u+1)
5      n = n+1
6  print(n)
```

On trouve $n = 10$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

On considère une fonction $f$ continue et strictement monotone sur un intervalle $[a\,;b]$ (avec $a<b$ ) telle que $f(a)f(b)<0$.
Le théorème de la bijection assure que l'équation $f(x)=0$ possède une unique solution $\alpha$ sur $]a\,;b[$.
On construit deux suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies de la manière suivante :
$a_0=a$, $b_0=b$ et, pour tout entier naturel $n$,

- Si $f(a_n)\,f\left(\dfrac{a_n+b_n}{2}\right)\leqslant 0$, alors $\alpha\in\left]a_n\,;\dfrac{a_n+b_n}{2}\right[$. On pose $a_{n+1}=a_n$ et $b_{n+1}=\dfrac{a_n+b_n}{2}$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

On considère une fonction $f$ continue et strictement monotone sur un intervalle $[a\,;b]$ (avec $a<b$ ) telle que $f(a)f(b)<0$.
Le théorème de la bijection assure que l'équation $f(x)=0$ possède une unique solution $\alpha$ sur $]a\,;b[$.
On construit deux suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies de la manière suivante :
$a_0=a$, $b_0=b$ et, pour tout entier naturel $n$,

- Si $f(a_n)\,f\left(\dfrac{a_n+b_n}{2}\right)\leqslant 0$, alors $\alpha\in\left]a_n\,;\dfrac{a_n+b_n}{2}\right[$. On pose $a_{n+1}=a_n$ et $b_{n+1}=\dfrac{a_n+b_n}{2}$.
- Si $f(a_n)\,f\left(\dfrac{a_n+b_n}{2}\right)>0$, alors $\alpha\in\left]\dfrac{a_n+b_n}{2}\,;b_n\right[$. On pose $a_{n+1}=\dfrac{a_n+b_n}{2}$ et $b_{n+1}=b_n$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

$a_0 = a$, $b_0 = b$ et, pour tout entier naturel $n$,

- Si $f(a_n) f\left(\dfrac{a_n + b_n}{2}\right) \leqslant 0$, alors $\alpha \in \left]a_n ; \dfrac{a_n + b_n}{2}\right[$. On pose $a_{n+1} = a_n$ et $b_{n+1} = \dfrac{a_n + b_n}{2}$.
- Si $f(a_n) f\left(\dfrac{a_n + b_n}{2}\right) > 0$, alors $\alpha \in \left]\dfrac{a_n + b_n}{2} ; b_n\right[$. On pose $a_{n+1} = \dfrac{a_n + b_n}{2}$ et $b_{n+1} = b_n$.

1. Étude des suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ :

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

$a_0 = a$, $b_0 = b$ et, pour tout entier naturel $n$,

- Si $f(a_n) f\left(\dfrac{a_n + b_n}{2}\right) \leqslant 0$, alors $\alpha \in \left]a_n ; \dfrac{a_n + b_n}{2}\right[$. On pose $a_{n+1} = a_n$ et $b_{n+1} = \dfrac{a_n + b_n}{2}$.
- Si $f(a_n) f\left(\dfrac{a_n + b_n}{2}\right) > 0$, alors $\alpha \in \left]\dfrac{a_n + b_n}{2} ; b_n\right[$. On pose $a_{n+1} = \dfrac{a_n + b_n}{2}$ et $b_{n+1} = b_n$.

1. Étude des suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ :
   1. Montrer que les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ sont adjacentes et convergent vers $\alpha$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

$a_0 = a$, $b_0 = b$ et, pour tout entier naturel $n$,

- Si $f(a_n) f\left(\dfrac{a_n + b_n}{2}\right) \leqslant 0$, alors $\alpha \in \left]a_n ; \dfrac{a_n + b_n}{2}\right[$. On pose $a_{n+1} = a_n$ et $b_{n+1} = \dfrac{a_n + b_n}{2}$.
- Si $f(a_n) f\left(\dfrac{a_n + b_n}{2}\right) > 0$, alors $\alpha \in \left]\dfrac{a_n + b_n}{2} ; b_n\right[$. On pose $a_{n+1} = \dfrac{a_n + b_n}{2}$ et $b_{n+1} = b_n$.

1. Étude des suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ :
   1. Montrer que les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ sont adjacentes et convergent vers $\alpha$.
   2. Montrer que pour tout $n \in \mathbb{N}$, on a $b_{n+1} - a_{n+1} = \dfrac{b_n - a_n}{2}$.

      Puis que $b_n - a_n = \dfrac{b-a}{2^n}$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

1. 1. Commençons pour cela par montrer par récurrence que pour tout $n \in \mathbb{N}$, on a :
$$a_n \leqslant a_{n+1} \leqslant b_{n+1} \leqslant b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n) f(b_n) \leqslant 0.$$

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

1. 1. Commençons pour cela par montrer par récurrence que pour tout $n \in \mathbb{N}$, on a :

$$a_n \leqslant a_{n+1} \leqslant b_{n+1} \leqslant b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n) f(b_n) \leqslant 0.$$

**Init :** Par construction, on a soit $a_1 = a_0$ et $b_1 = \dfrac{a_0 + b_0}{2}$, soit

$a_1 = \dfrac{a_0 + b_0}{2}$ et $b_1 = b_0$.
Comme $a_0 = a < b = b_0$, on a ainsi l'inégalité $a_0 \leqslant a_1 \leqslant b_1 \leqslant b_0$.
D'autre part, $b_0 - a_0 = b - a = \dfrac{b-a}{2^0}$ et
$f(a_0) f(b_0) = f(a) f(b) \leqslant 0$ par hypothèse.
Donc la propriété est vraie au rang $n = 0$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

1 1 **Hér :** Soit $n$ un entier naturel. Supposons la propriété au rang $n$ vraie. Montrons la propriété au rang $n+1$.
Par construction, on a soit $a_{n+2} = a_{n+1}$ et $b_{n+2} = \dfrac{a_{n+1}+b_{n+1}}{2}$, soit $a_{n+2} = \dfrac{a_{n+1}+b_{n+1}}{2}$.
Par hypothèse de récurrence, $a_{n+1} \leqslant b_{n+1}$.
On a donc, dans les deux cas, l'inégalité $a_{n+1} \leqslant a_{n+2} \leqslant b_{n+2} \leqslant b_{n+1}$.

De plus, par construction, on a $b_{n+1} - a_{n+1} = \dfrac{b_n - a_n}{2}$.

Par hypothèse de récurrence, $b_n - a_n = \dfrac{b-a}{2^n}$.

Donc $b_{n+1} - a_{n+1} = \dfrac{b-a}{2^{n+1}}$.

Enfin, comme par hypothèse de récurrence $f(a_n)\,f(b_n) \leqslant 0$, $f(a_n)$ et $f(b_n)$ sont de signe opposé.

Donc $f(m_{n+1}) = f\left(\dfrac{a_n+b_n}{2}\right)$ n'a pas le même signe que $f(a_n)$ ou $f(b_n)$.

Par définition de $a_{n+1}$ et de $b_{n+1}$, on a bien $f(a_{n+1})\,f(b_{n+1}) \leqslant 0$. Donc la propriété au rang $n+1$ est vraie.

### Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

1. 1. **Concl :** Par le principe de récurrence, on a bien pour tout $n \in \mathbb{N}$ :

$$a_n \leqslant a_{n+1} \leqslant b_{n+1} \leqslant b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n)\, f(b_n) \leqslant 0.$$

### Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

1. 1. **Concl :** Par le principe de récurrence, on a bien pour tout $n \in \mathbb{N}$ :

$$a_n \leqslant a_{n+1} \leqslant b_{n+1} \leqslant b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n)\, f(b_n) \leqslant 0.$$

On en déduit donc que $(a_n)_{n\in\mathbb{N}}$ est croissante, $(b_n)_{n\in\mathbb{N}}$ est décroissante, et on a :

$$\forall\, n \in \mathbb{N}, \quad b_n - a_n = \frac{b-a}{2^n} \xrightarrow[n\to+\infty]{} 0.$$

Les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ sont donc adjacentes et, d'après le théorème des suites adjacentes, convergent vers la même limite $\ell \in ]a\,;b[$.
Reste enfin à montrer que $\ell = \alpha$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

1 1 **Concl :** Par le principe de récurrence, on a bien pour tout $n \in \mathbb{N}$ :

$$a_n \leqslant a_{n+1} \leqslant b_{n+1} \leqslant b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n)\, f(b_n) \leqslant 0.$$

On en déduit donc que $(a_n)_{n\in\mathbb{N}}$ est croissante, $(b_n)_{n\in\mathbb{N}}$ est décroissante, et on a :

$$\forall\, n \in \mathbb{N}, \quad b_n - a_n = \frac{b-a}{2^n} \xrightarrow[n\to+\infty]{} 0.$$

Les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ sont donc adjacentes et, d'après le théorème des suites adjacentes, convergent vers la même limite $\ell \in ]a\,;b[$.
Reste enfin à montrer que $\ell = \alpha$.

En passant à la limite dans l'inégalité $f(a_n)\, f(b_n) \leqslant 0$ où $f$ est continue sur $[a\,;b]$ :

$$\lim_{n\to+\infty} (f(a_n) \cdot f(b_n)) = \left(\lim_{n\to+\infty} f(a_n)\right)\left(\lim_{n\to+\infty} f(b_n)\right) = (f(\ell))^2 \leqslant 0.$$

D'où $(f(\ell))^2 \leqslant 0$, ce qui implique $f(\ell) = 0$. Et comme par hypothèse $f$ s'annule en une seule valeur $\alpha \in ]a\,;b[$, on a donc $\ell = \alpha$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

1. 2. On l'a démontré à la question précédente, on a pour tout $n \in \mathbb{N}$ :

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n).$$

La suite $(b_n - a_n)_n$ est donc géométrique de raison $\frac{1}{2}$. *i.e.*

$$\forall\, n \in \mathbb{N}, \quad b_n - a_n = \left(\frac{1}{2}\right)^n (b_0 - a_0).$$

### Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. On considère l'équation (E) : $1 - x - x^3 = 0$, où $x$ désigne un réel.
   On note $f$ la fonction définie sur $\mathbb{R}$ par $f(x) = 1 - x - x^3$.
   1. Construire une fonction `f` permettant de calculer des valeurs de $f$.

### Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. On considère l'équation (E) : $1 - x - x^3 = 0$, où $x$ désigne un réel.
   On note $f$ la fonction définie sur $\mathbb{R}$ par $f(x) = 1 - x - x^3$.
   1. Construire une fonction `f` permettant de calculer des valeurs de $f$.
   2. Justifier que (E) possède une unique solution $\alpha$, et vérifier que $\alpha$ appartient à $]0\,;1[$.

### Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. On considère l'équation (E) : $1 - x - x^3 = 0$, où $x$ désigne un réel.
   On note $f$ la fonction définie sur $\mathbb{R}$ par $f(x) = 1 - x - x^3$.
   1. Construire une fonction `f` permettant de calculer des valeurs de $f$.
   2. Justifier que (E) possède une unique solution $\alpha$, et vérifier que $\alpha$ appartient à $]0\,;1[$.
   3. Écrire un script permettant de trouver et d'afficher une valeur approchée de $\alpha$ à $0,001$ près.

**Remarque**

Pour éviter des erreurs de cumul d'arrondis, il est meilleur de calculer $1 - x - x^3$ que $-x^3 - x + 1$ sur $[0\,;1]$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. 1. On peut utiliser le script suivant :

Éditeur

```
1  def f(x) :
2      return 1-x-x**3
```

**Remarque**

Pour éviter des erreurs de cumul d'arrondis, il est meilleur de calculer $1-x-x^3$ que $-x^3-x+1$ sur $[0\,;1]$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. 1. On peut utiliser le script suivant :

Éditeur

```
1  def f(x) :
2      return 1-x-x**3
```

2. On applique le théorème de la bijection :

**Remarque**

Pour éviter des erreurs de cumul d'arrondis, il est meilleur de calculer $1-x-x^3$ que $-x^3-x+1$ sur $[0\,;1]$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. 1. On peut utiliser le script suivant :

Éditeur

```
1  def f(x) :
2      return 1-x-x**3
```

2. On applique le théorème de la bijection :
   - La fonction $f$ est polynomiale, donc continue sur $\mathbb{R}$.

**Remarque**

Pour éviter des erreurs de cumul d'arrondis, il est meilleur de calculer $1 - x - x^3$ que $-x^3 - x + 1$ sur $[0\,;1]$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. 1. On peut utiliser le script suivant :

Éditeur

```
1  def f(x) :
2      return 1-x-x**3
```

2. On applique le théorème de la bijection :
   - La fonction $f$ est polynomiale, donc continue sur $\mathbb{R}$.
   - $f$ étant polynomiale, elle est aussi de classe $\mathcal{C}^1$ sur $\mathbb{R}$, et on a :

$$\forall\, x \in \mathbb{R}, \quad f'(x) = -1 - 3x^2 < 0.$$

   Donc $f$ est strictement décroissante sur $\mathbb{R}$.

**Remarque**

Pour éviter des erreurs de cumul d'arrondis, il est meilleur de calculer $1-x-x^3$ que $-x^3-x+1$ sur $[0\,;1]$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. 1. On peut utiliser le script suivant :

Éditeur

```
1  def f(x) :
2      return 1-x-x**3
```

2. On applique le théorème de la bijection :
   - La fonction $f$ est polynomiale, donc continue sur $\mathbb{R}$.
   - $f$ étant polynomiale, elle est aussi de classe $\mathcal{C}^1$ sur $\mathbb{R}$, et on a :
   $$\forall\, x \in \mathbb{R}, \quad f'(x) = -1 - 3x^2 < 0.$$
   Donc $f$ est strictement décroissante sur $\mathbb{R}$.
   - De plus, on a :
   $$-1 = f(1) < 0 = f(\alpha) < f(0) = 1.$$

**Remarque**

Pour éviter des erreurs de cumul d'arrondis, il est meilleur de calculer $1 - x - x^3$ que $-x^3 - x + 1$ sur $[0\,;1]$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

2. 1. On peut utiliser le script suivant :

Éditeur

```
1  def f(x) :
2      return 1-x-x**3
```

2. On applique le théorème de la bijection :
   - La fonction $f$ est polynomiale, donc continue sur $\mathbb{R}$.
   - $f$ étant polynomiale, elle est aussi de classe $\mathcal{C}^1$ sur $\mathbb{R}$, et on a :
   $$\forall\, x \in \mathbb{R}, \quad f'(x) = -1 - 3x^2 < 0.$$
   Donc $f$ est strictement décroissante sur $\mathbb{R}$.
   - De plus, on a :
   $$-1 = f(1) < 0 = f(\alpha) < f(0) = 1.$$

Par le théorème de la bijection, l'équation $f(x) = 0$ admet une unique solution $\alpha$ sur $[0\,;1]$.
Par stricte décroissance de $f$, on en déduit que $\alpha \in ]0, 1[$.

## Exercice 11 – (⋆⋆) - Raisonnement par dichotomie :

❷ ❸ On peut utiliser le script suivant :

Éditeur

```
a = 0
b = 1
while b-a>10**(-3):
    m = (a+b)/2
    if f(a)*f(m) <= 0 :
        b = m
    else :
        a = m
print(a)
```