Introduction à la science informatique

Semaine 2

Algorithmes formalisés

C’est quoi un algorithme ?

- La description non ambiguë d’une séquence finie d’instructions permettant de résoudre un problème

- Finitude = termine après un nombre fini d’étapes

- Non ambigu = précis (en termes d’opérations élémentaires)

- Entrées = données

- Sorties = résultat attendu

محمد بن موسى الخوارزمي

Muḥammad ibn Mūsā al-Khwārizmī,
auteur de الجبر (al-Jabr)

Résoudre un problème

- On cherche un algorithme
- On le décrit précisément, de manière non ambigüe
- On prouve qu'il est correct
- On vérifie qu'il est efficace (idéalement, on choisit l'algorithme optimal)
- On le met en œuvre
- On le teste

} en séance de TP

Décrire des algorithmes

- En langage naturel (par exemple, en français avec mon accent italien)

- En langage de programmation (formel) : par exemple, en Python comme ici

Recherche dans une séquence en langage naturel

- Pour chaque élément de la séquence à partir du premier :
    - Si cet élément est l'élément cherché, on a terminé
    - Sinon, on continue avec l'élément suivant
- S'il n'y a plus d'éléments et on n'a pas trouvé ce qu'on cherchait, alors il n'est pas là

Recherche dans une séquence en Python 🐍

```
def chercher(element, sequence):
    n = len(sequence)
    i = 0
    while i < n:
        if sequence[i] == element:
            return i
        i = i + 1
    return -1
```

Algorithmes en Python

Variables, affectations et séquence

Un morceau de code

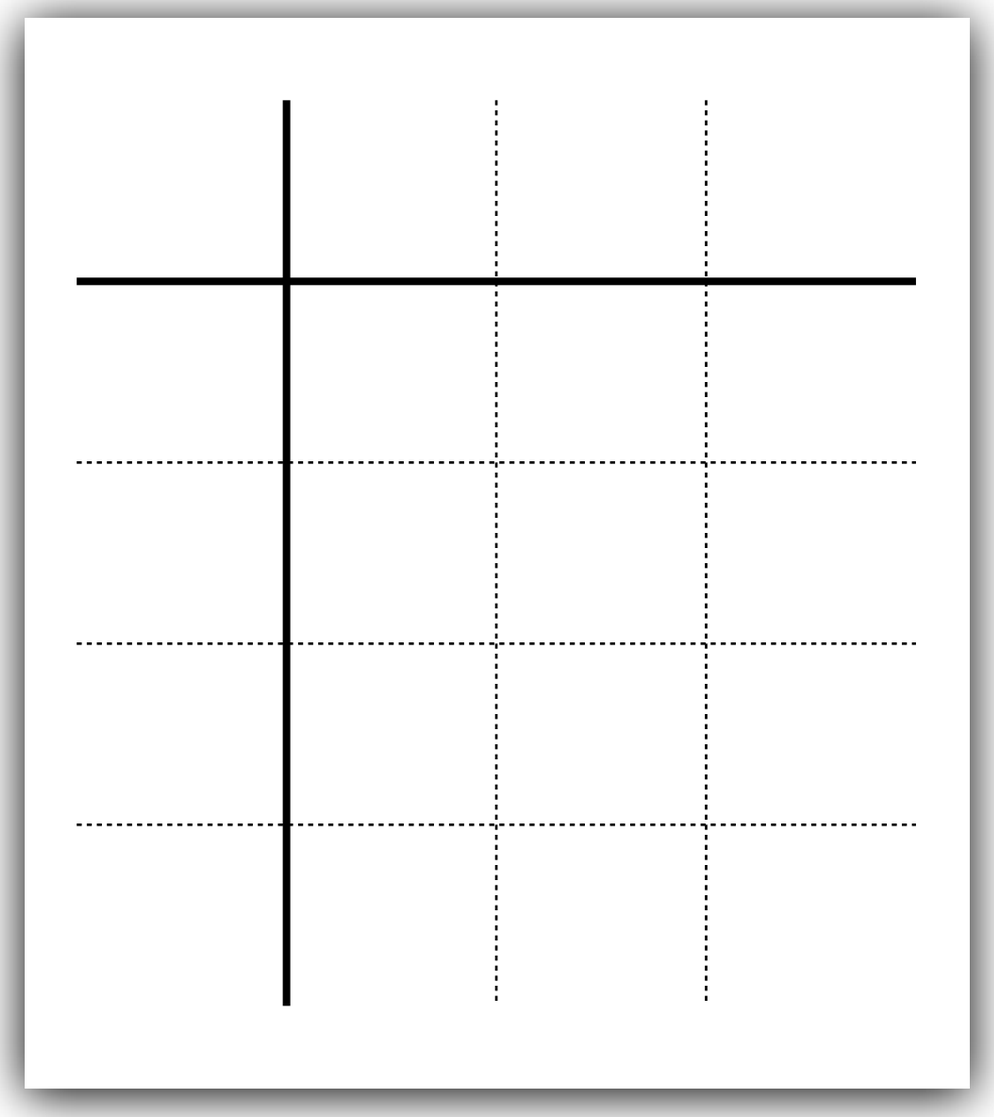
```
x = 5
z = x + 3
y = 7
z = x + y
```

Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

Un morceau de code

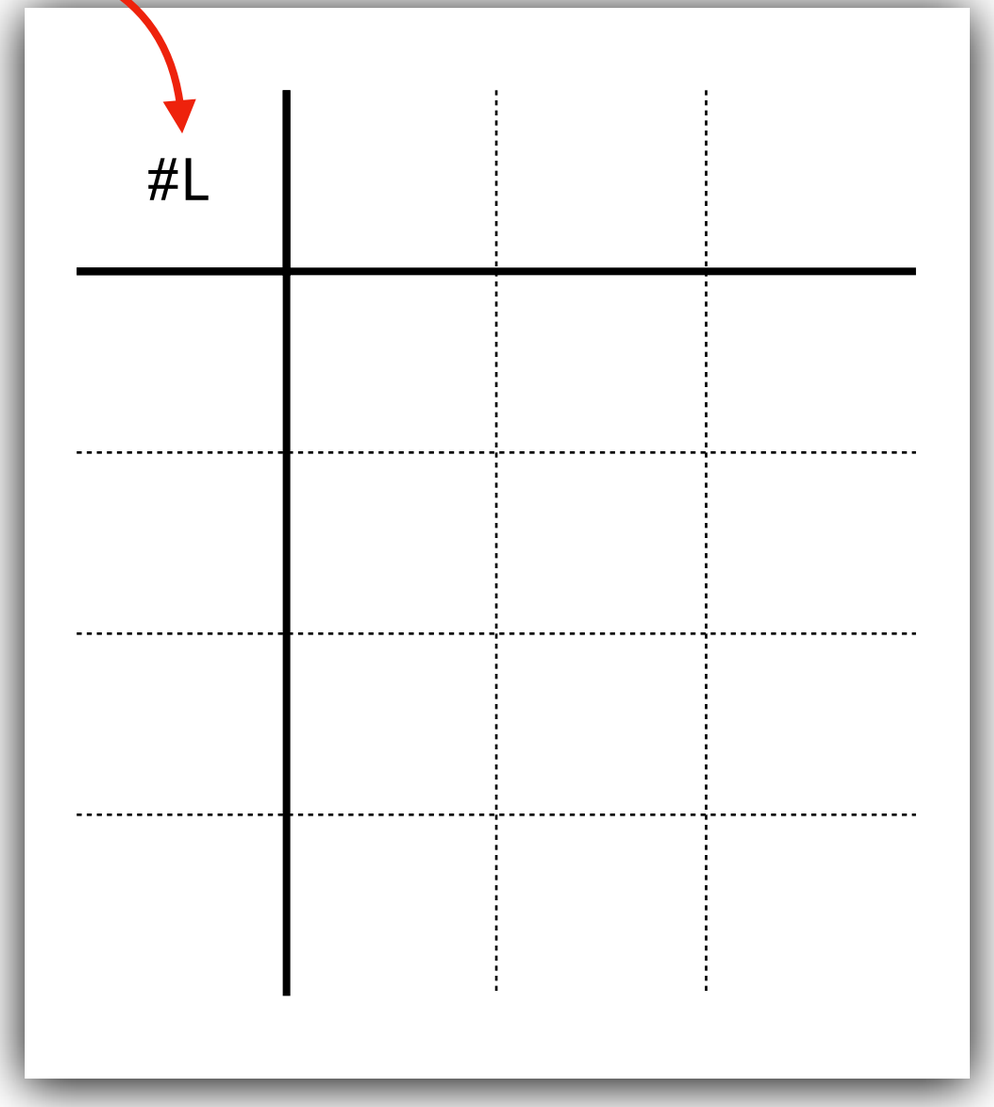
```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```



Un morceau de code

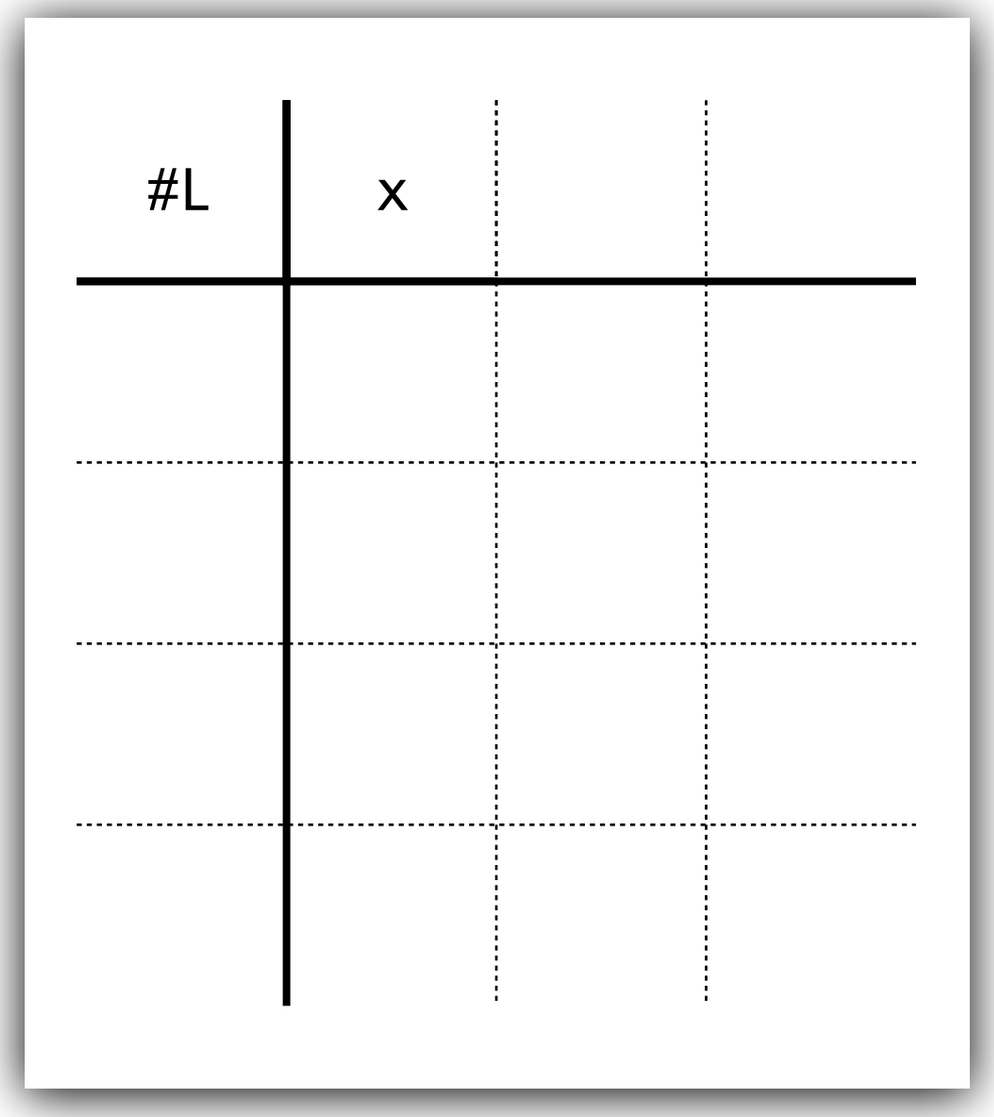
numéro
de ligne

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```



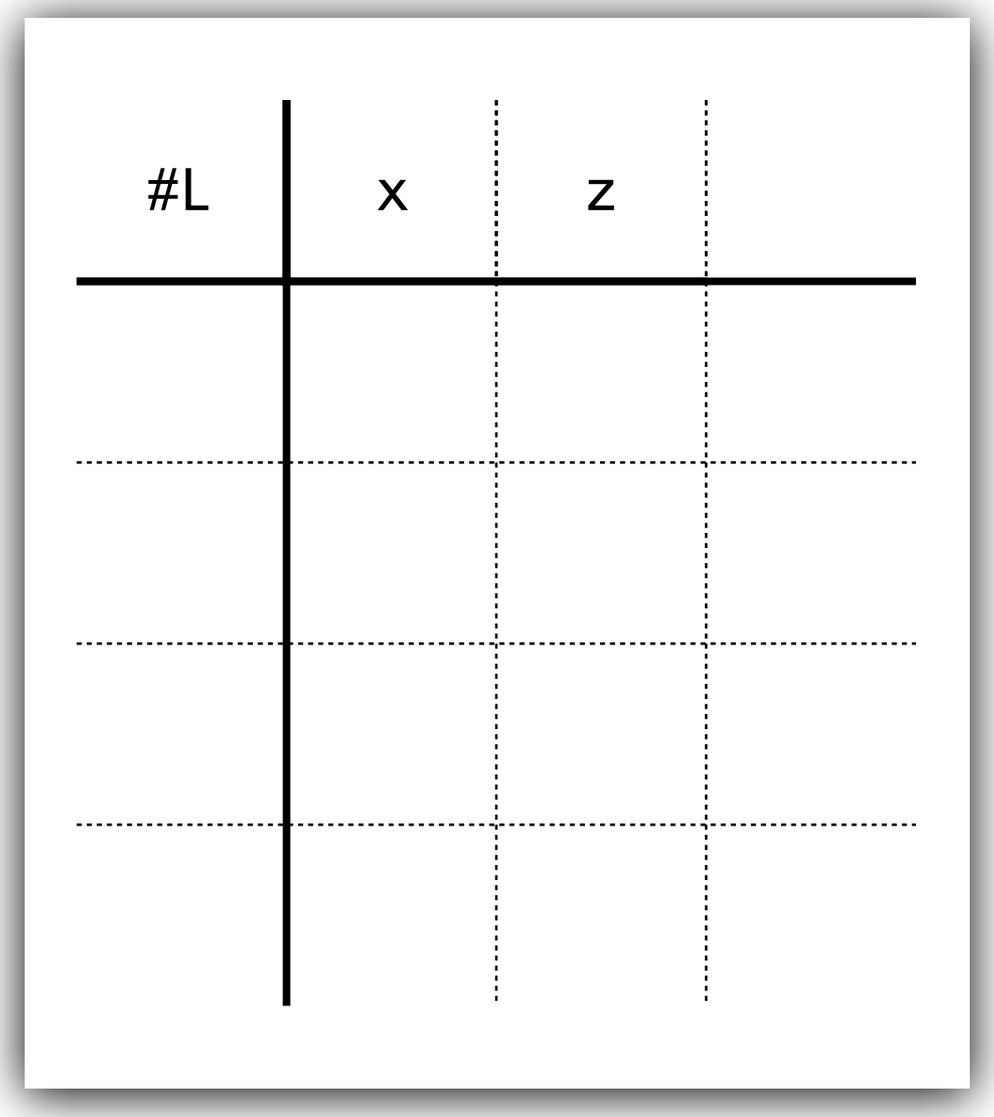
Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```



Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

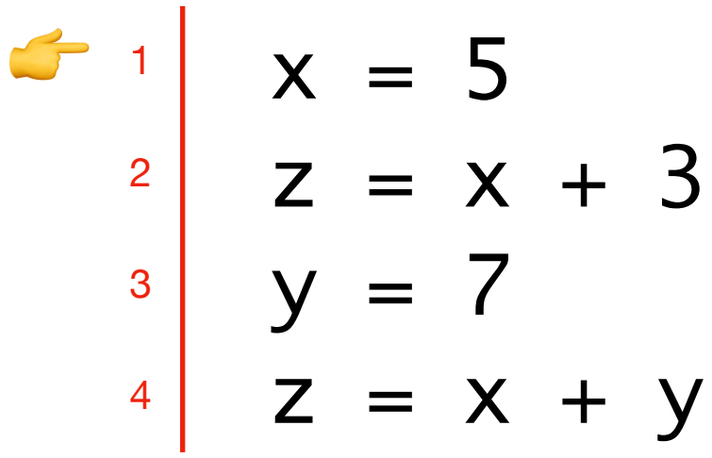


Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

#L	x	z	y

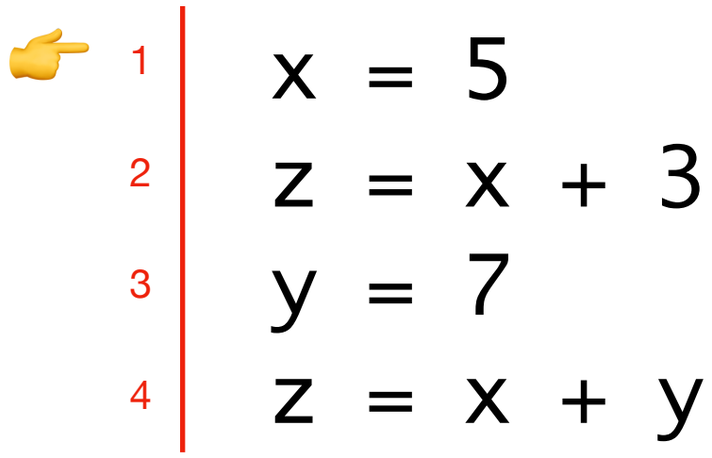
Un morceau de code



```
1   x = 5
2   z = x + 3
3   y = 7
4   z = x + y
```

#L	x	z	y
1			

Un morceau de code



```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

#L	x	z	y
1	5		

Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

👉

#L	x	z	y
1	5		
2			

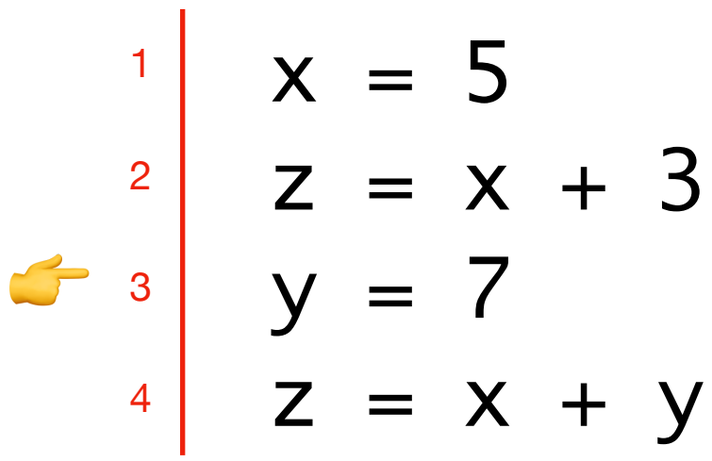
Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

👉

#L	x	z	y
1	5		
2		8	

Un morceau de code



#L	x	z	y
1	5		
2		8	
3			

Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

👉

#L	x	z	y
1	5		
2		8	
3			7

Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

👉 4

#L	x	z	y
1	5		
2		8	
3			7
4			

Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

👉

#L	x	z	y
1	5		
2		8	
3			7
4		12	

Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

#L	x	z	y
1	5		
2		8	
3			7
4		12	

Un morceau de code

```
1  x = 5
2  z = x + 3
3  y = 7
4  z = x + y
```

#L	x	z	y
1	5		
2		8	
3			7
4		12	

4 op

Exercice 1 du TD2

Définition de fonctions

Définition de fonctions

```
def carré(x):
    y = x * x
    return y
```

Calcul du carré de 4

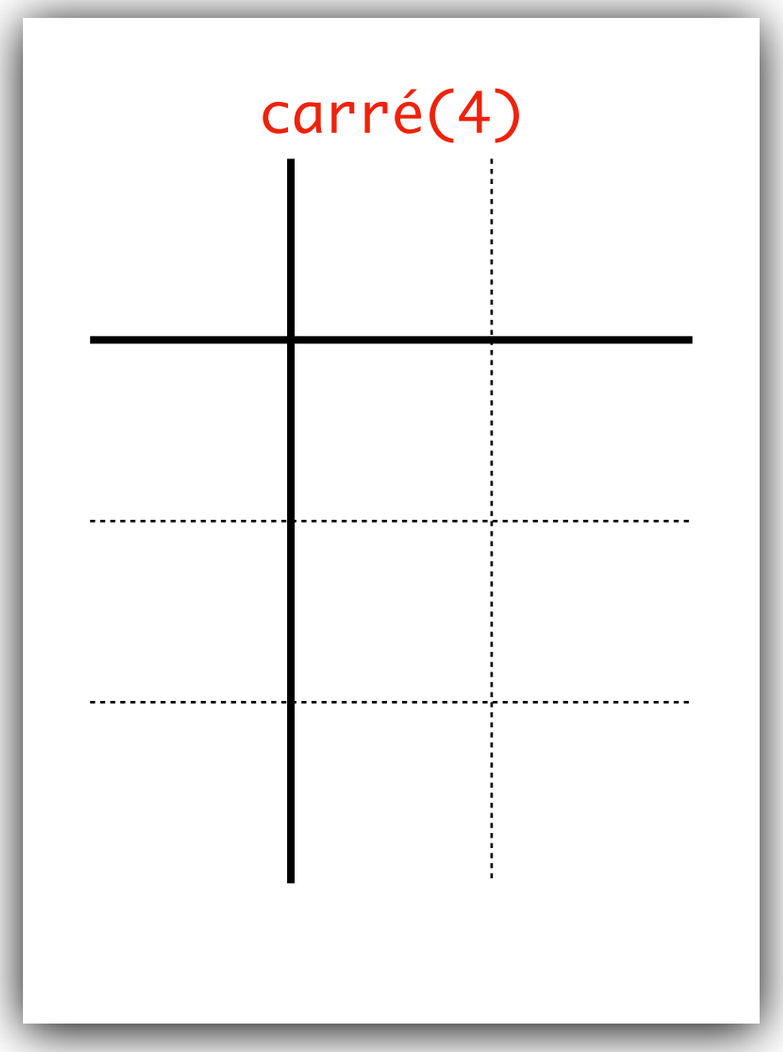
```
1  def carré(x):
2      y = x * x
3      return y
```

```
>>> carré(4)
```

Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

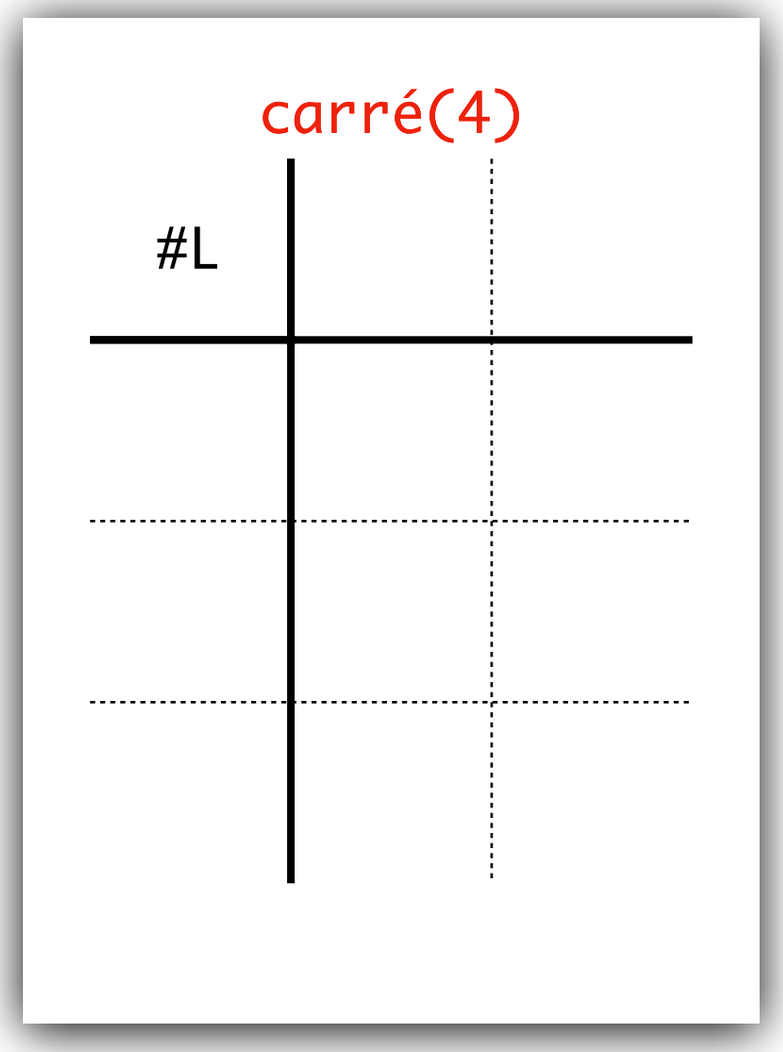
```
>>> carré(4)
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

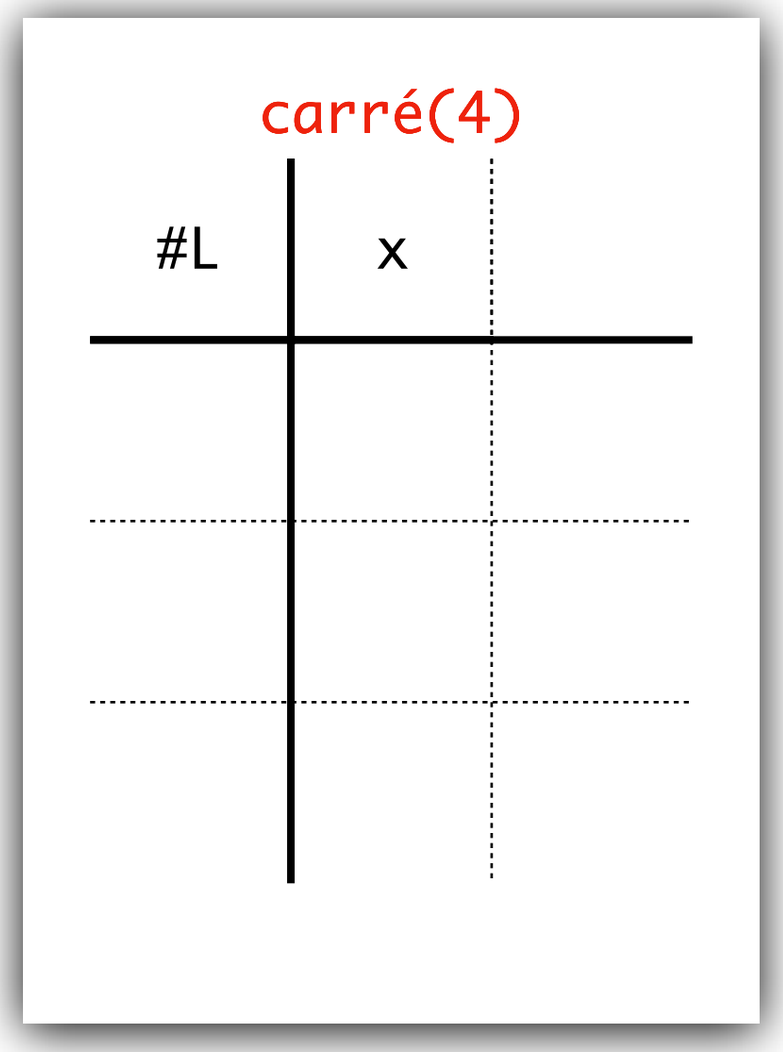
```
>>> carré(4)
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

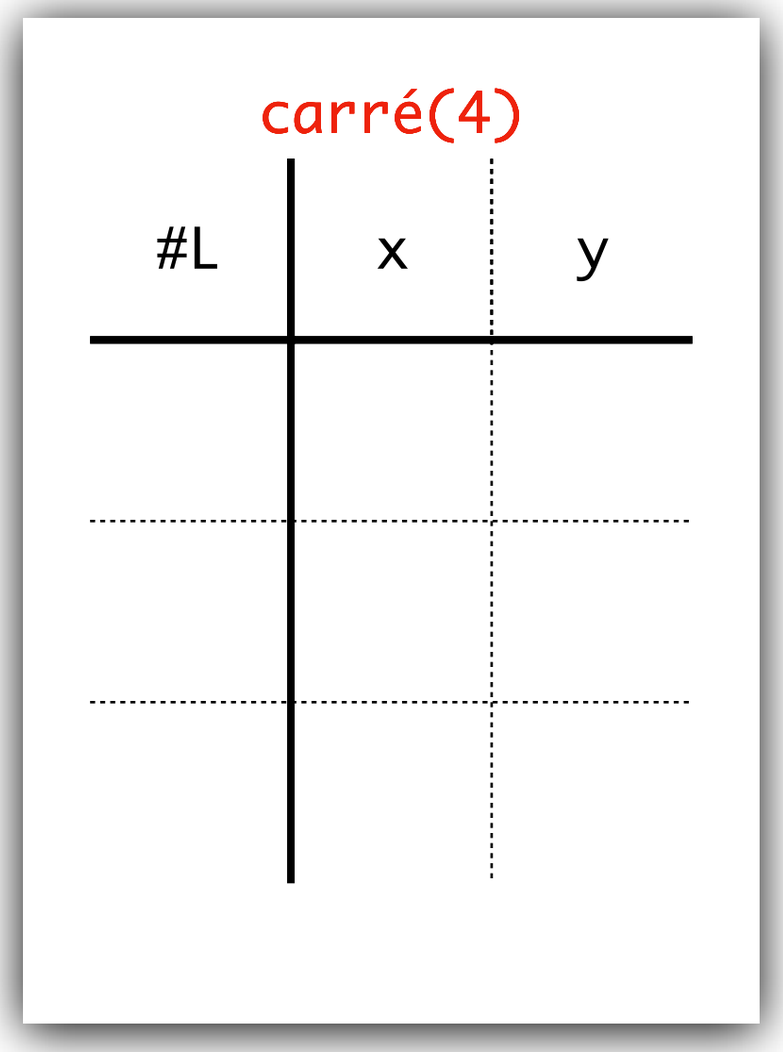
```
>>> carré(4)
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

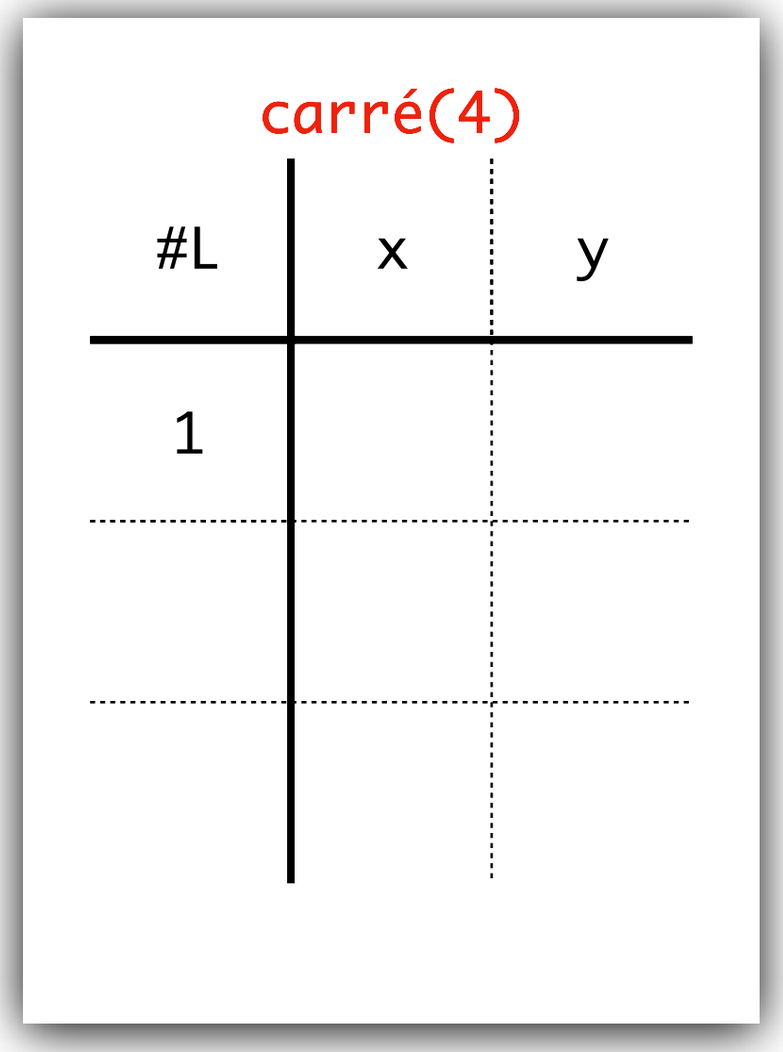
```
>>> carré(4)
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

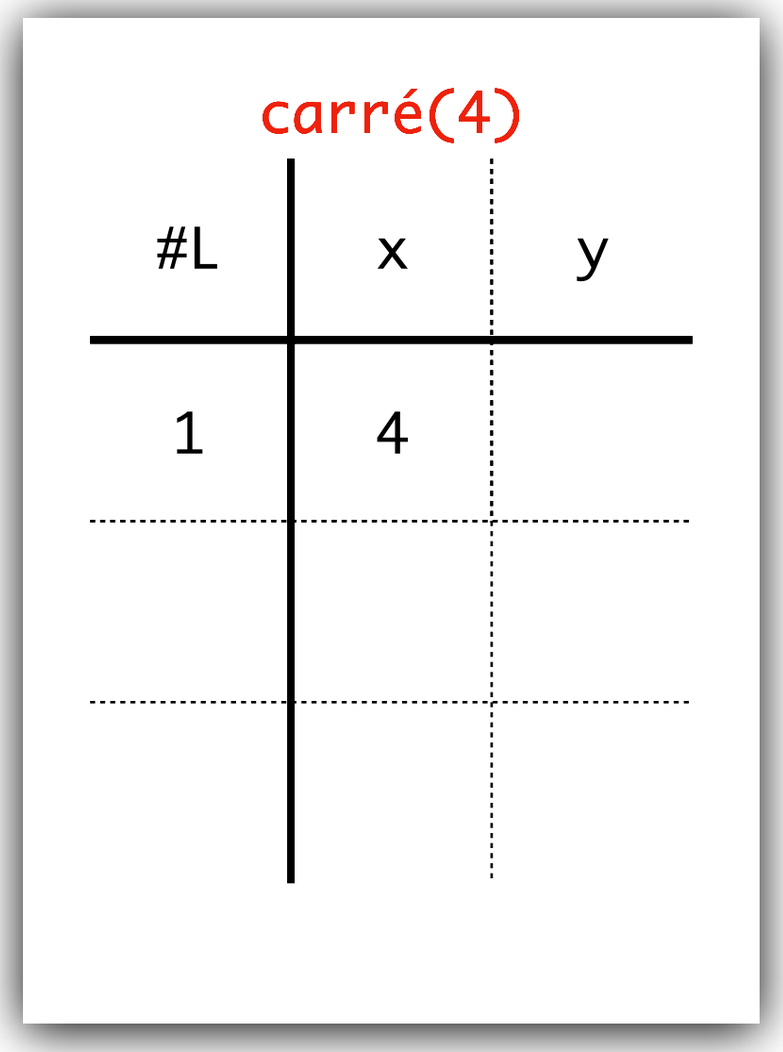
```
>>> carré(4)
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

```
>>> carré(4)
```

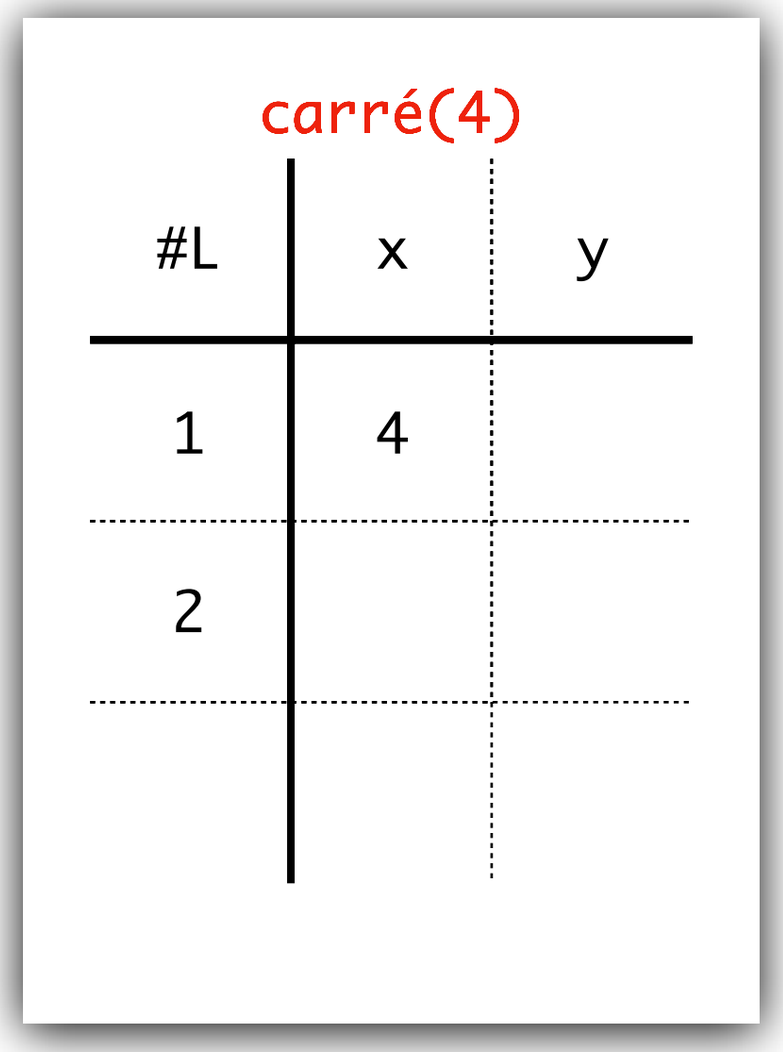


Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

👉

```
>>> carré(4)
```

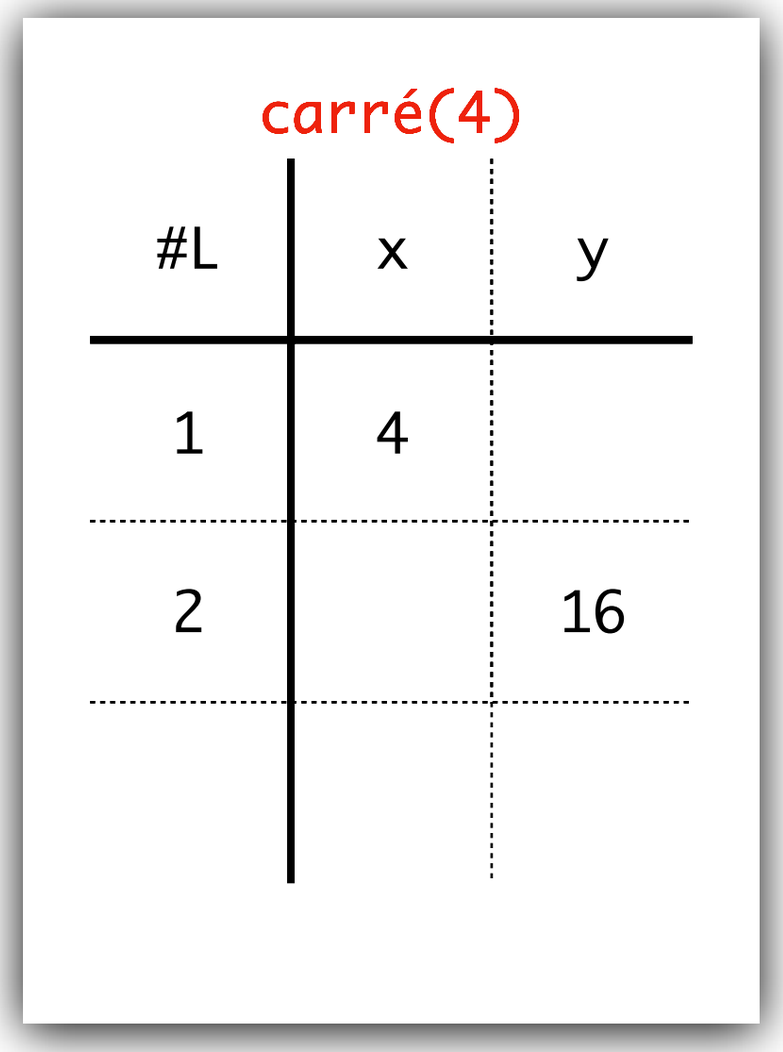


Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

👉

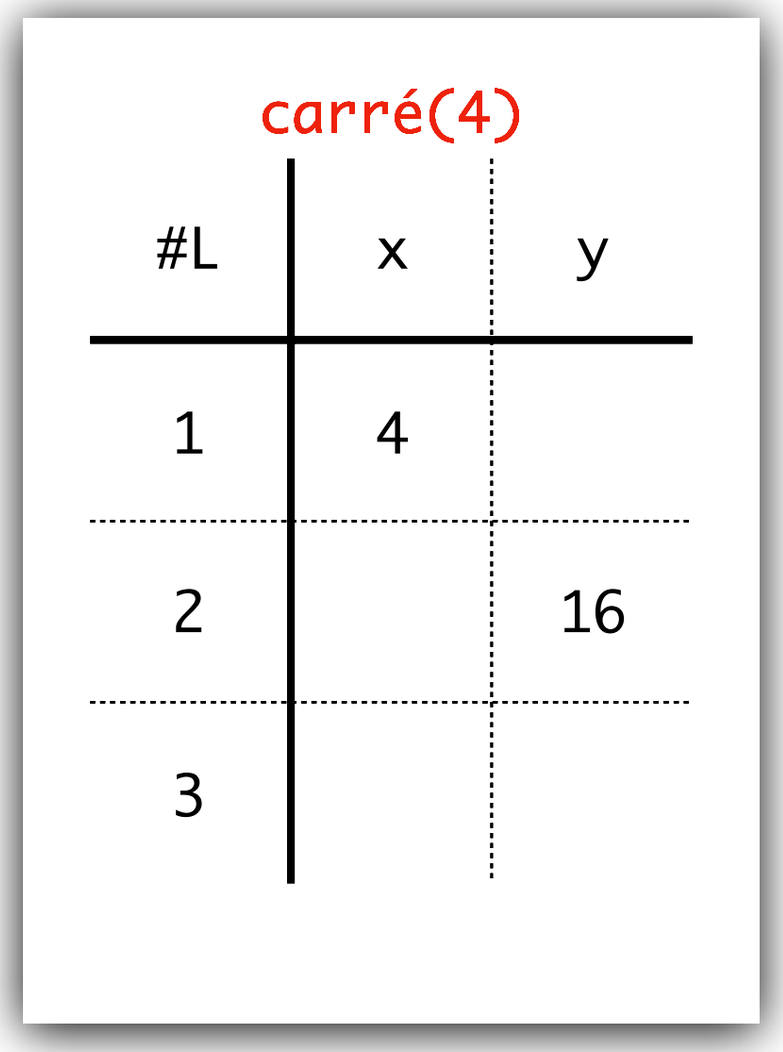
```
>>> carré(4)
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

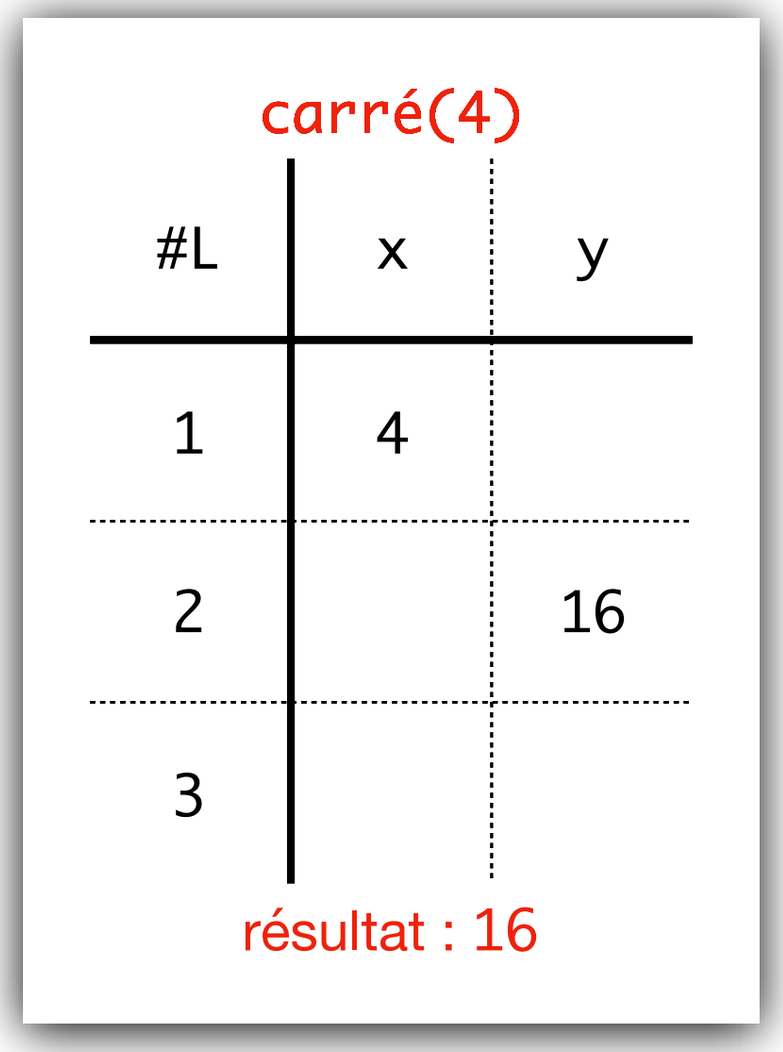
```
>>> carré(4)
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

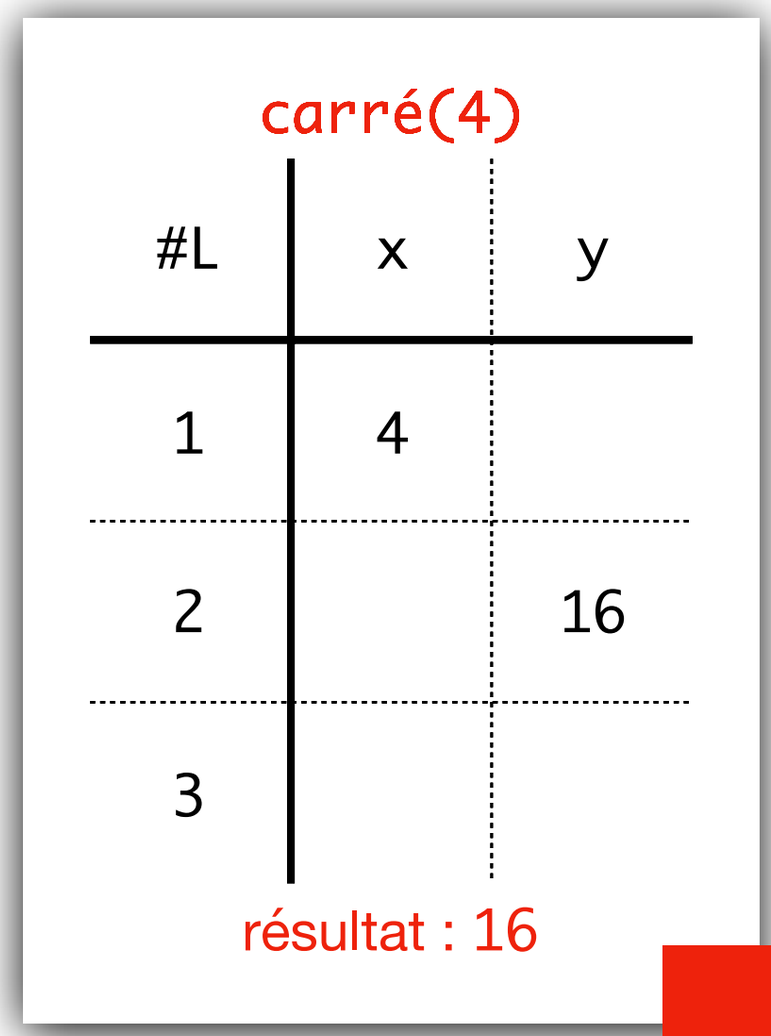
```
>>> carré(4)
16
```



Calcul du carré de 4

```
1  def carré(x):
2      y = x * x
3      return y
```

```
>>> carré(4)
16
```



3 op

Exercice 2 du TD2

Discriminant de $ax^2 + bx + c$

```
def carré(x):
    y = x * x
    return y
```

Discriminant de $ax^2 + bx + c$

```
def carré(x):
    y = x * x
    return y


def discriminant(a, b, c):
    d = carré(b) - 4*a*c
    return d
```

Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

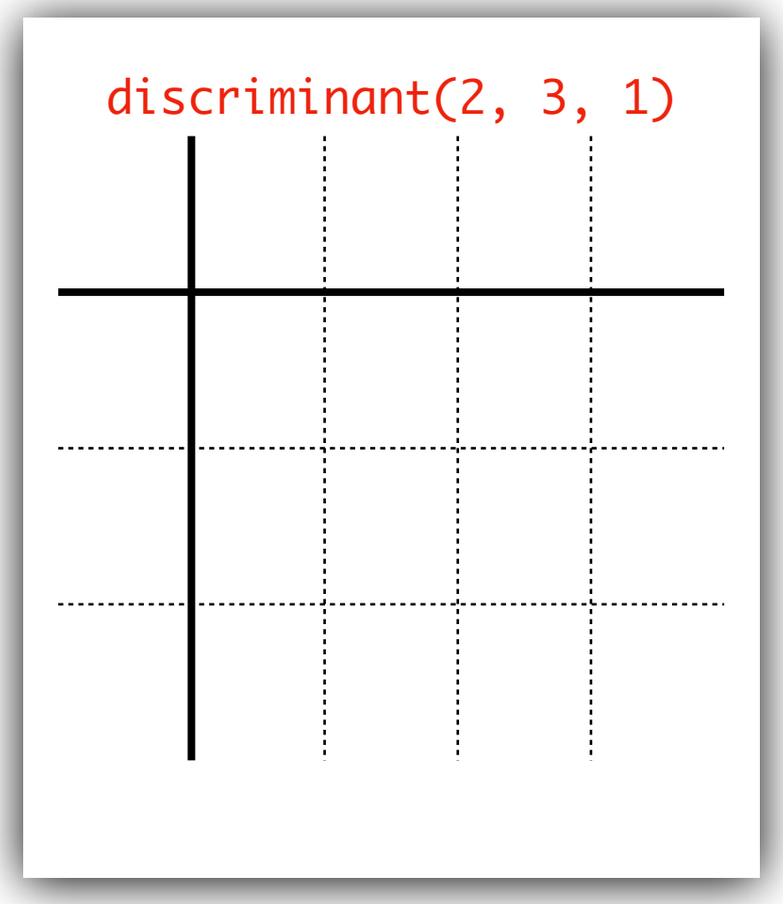
```
>>> discriminant(2, 3, 1)
```

Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

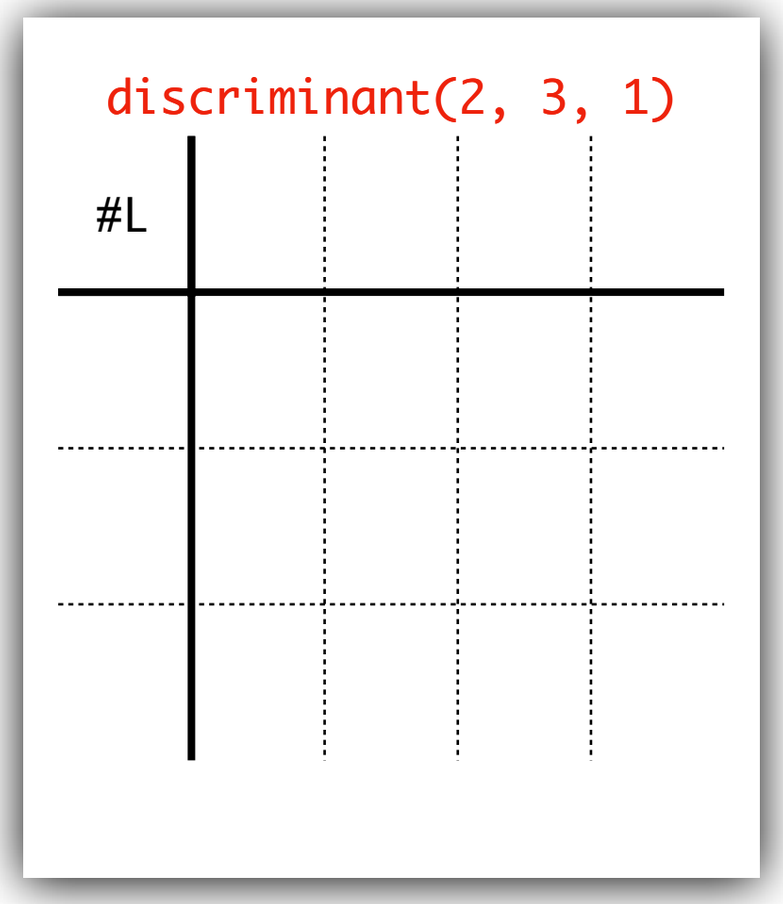


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

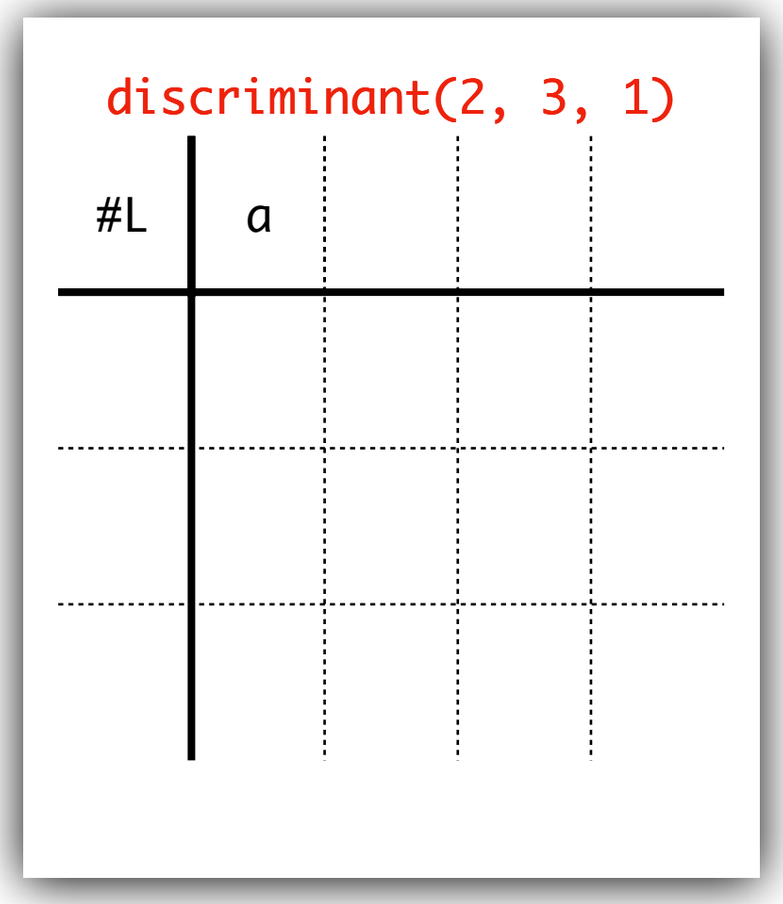


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

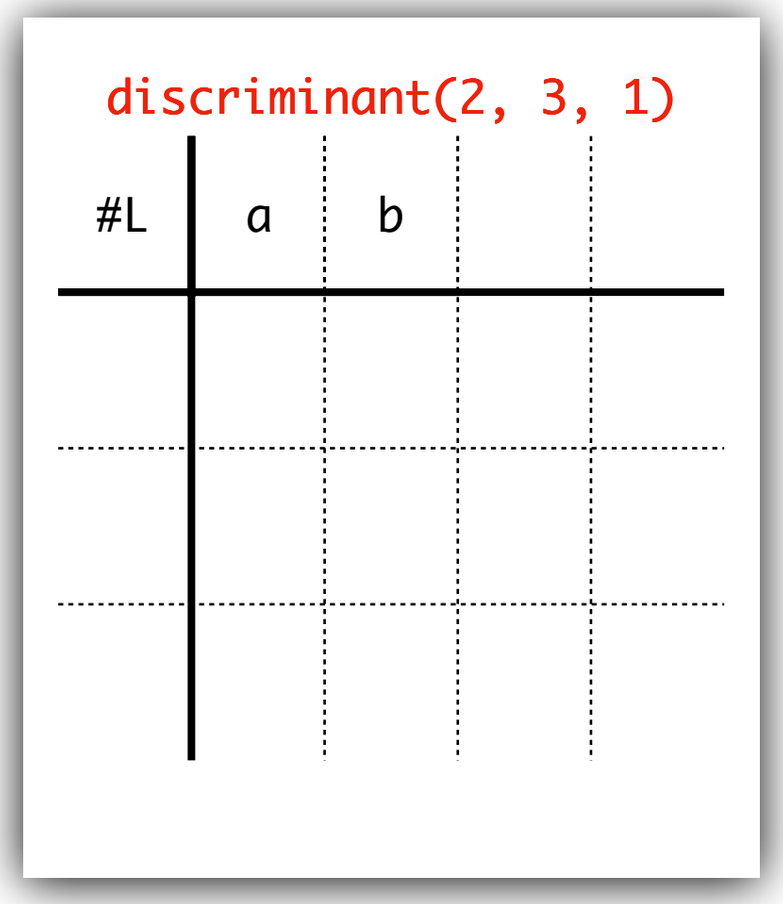


Discriminant de $2x^2 + 3x + 1$

```
1 def carré(x):
2     y = x * x
3     return y
```

```
4 def discriminant(a, b, c):
5     d = carré(b) - 4*a*c
6     return d
```

```
>>> discriminant(2, 3, 1)
```

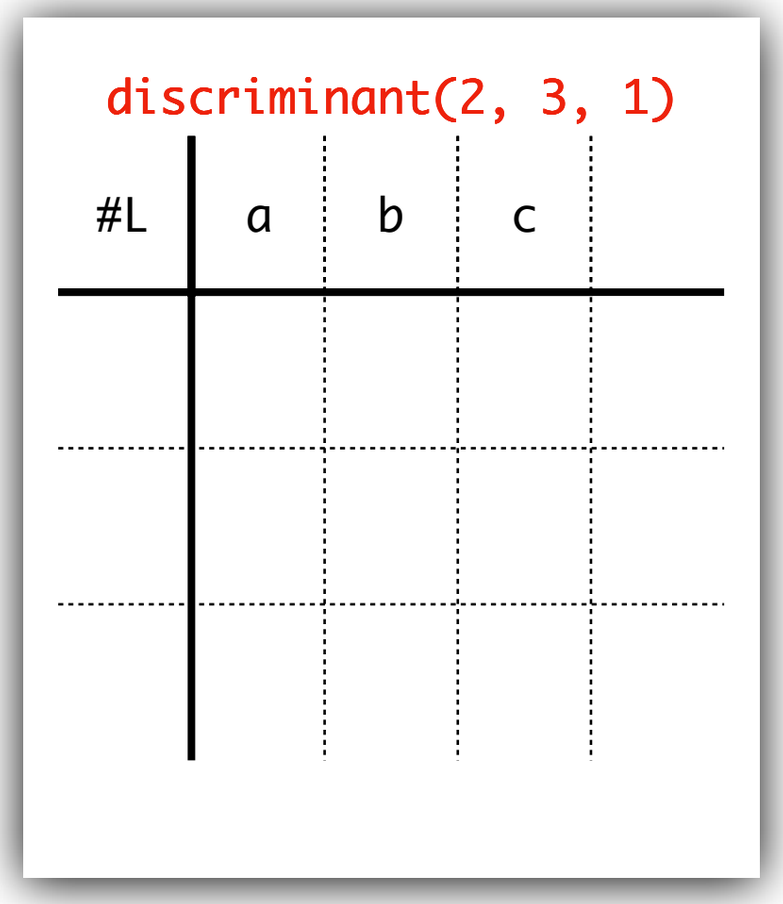


Discriminant de $2x^2 + 3x + 1$

```
1 def carré(x):
2     y = x * x
3     return y
```

```
4 def discriminant(a, b, c):
5     d = carré(b) - 4*a*c
6     return d
```

```
>>> discriminant(2, 3, 1)
```

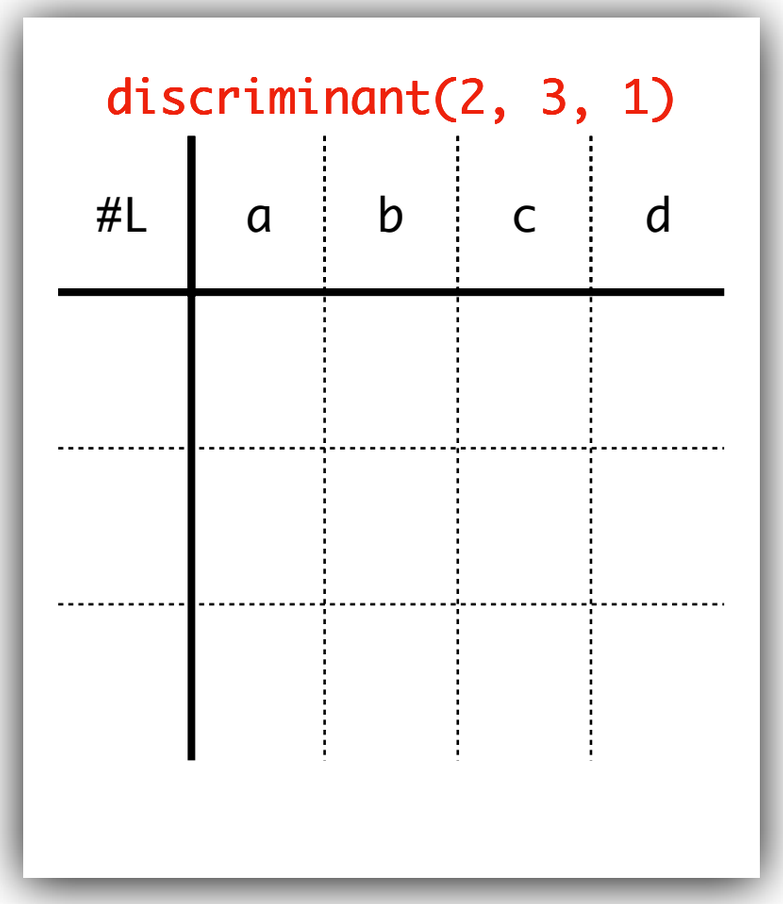


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

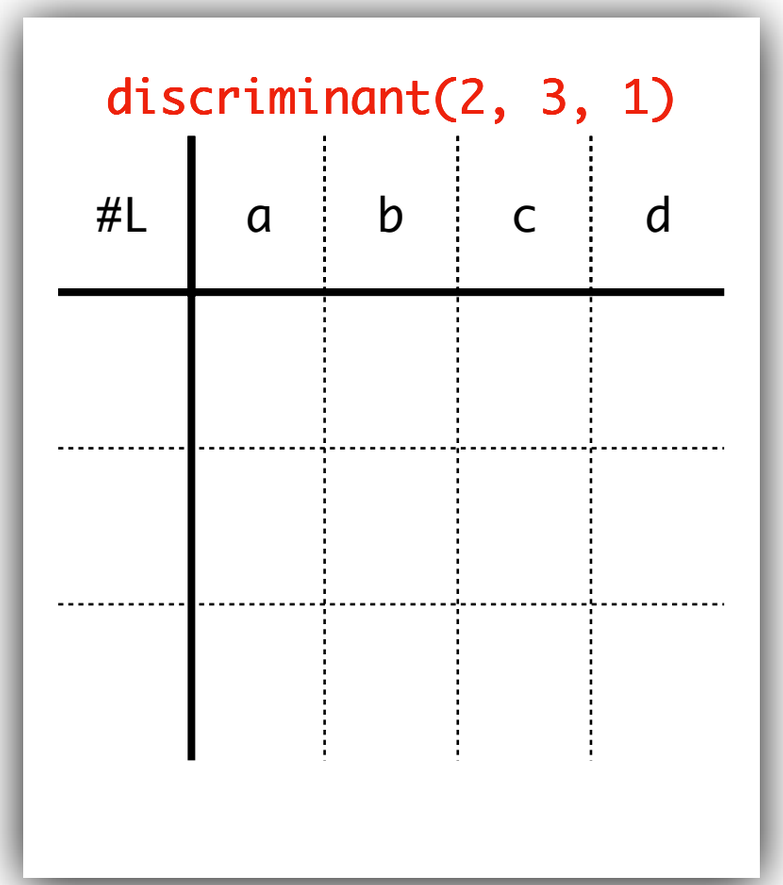


Discriminant de $2x^2 + 3x + 1$

```
1 | def carré(x):
2 |     y = x * x
3 |     return y
```

```
4 | def discriminant(a, b, c):
5 |     d = carré(b) - 4*a*c
6 |     return d
```

```
>>> discriminant(2, 3, 1)
```

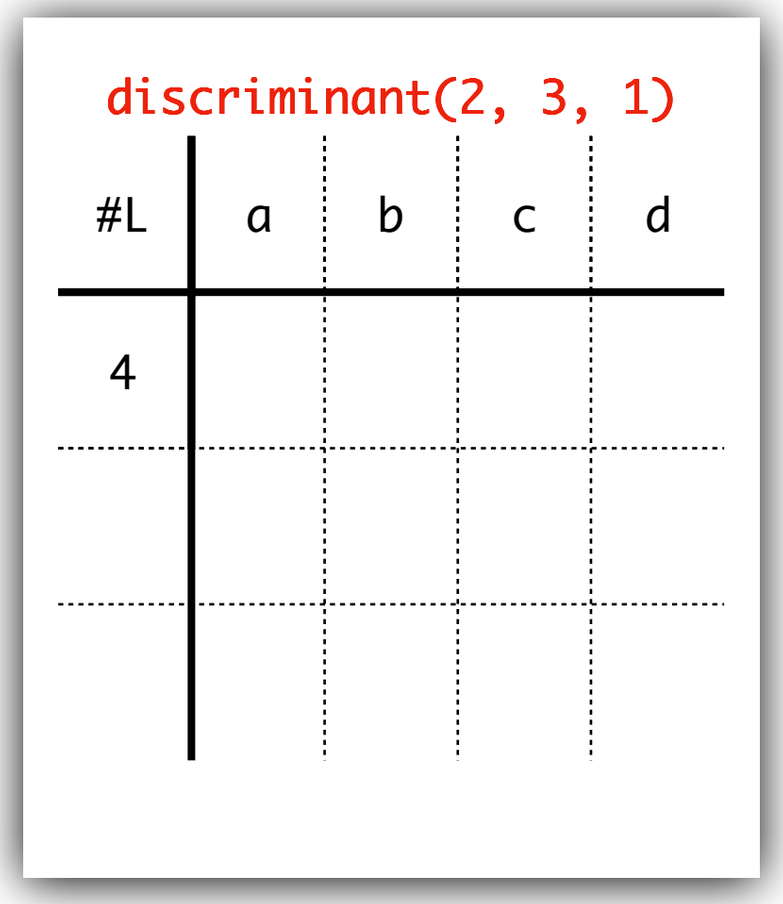


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```



Discriminant de $2x^2 + 3x + 1$

```
1 | def carré(x):
2 |     y = x * x
3 |     return y
```

```
4 | def discriminant(a, b, c):
5 |     d = carré(b) - 4*a*c
6 |     return d
```

```
>>> discriminant(2, 3, 1)
```

discriminant(2, 3, 1)

#L	a	b	c	d
4				

Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

discriminant(2, 3, 1)

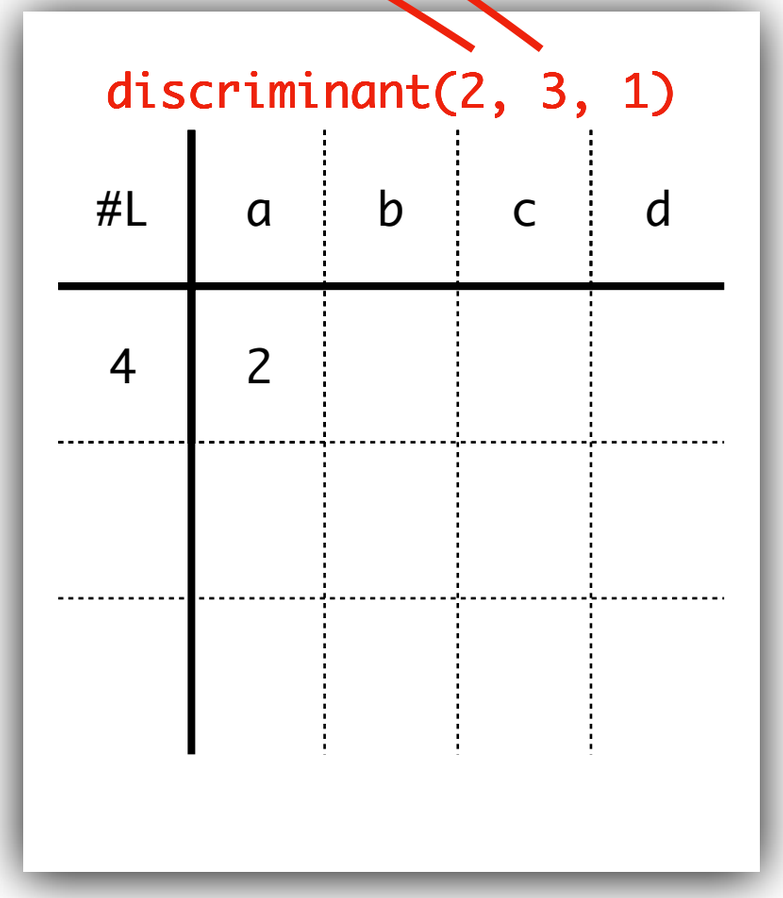
#L	a	b	c	d
4	2			

Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

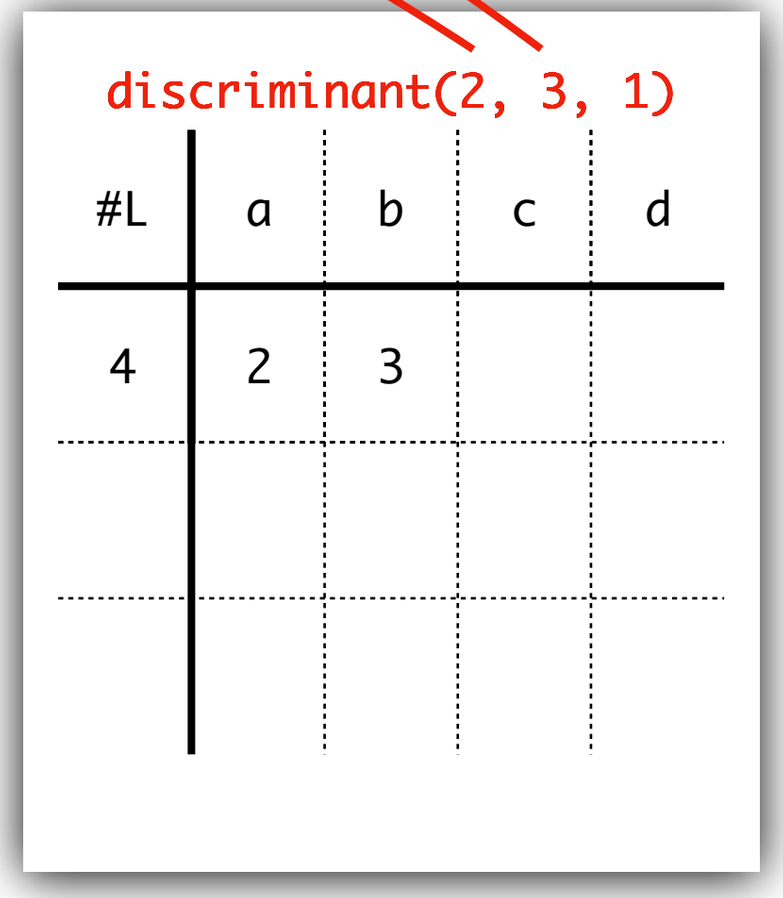


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

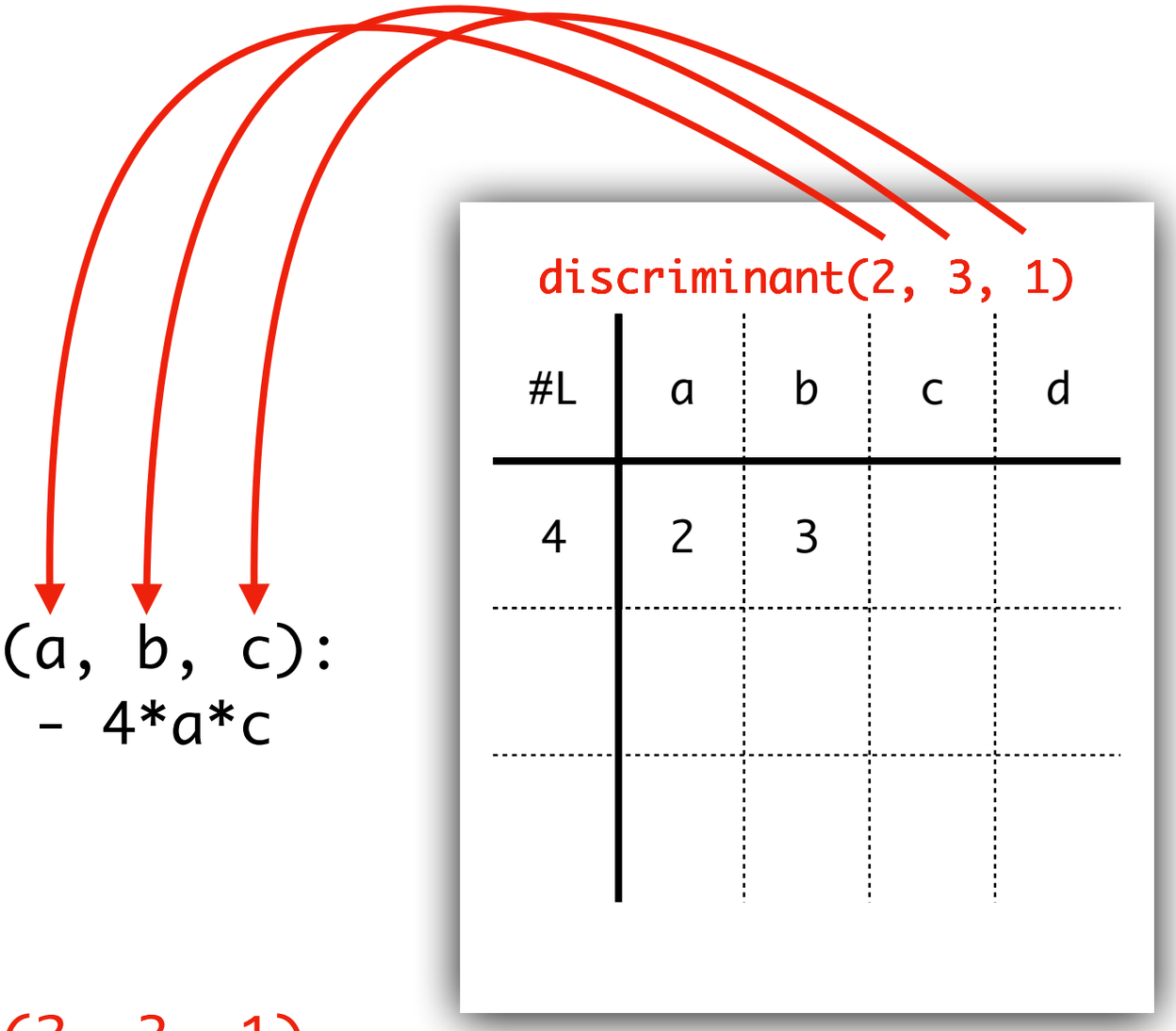


Discriminant de $2x^2 + 3x + 1$

```
1 | def carré(x):
2 |     y = x * x
3 |     return y
```

```
4 | def discriminant(a, b, c):
5 |     d = carré(b) - 4*a*c
6 |     return d
```

```
>>> discriminant(2, 3, 1)
```

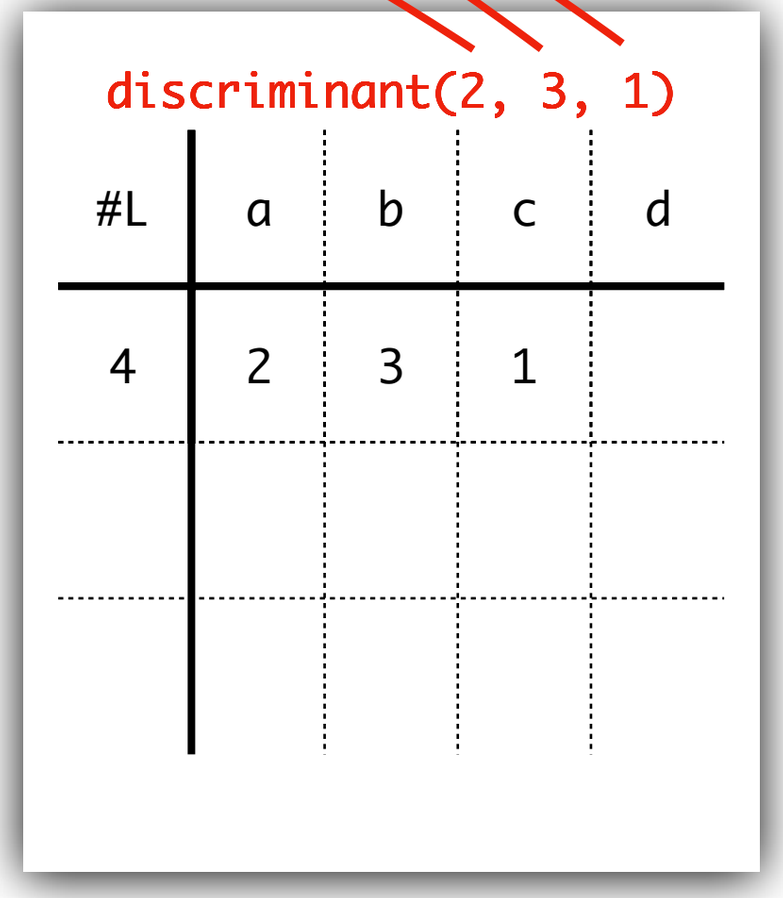


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

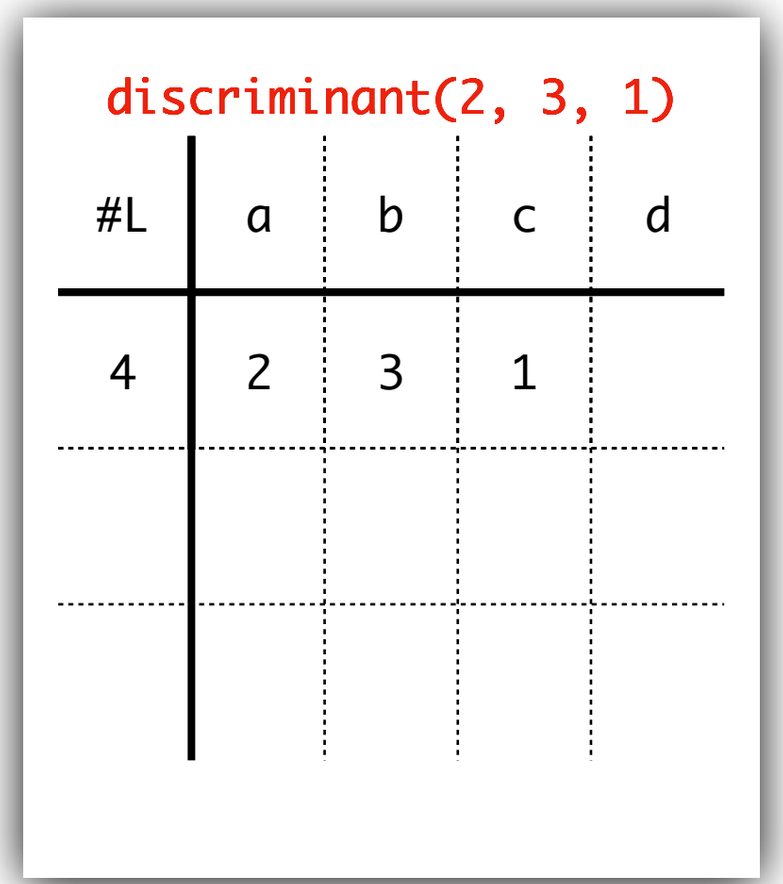


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```



Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

👉

```
>>> discriminant(2, 3, 1)
```

discriminant(2, 3, 1)

#L	a	b	c	d
4	2	3	1	
5				

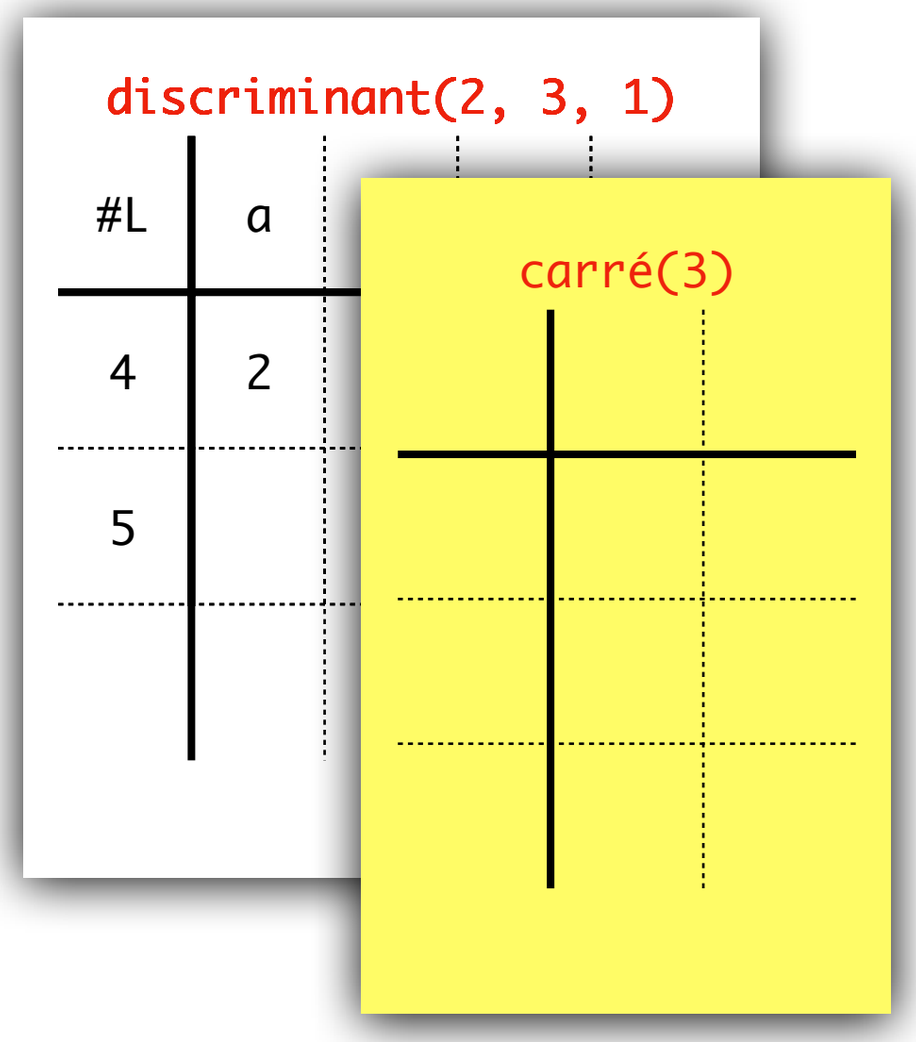
Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

👉

```
>>> discriminant(2, 3, 1)
```



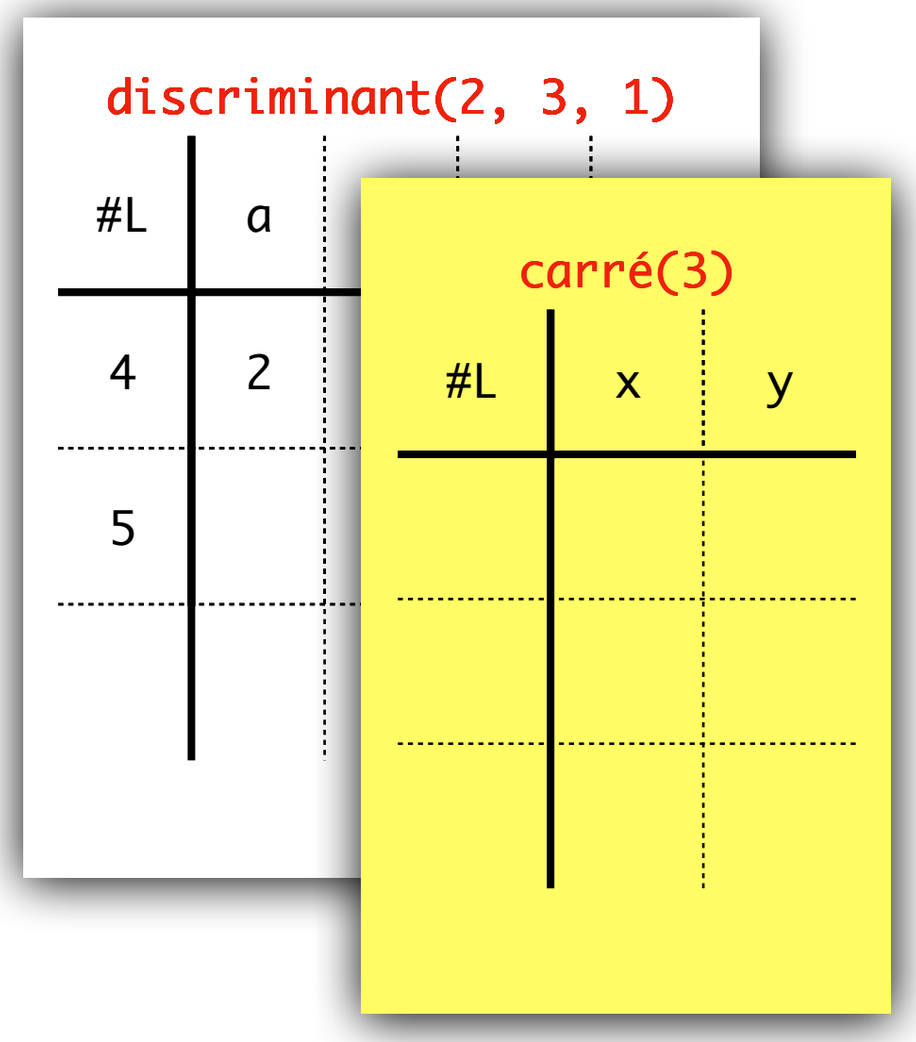
Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

👉

```
>>> discriminant(2, 3, 1)
```

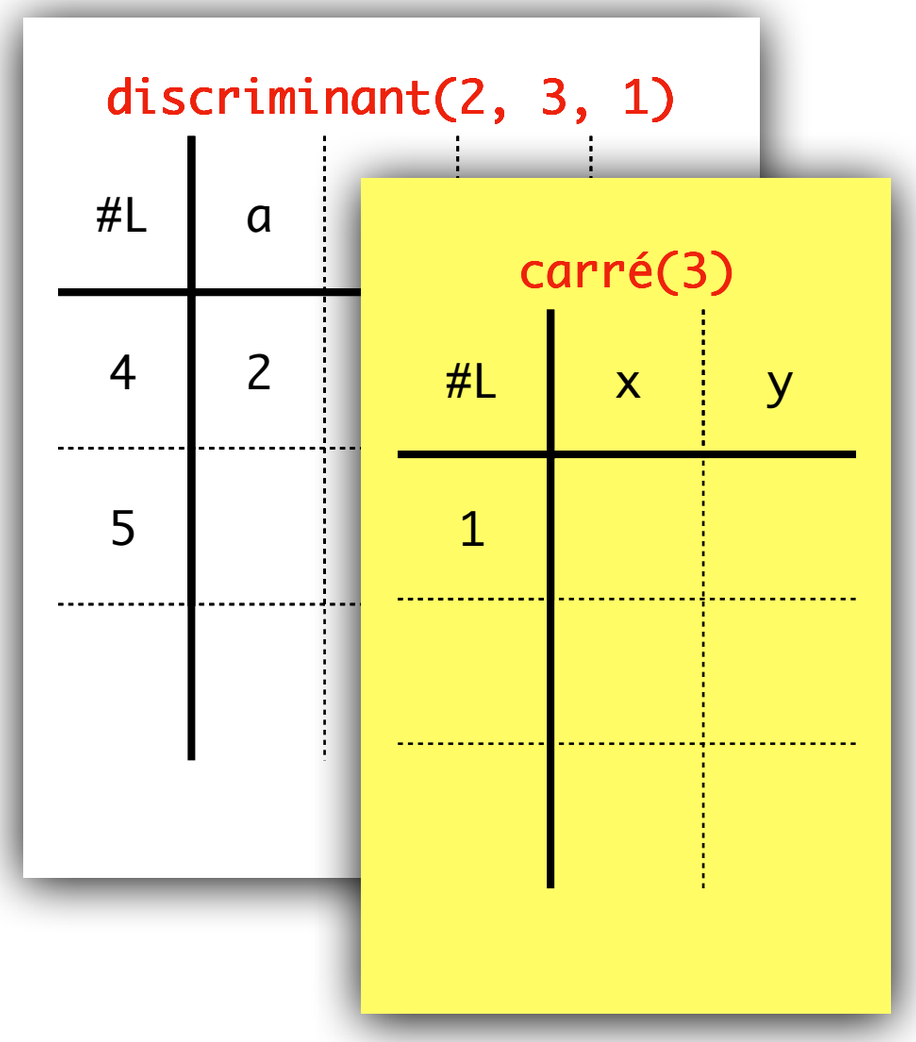


Discriminant de $2x^2 + 3x + 1$

👉
```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```



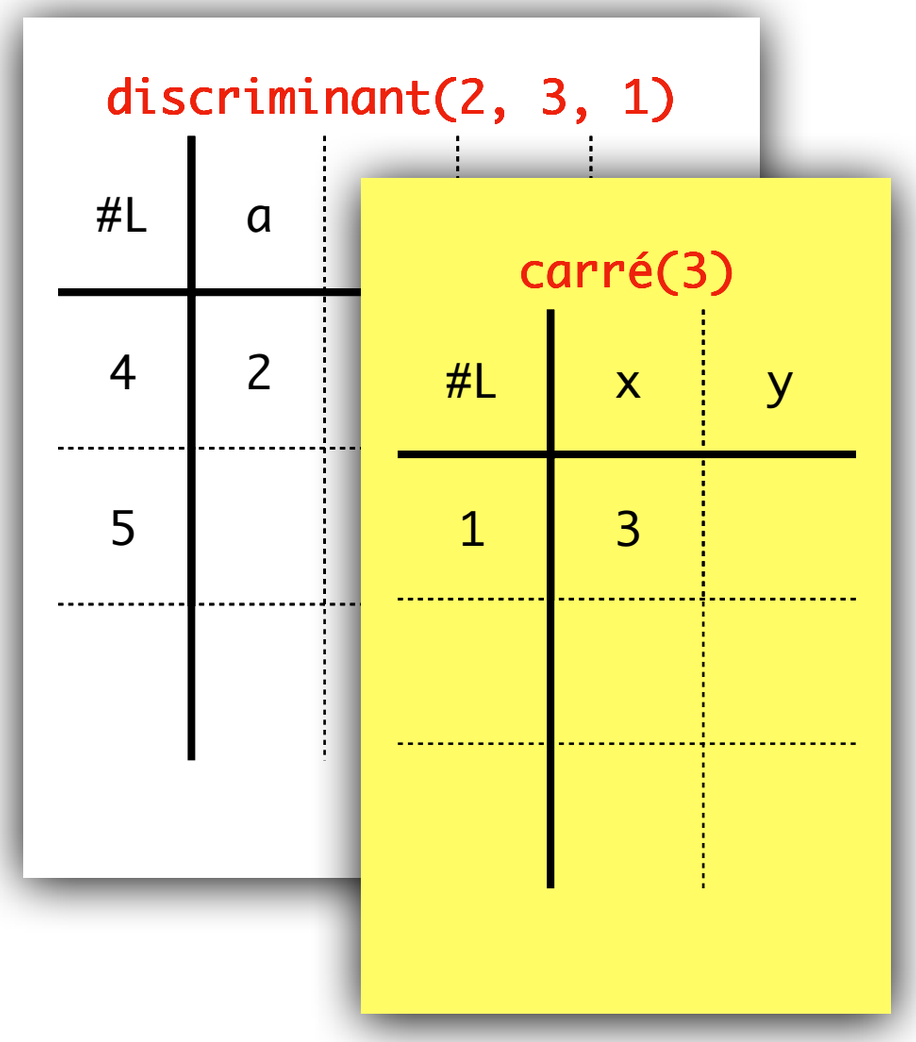
Discriminant de $2x^2 + 3x + 1$

👉

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

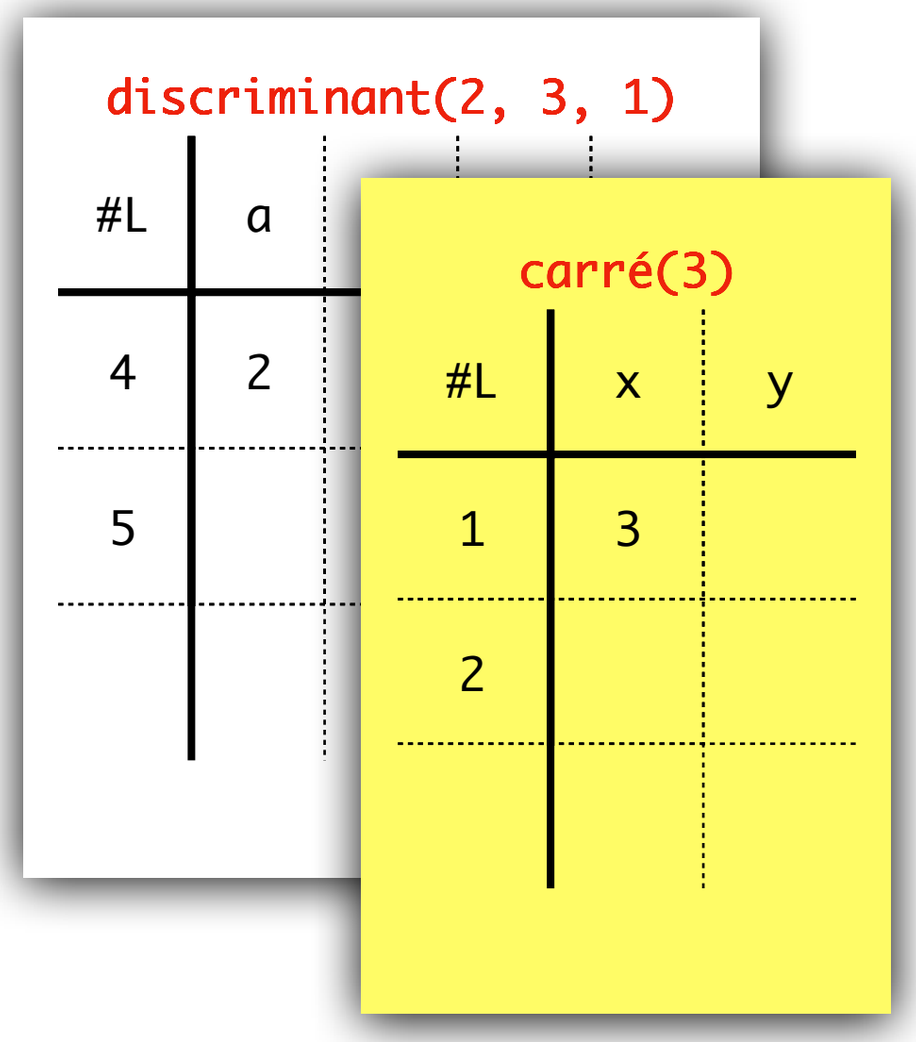


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

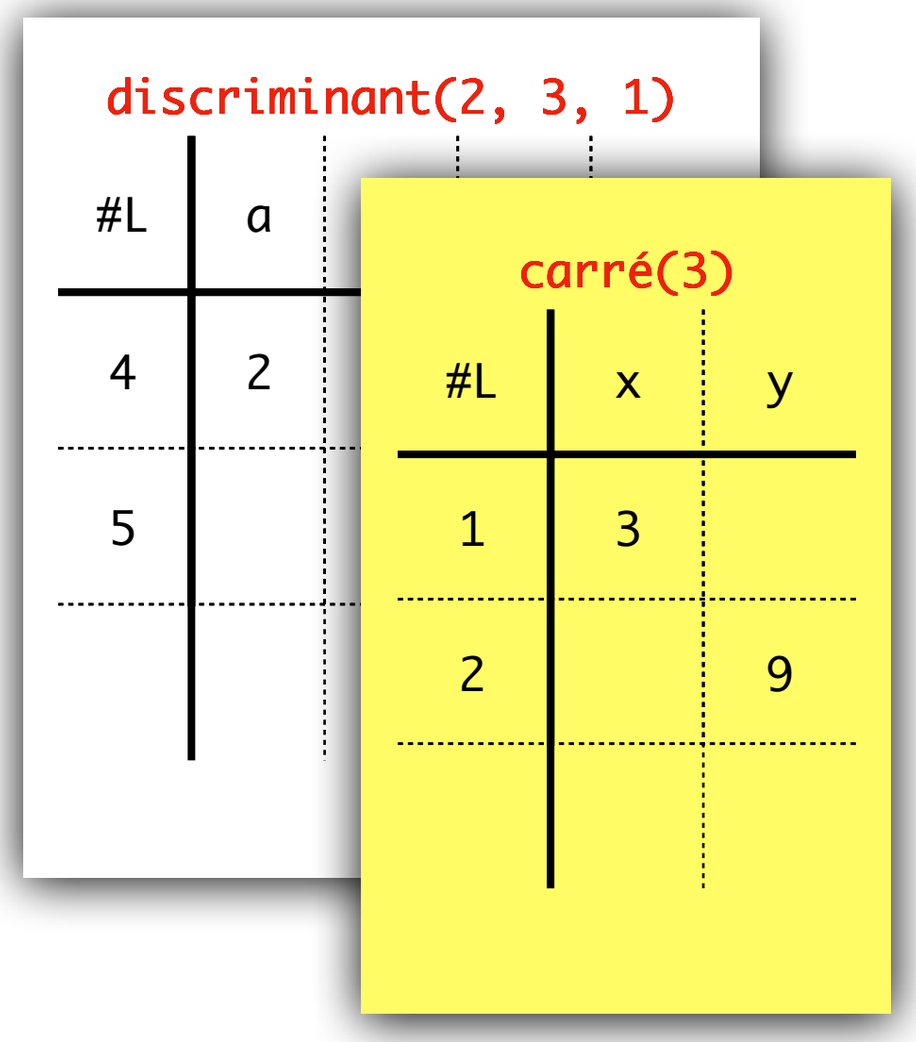


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

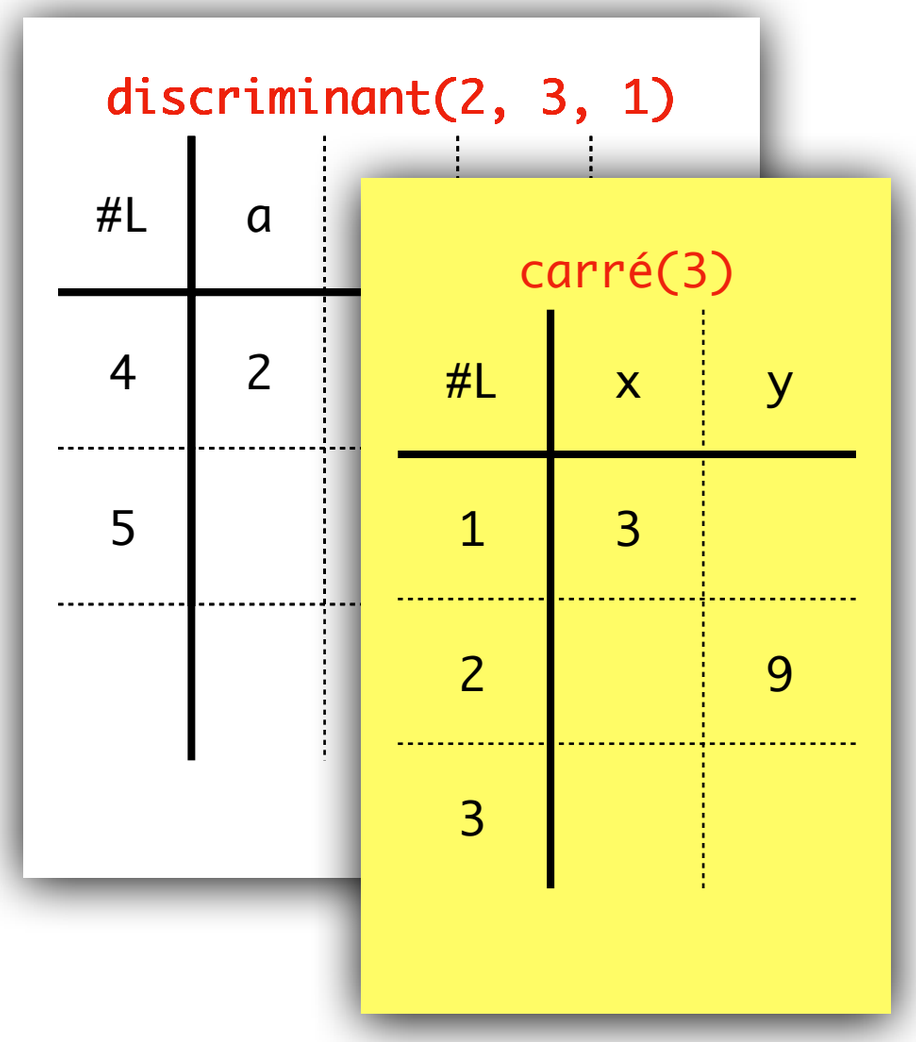


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```

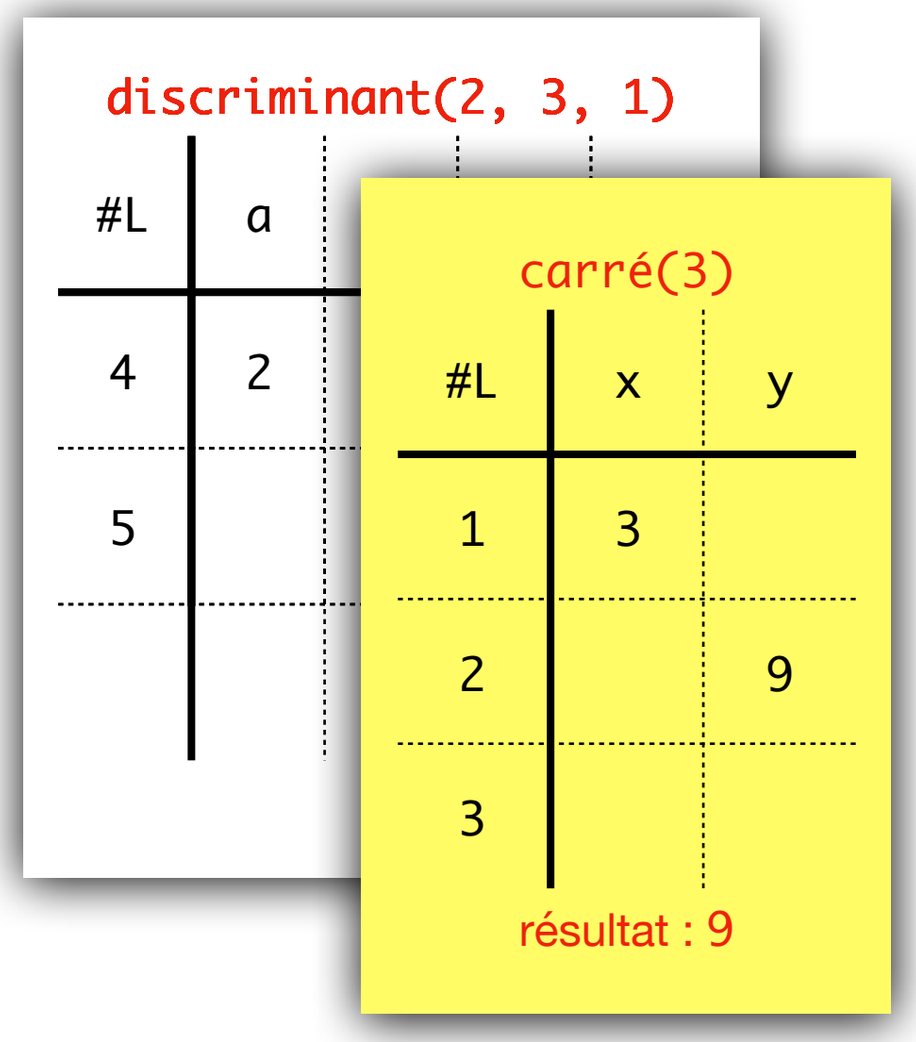


Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

```
>>> discriminant(2, 3, 1)
```



Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

👉

```
>>> discriminant(2, 3, 1)
```

discriminant(2, 3, 1)

#L	a	b	c	d
4	2	3	1	
5				

Discriminant de $2x^2 + 3x + 1$

```
1 | def carré(x):
2 |     y = x * x
3 |     return y
```

```
4 | def discriminant(a, b, c):
5 |     d = carré(b) - 4*a*c
6 |     return d
```

👉

```
>>> discriminant(2, 3, 1)
```

discriminant(2, 3, 1)

#L	a	b	c	d
4	2	3	1	
5				1

Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

👉

```
>>> discriminant(2, 3, 1)
```

discriminant(2, 3, 1)

#L	a	b	c	d
4	2	3	1	
5				1

Discriminant de $2x^2 + 3x + 1$

```
1  def carré(x):
2      y = x * x
3      return y
```

```
4  def discriminant(a, b, c):
5      d = carré(b) - 4*a*c
6      return d
```

👉

```
>>> discriminant(2, 3, 1)
```

discriminant(2, 3, 1)

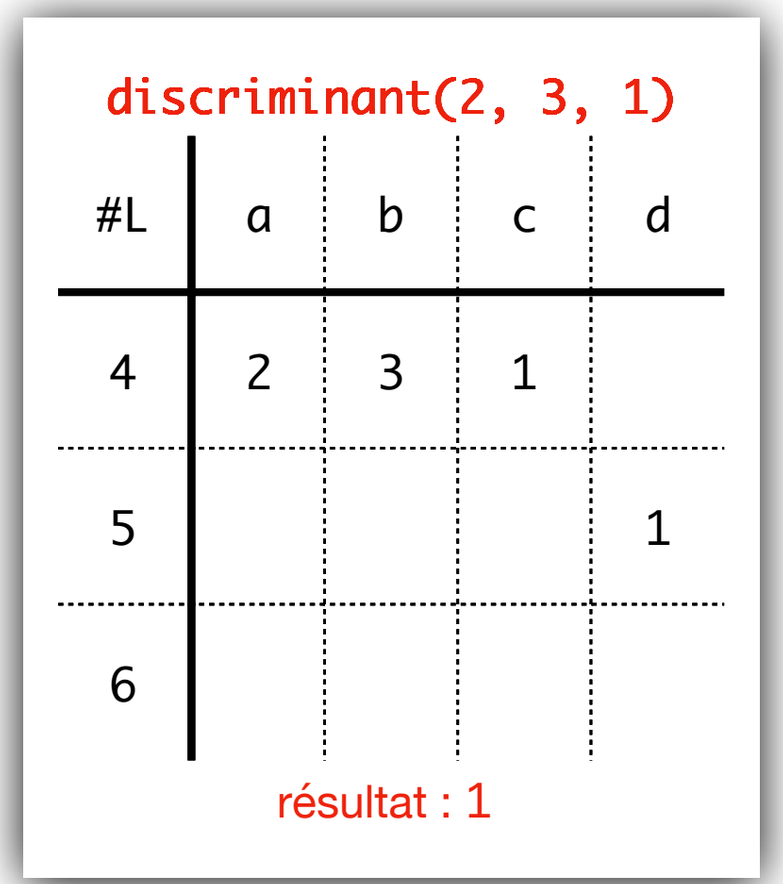
#L	a	b	c	d
4	2	3	1	
5				1
6				

Discriminant de $2x^2 + 3x + 1$

```
1 def carré(x):
2     y = x * x
3     return y
```

```
4 def discriminant(a, b, c):
5     d = carré(b) - 4*a*c
6     return d
```

```
>>> discriminant(2, 3, 1)
1
```



Évolution du calcul du discriminant de $2x^2 + 3x + 1$

discriminant(2, 3, 1)

#L	a	b	c	d
4	2	3	1	
5				1
6				

résultat : 1

carré(3)

#L	x	y
1	3	
2		9
3		

résultat : 9

Évolution du calcul du discriminant de $2x^2 + 3x + 1$

discriminant(2, 3, 1)

#L	a	b	c	d
4	2	3	1	
5				1
6				

résultat : 1

carré(3)

#L	x	y
1	3	
2		9
3		

résultat : 9

3 + 3 = 6 op

Exercice 3 du TD2

Conditions

Valeur absolue

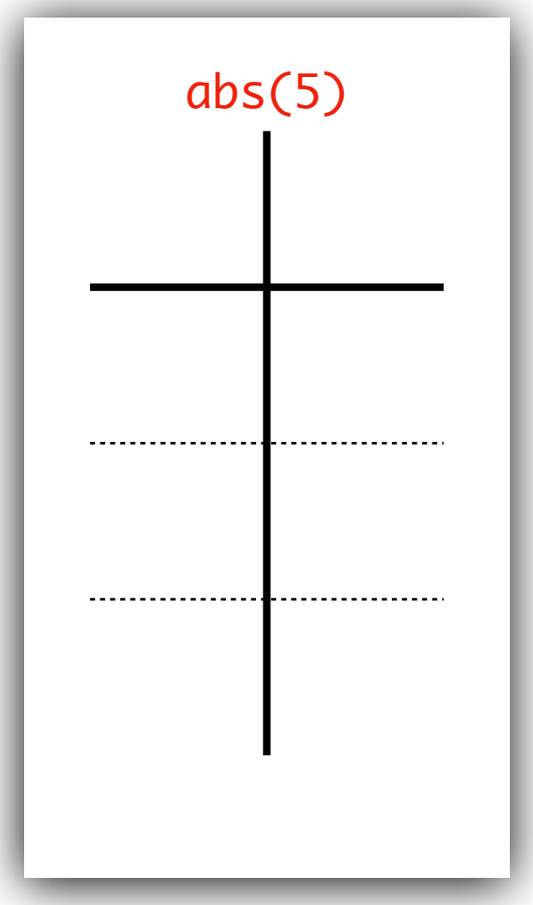
```
def abs(x):
    if x >= 0:
        return x
    else:
        return -x
```

Valeur absolue de 5

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```

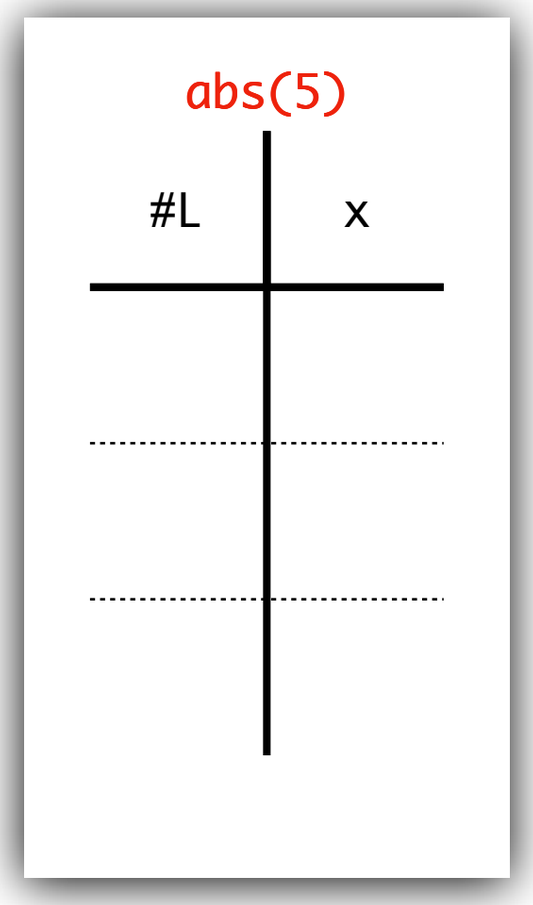
Valeur absolue de 5

```
1 def abs(x):
2     if x >= 0:
3         return x
4     else:
5         return -x
```



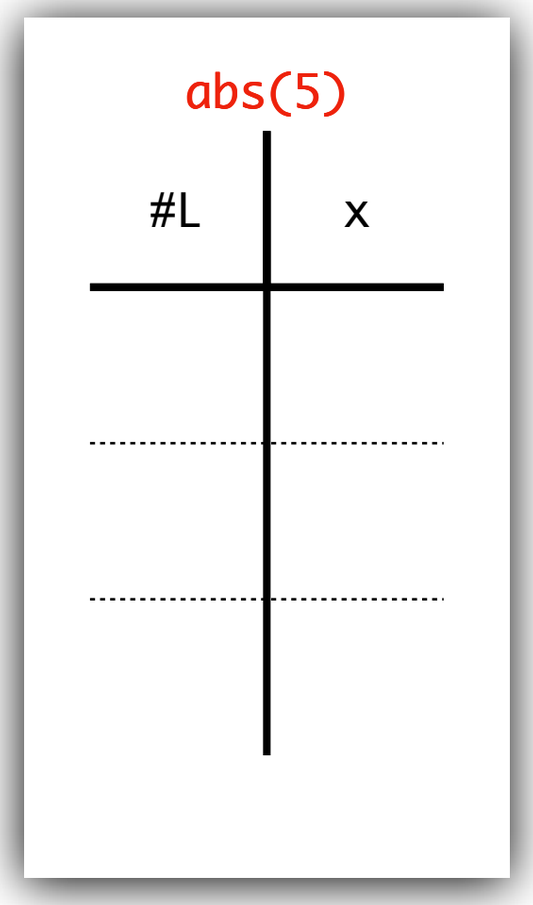
Valeur absolue de 5

```
1 def abs(x):
2     if x >= 0:
3         return x
4     else:
5         return -x
```



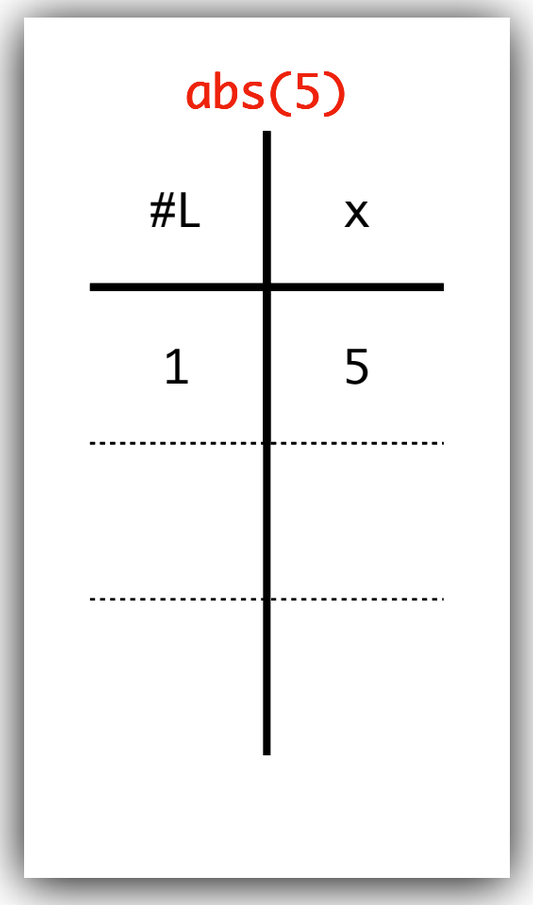
Valeur absolue de 5

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



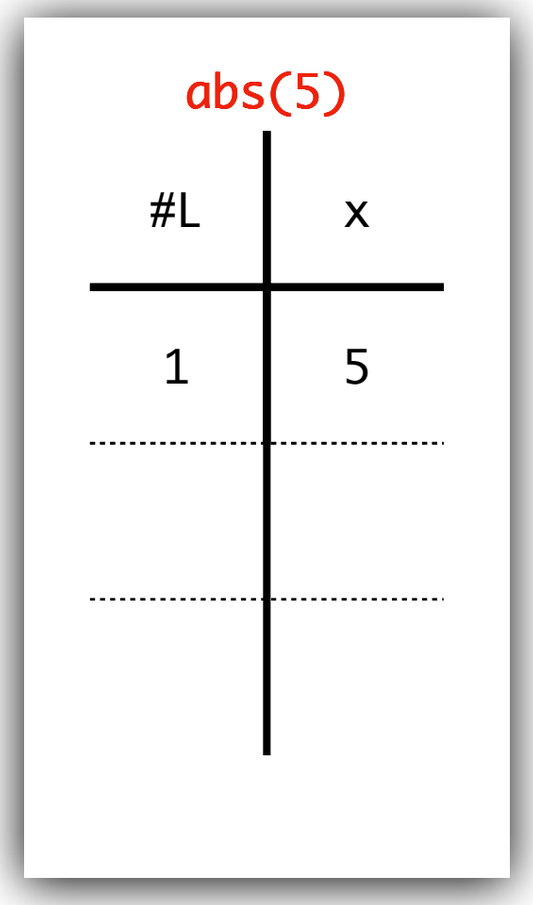
Valeur absolue de 5

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



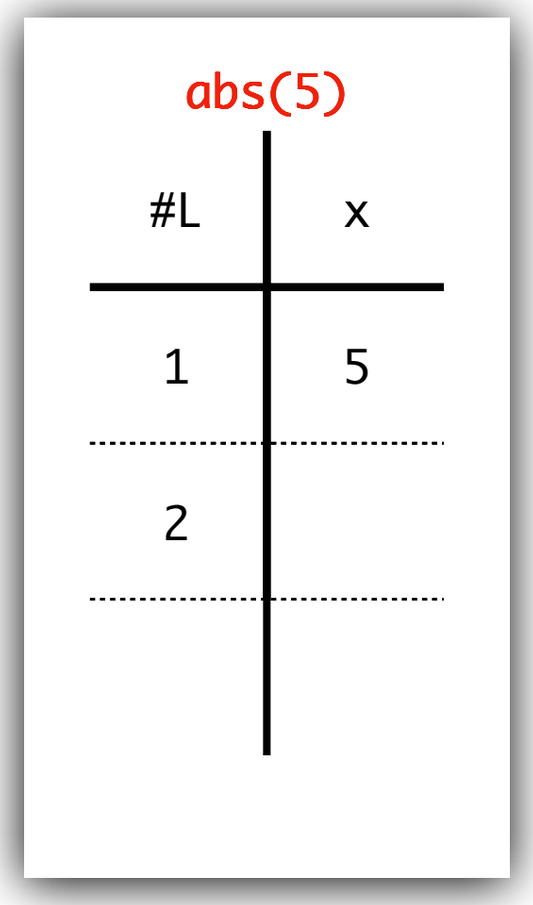
Valeur absolue de 5

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



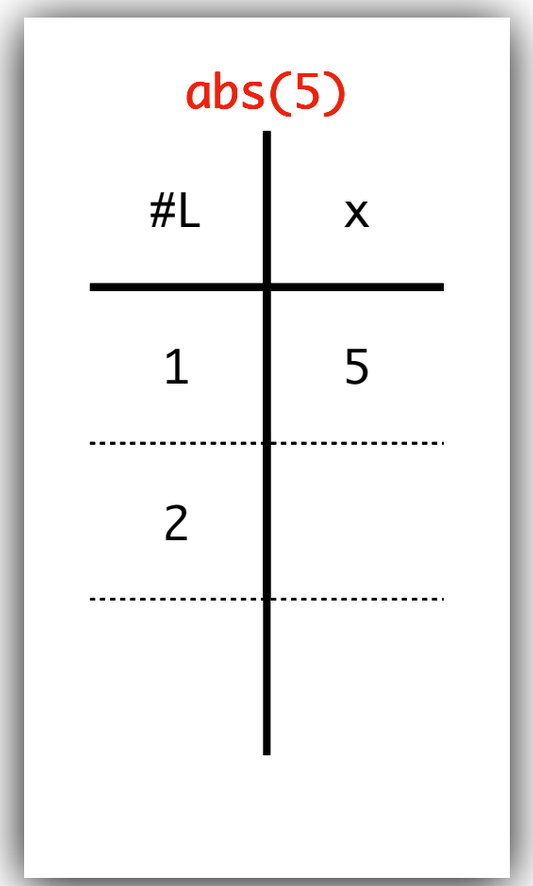
Valeur absolue de 5

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



Valeur absolue de 5

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



Valeur absolue de 5

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```

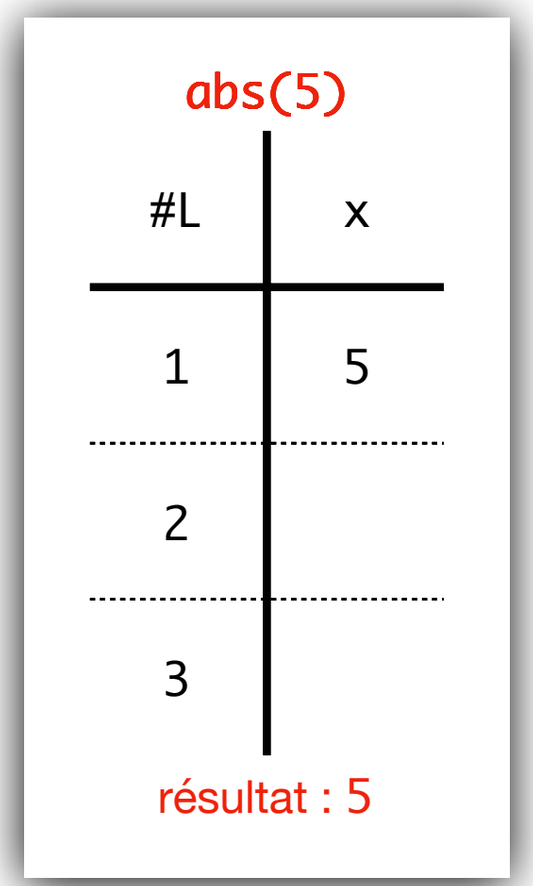
👉

abs(5)

#L	x
1	5
2	
3	

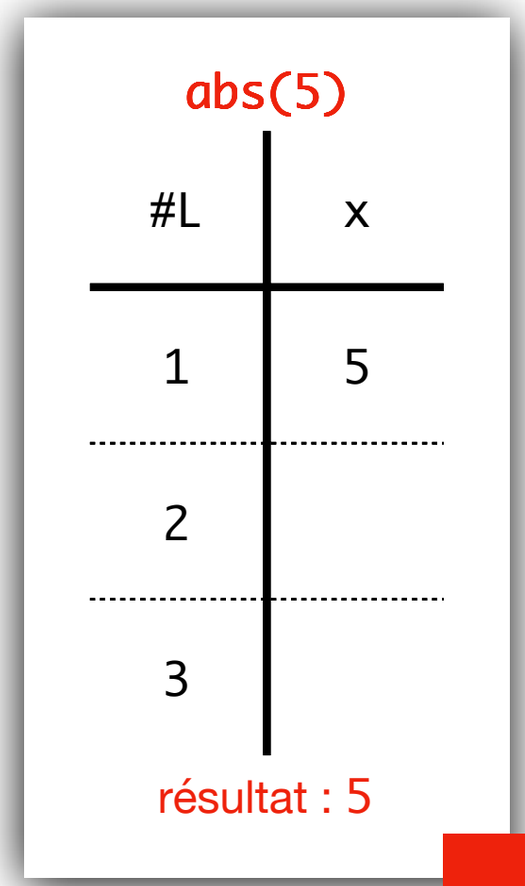
Valeur absolue de 5

```
1 | def abs(x):
2 |     if x >= 0:
3 |         return x
4 |     else:
5 |         return -x
```



Valeur absolue de 5

```
1 def abs(x):
2     if x >= 0:
3         return x
4     else:
5         return -x
```



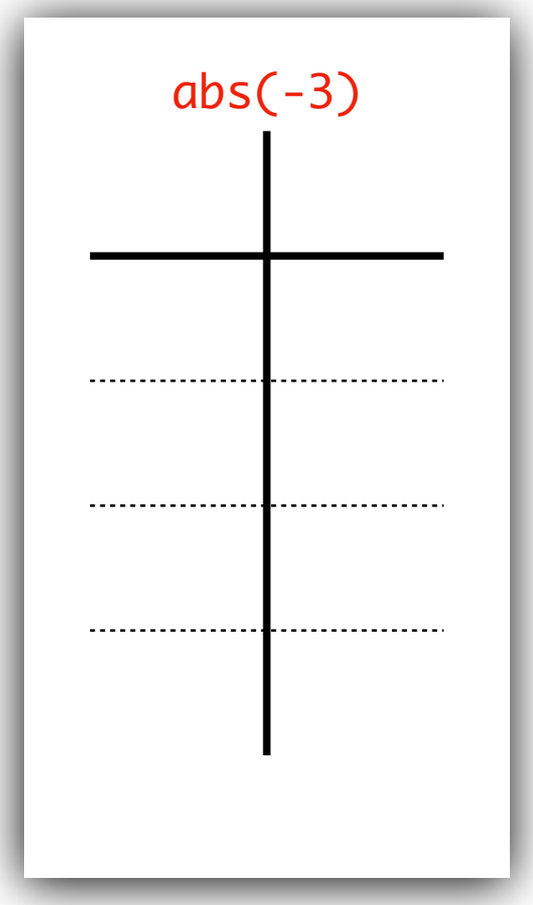
3 op

Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```

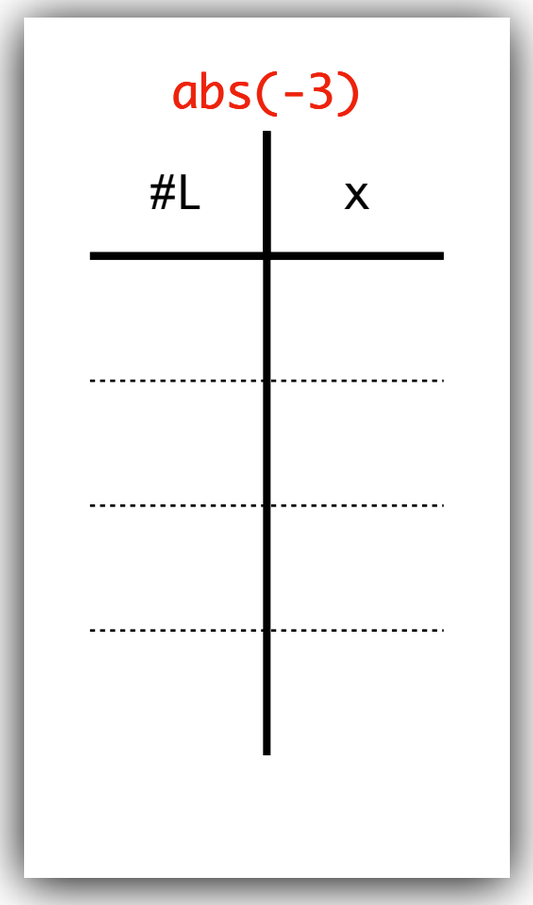
Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



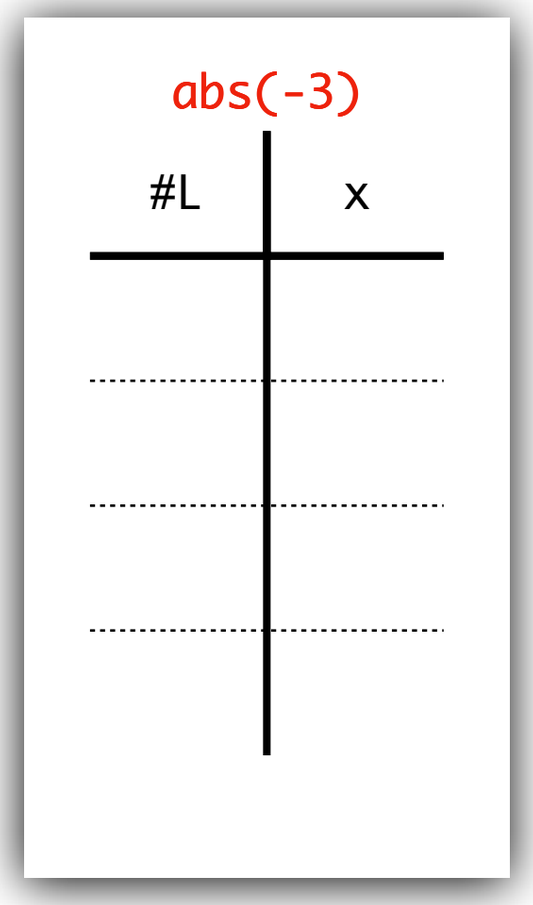
Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



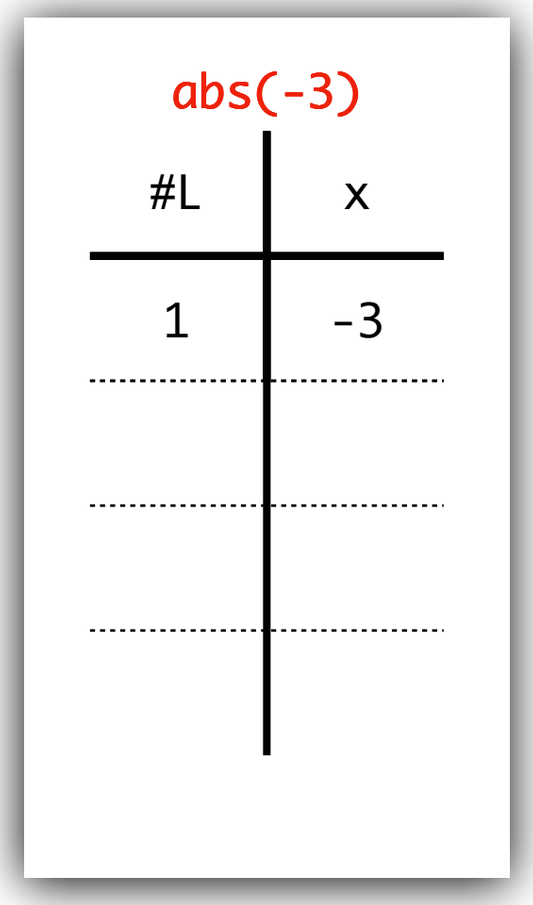
Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



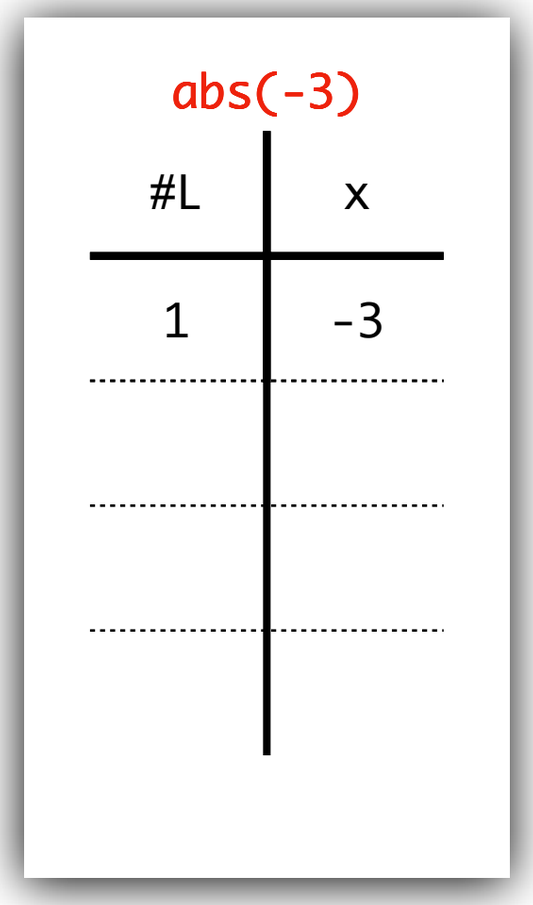
Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



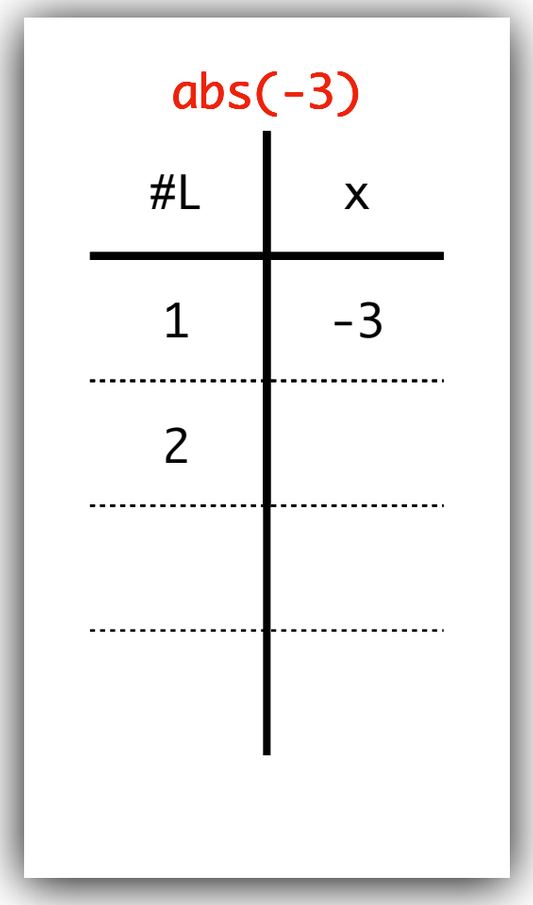
Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



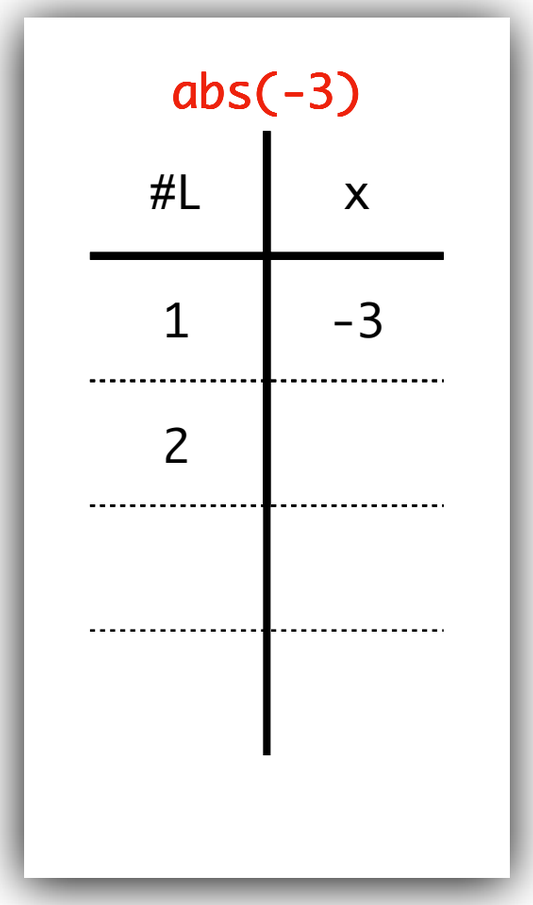
Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



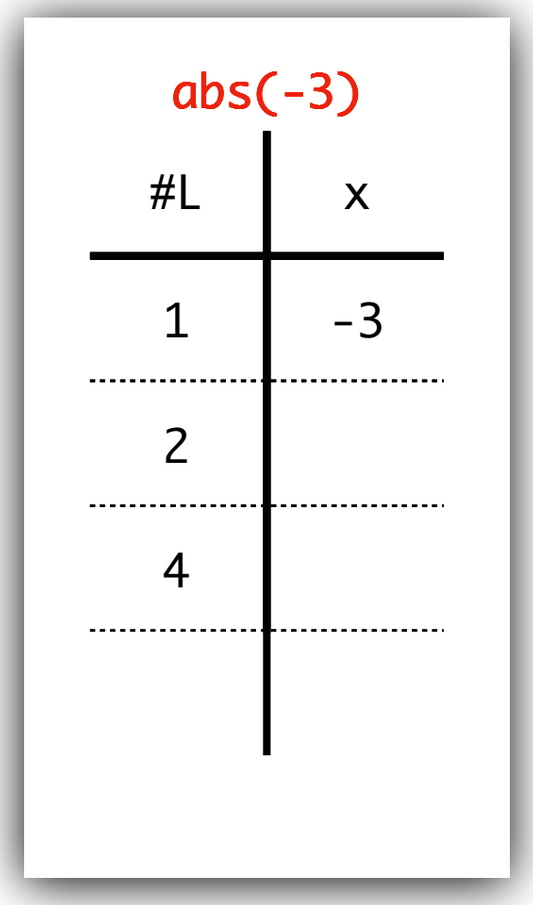
Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```

👉

abs(-3)

#L	x
1	-3
2	
4	

Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```

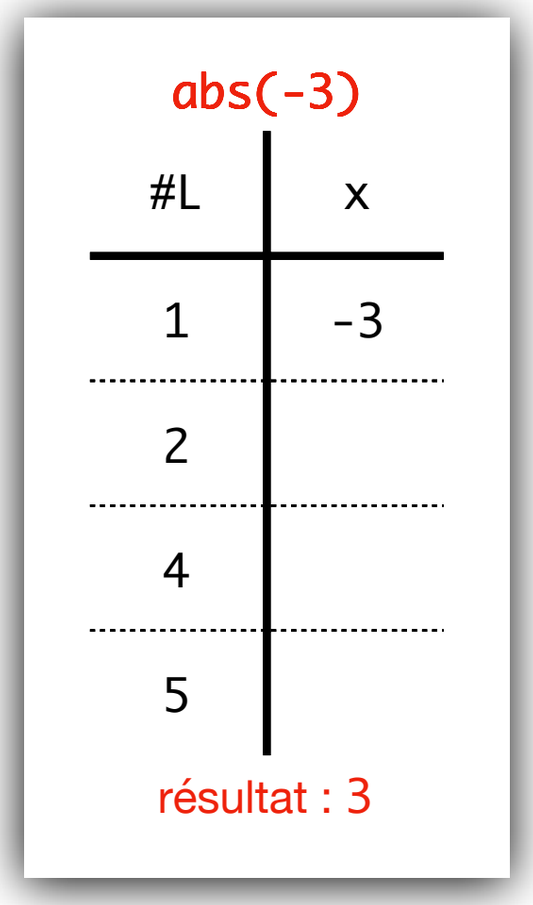
👉

abs(-3)

#L	x
1	-3
2	
4	
5	

Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```



Valeur absolue de –3

```
1  def abs(x):
2      if x >= 0:
3          return x
4      else:
5          return -x
```

abs(-3)

#L	x
1	-3
2	
4	
5	

résultat : 3

4 op

Exercice 4 du TD2

Valeur absolue (sans `else`)

```
def abs2(x):
    if x < 0:
        x = -x
    return x
```

Exercice 5 du TD2

Fonction signe

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{si } x < 0 \\ 0 & \text{si } x = 0 \\ 1 & \text{sinon} \end{cases}$$

Fonction signe

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{si } x < 0 \\ 0 & \text{si } x = 0 \\ 1 & \text{sinon} \end{cases}$$

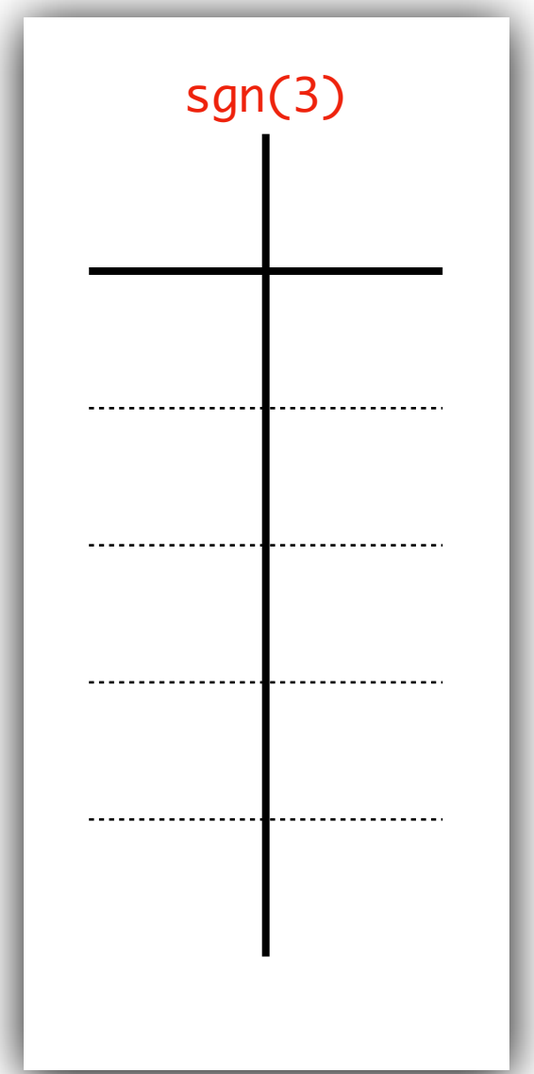
```
def sgn(x):
    if x < 0:
        return -1
    elif x == 0:
        return 0
    else:
        return 1
```

Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```

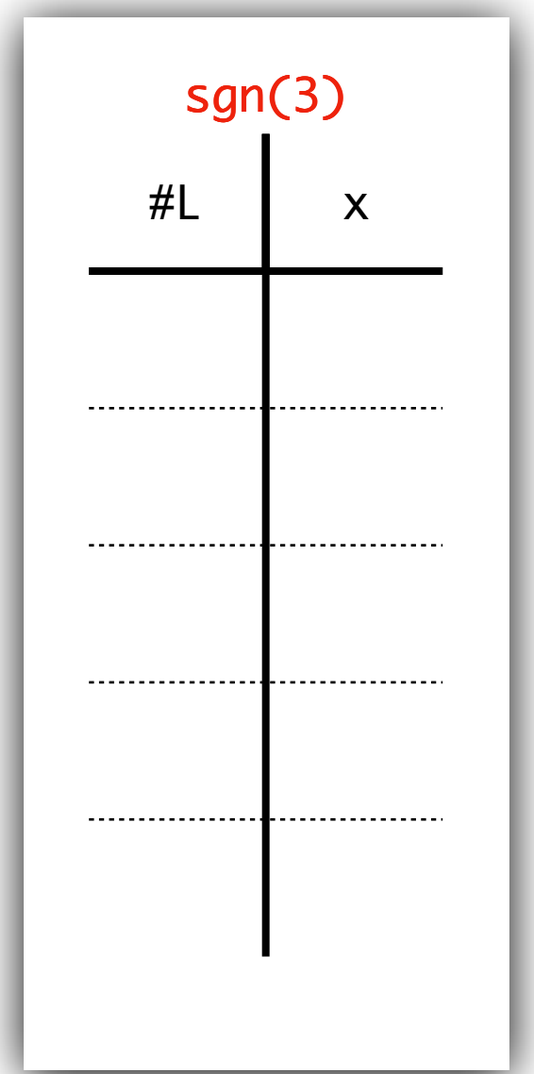
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



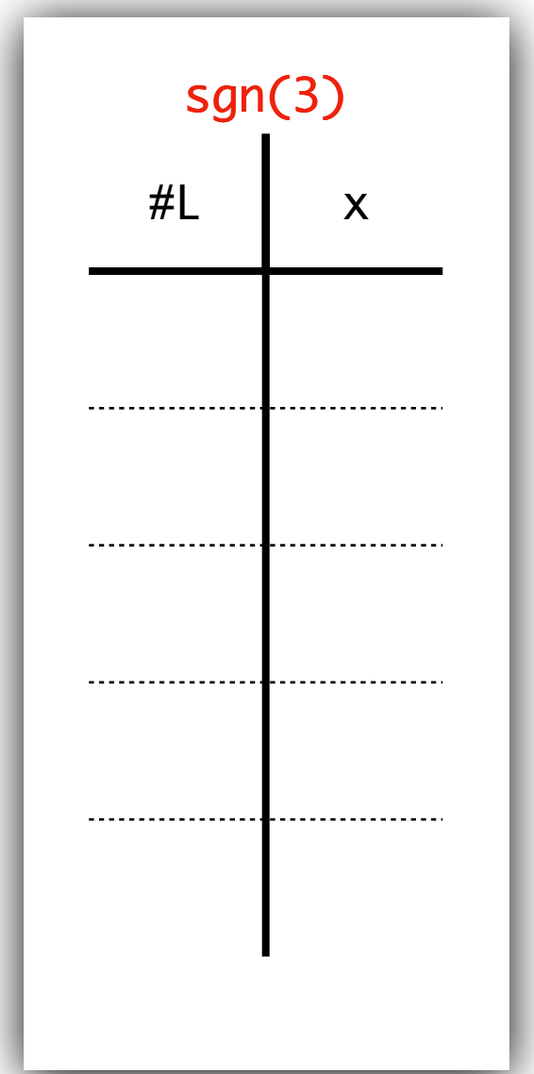
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



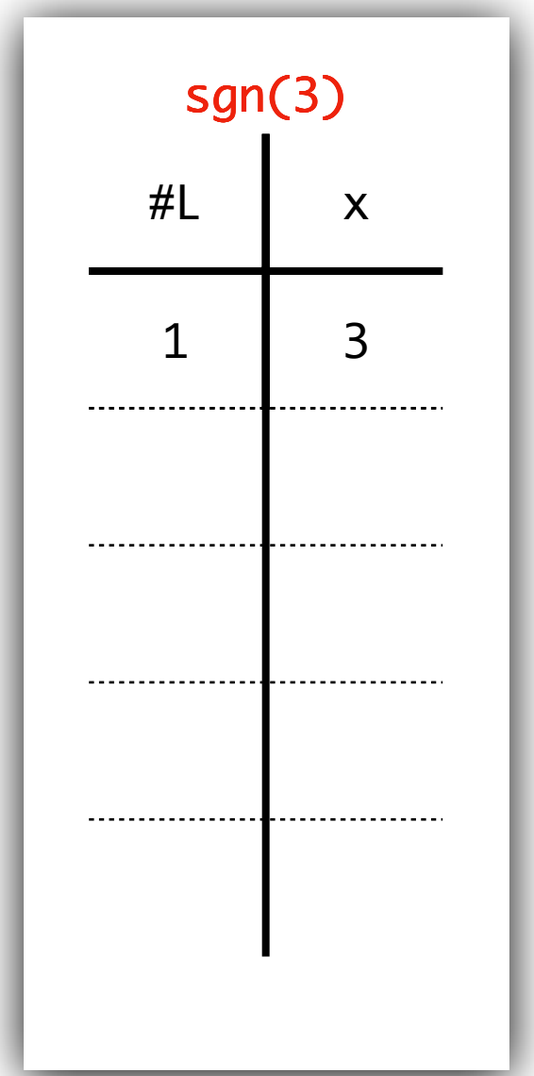
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



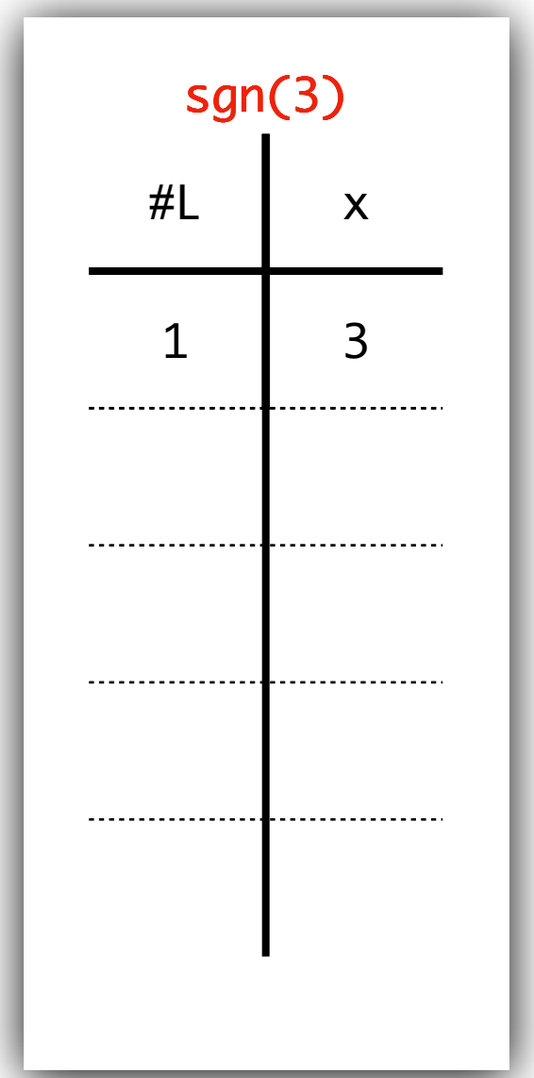
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



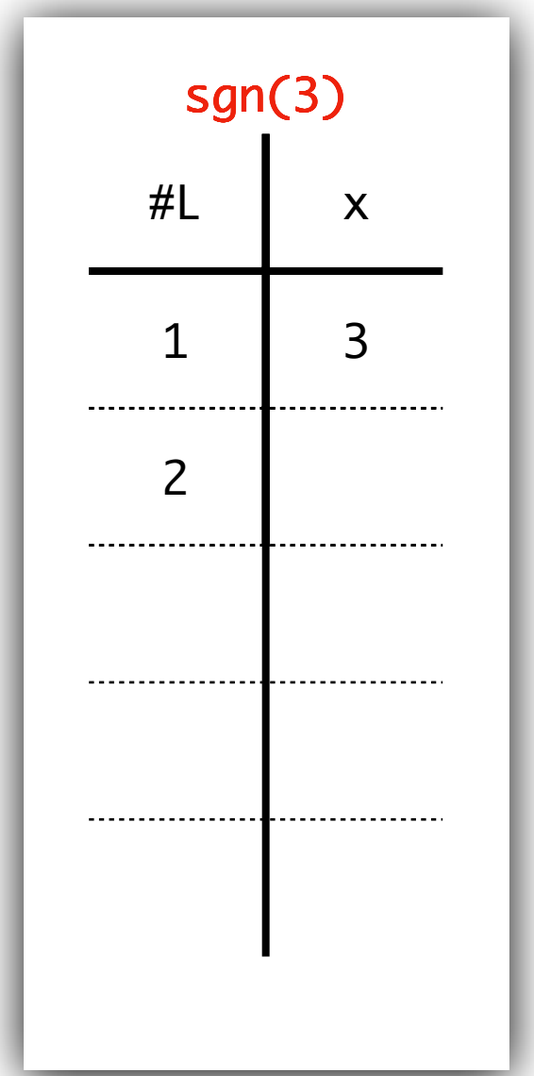
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



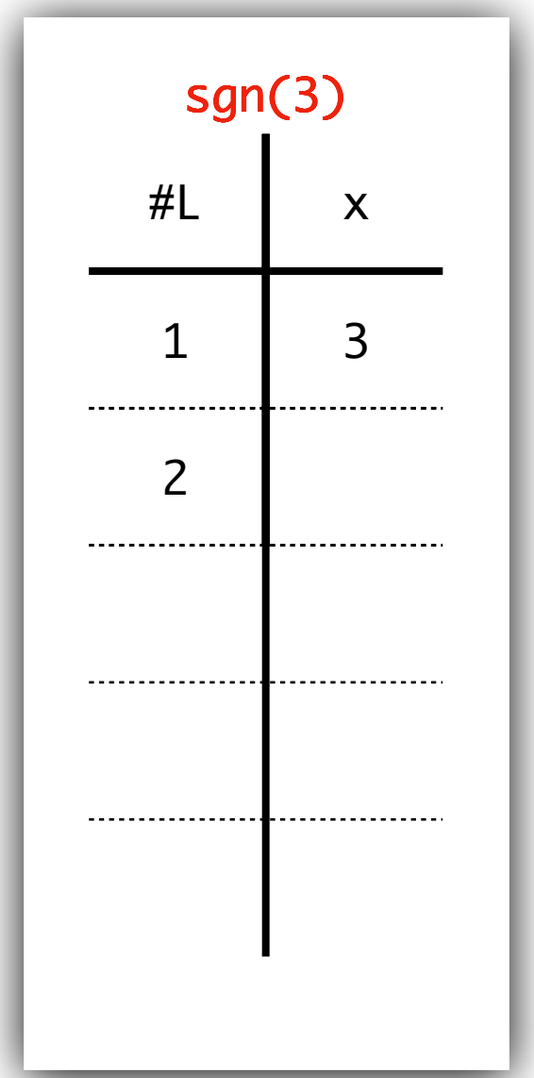
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



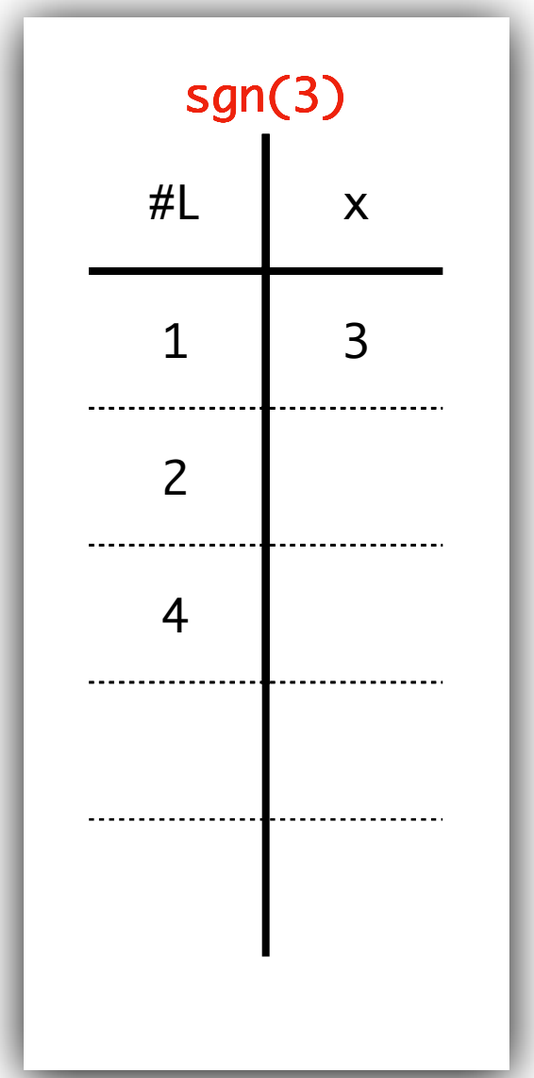
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



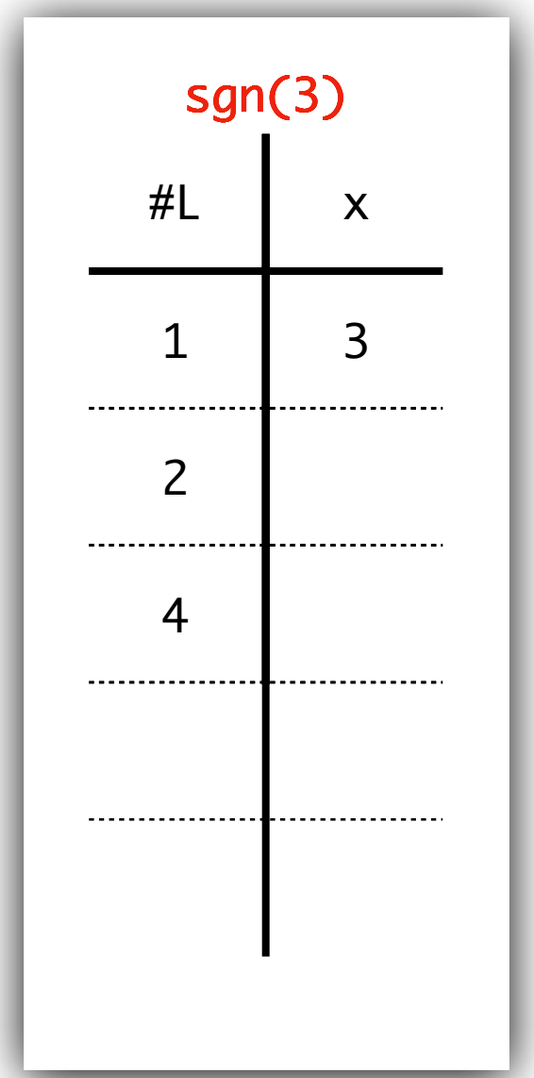
Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



Signe de 3

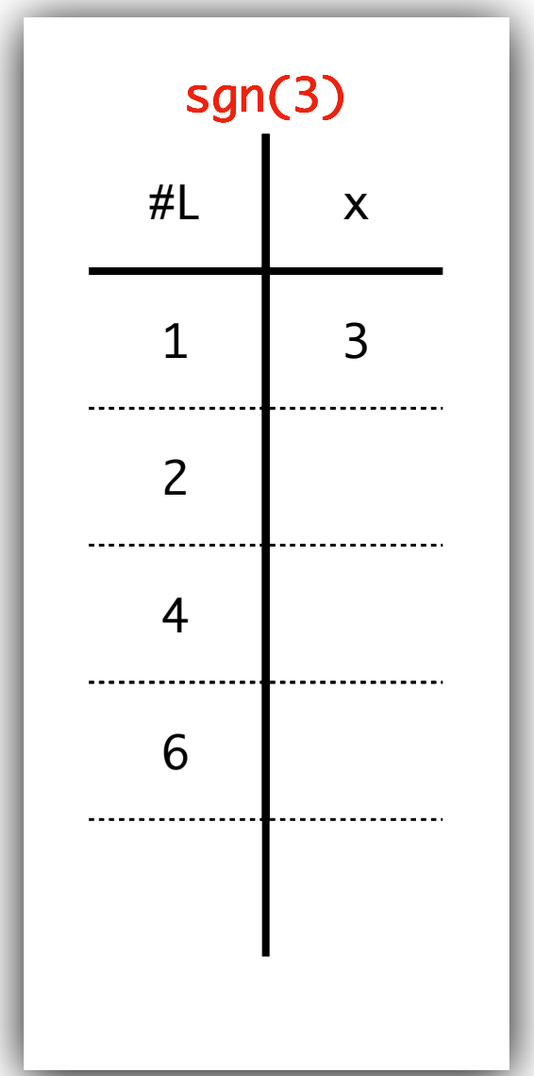
```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```

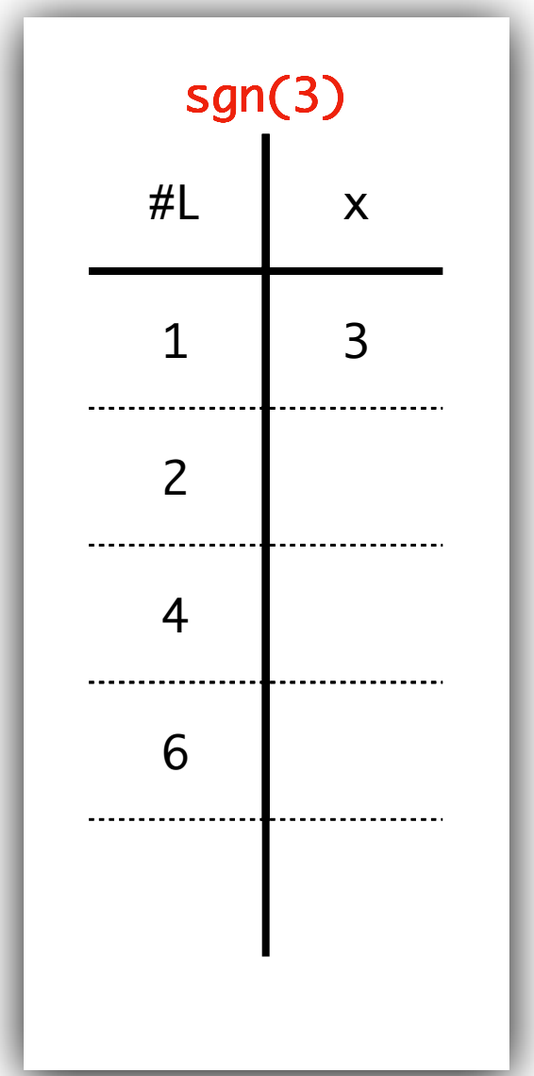
👉 6



Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```

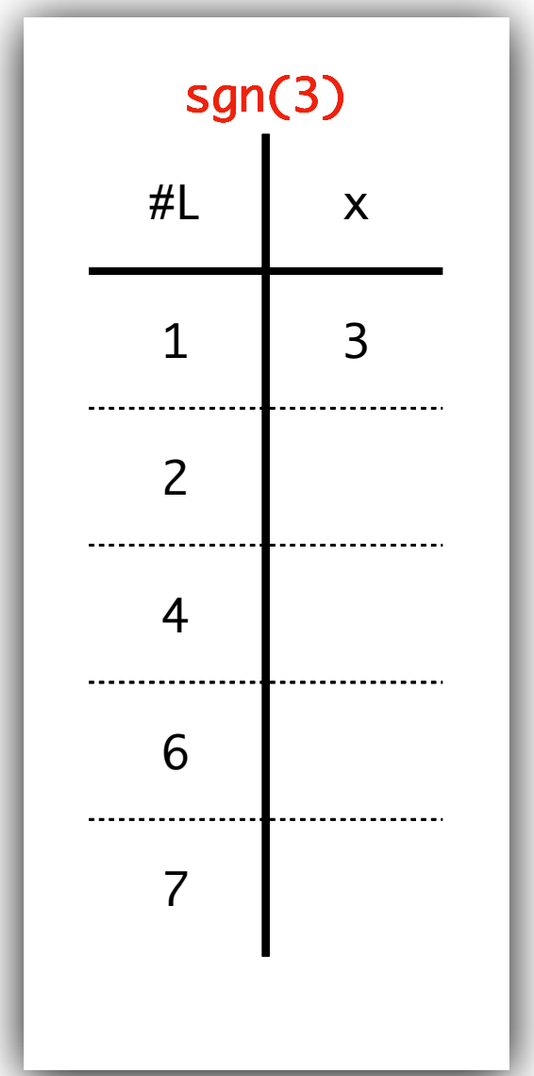
👉



Signe de 3

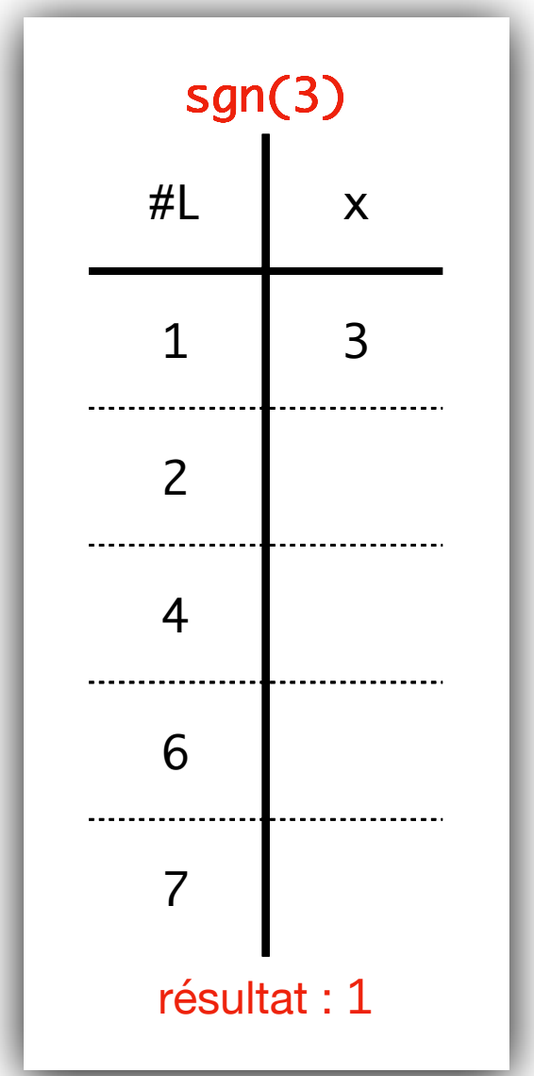
```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```

👉



Signe de 3

```
1  def sgn(x):
2      if x < 0:
3          return -1
4      elif x == 0:
5          return 0
6      else:
7          return 1
```



Exercice 6 du TD2

Itération

Somme de 1 à n

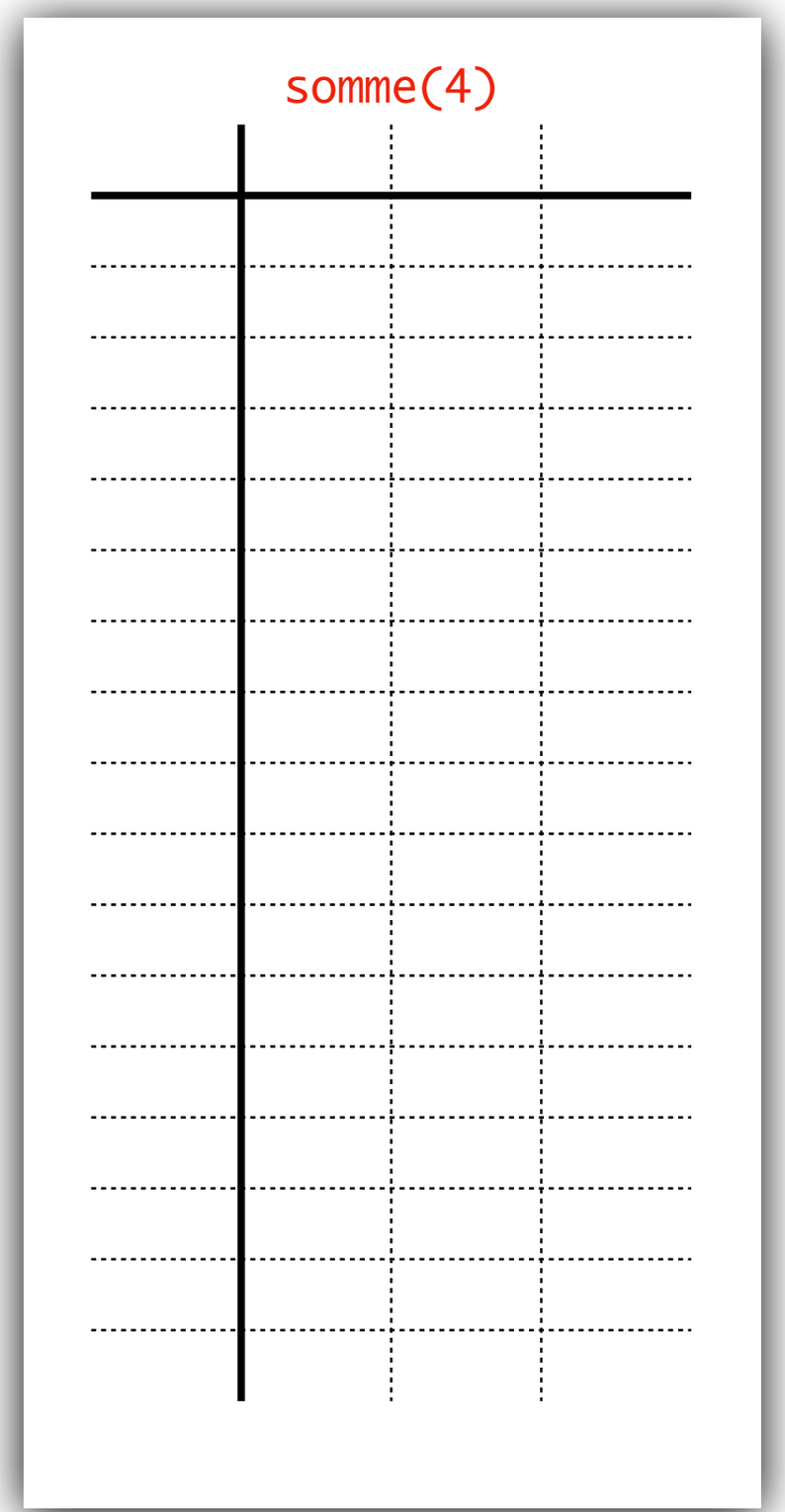
```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

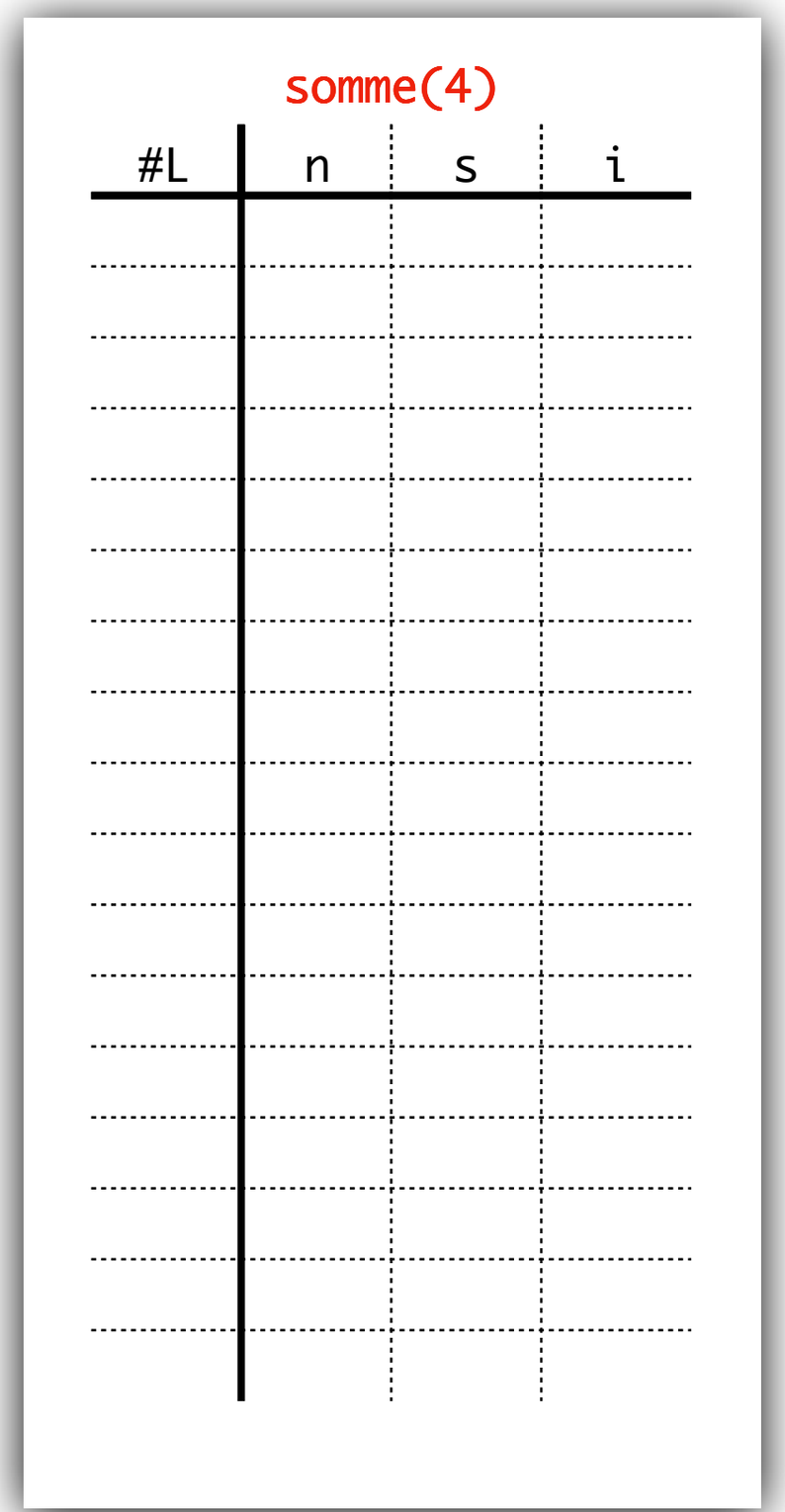
Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```



Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```



Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

somme(4)

#L	n	s	i

Somme de 1 à 4

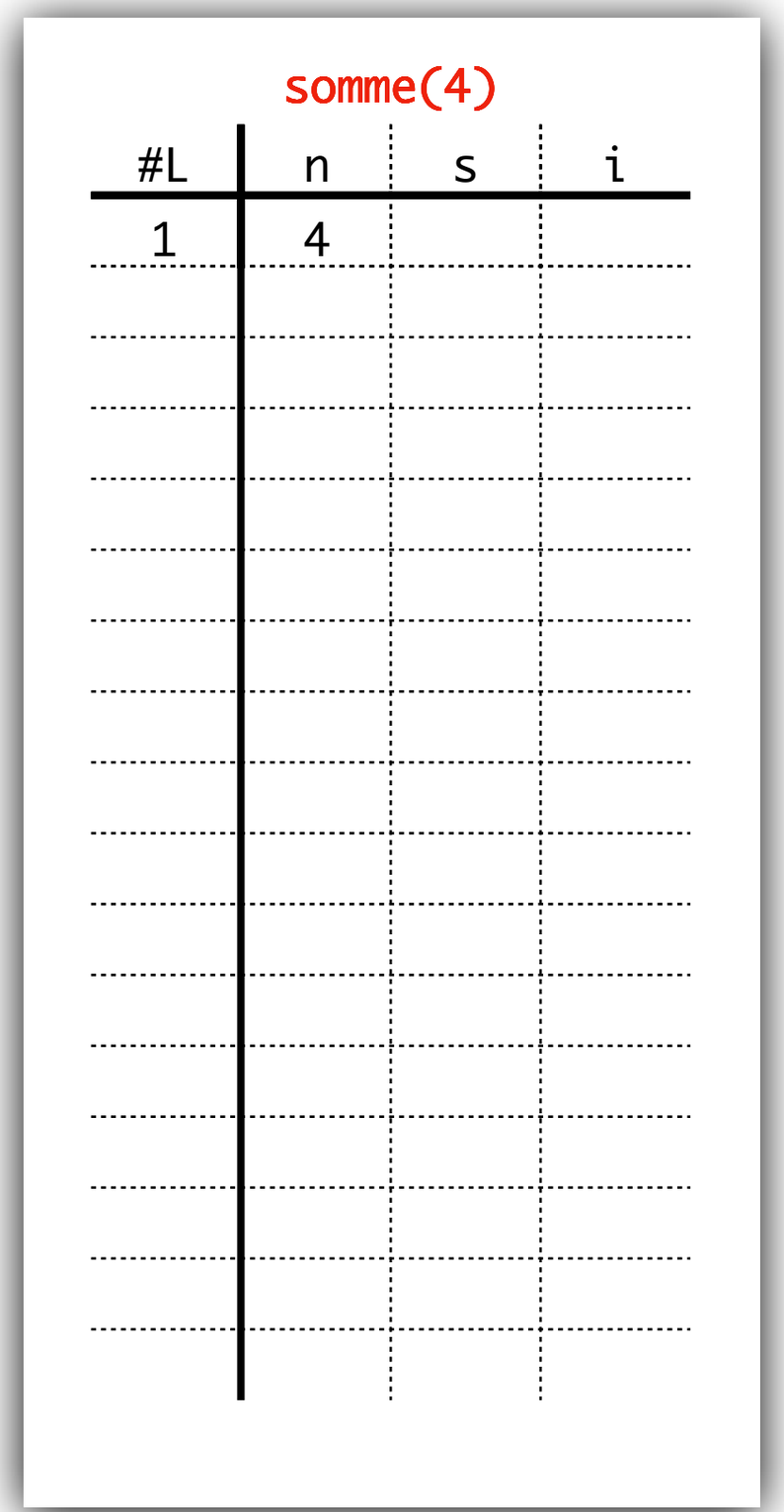
```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

somme(4)

#L	n	s	i
1	4		

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```



Somme de 1 à 4

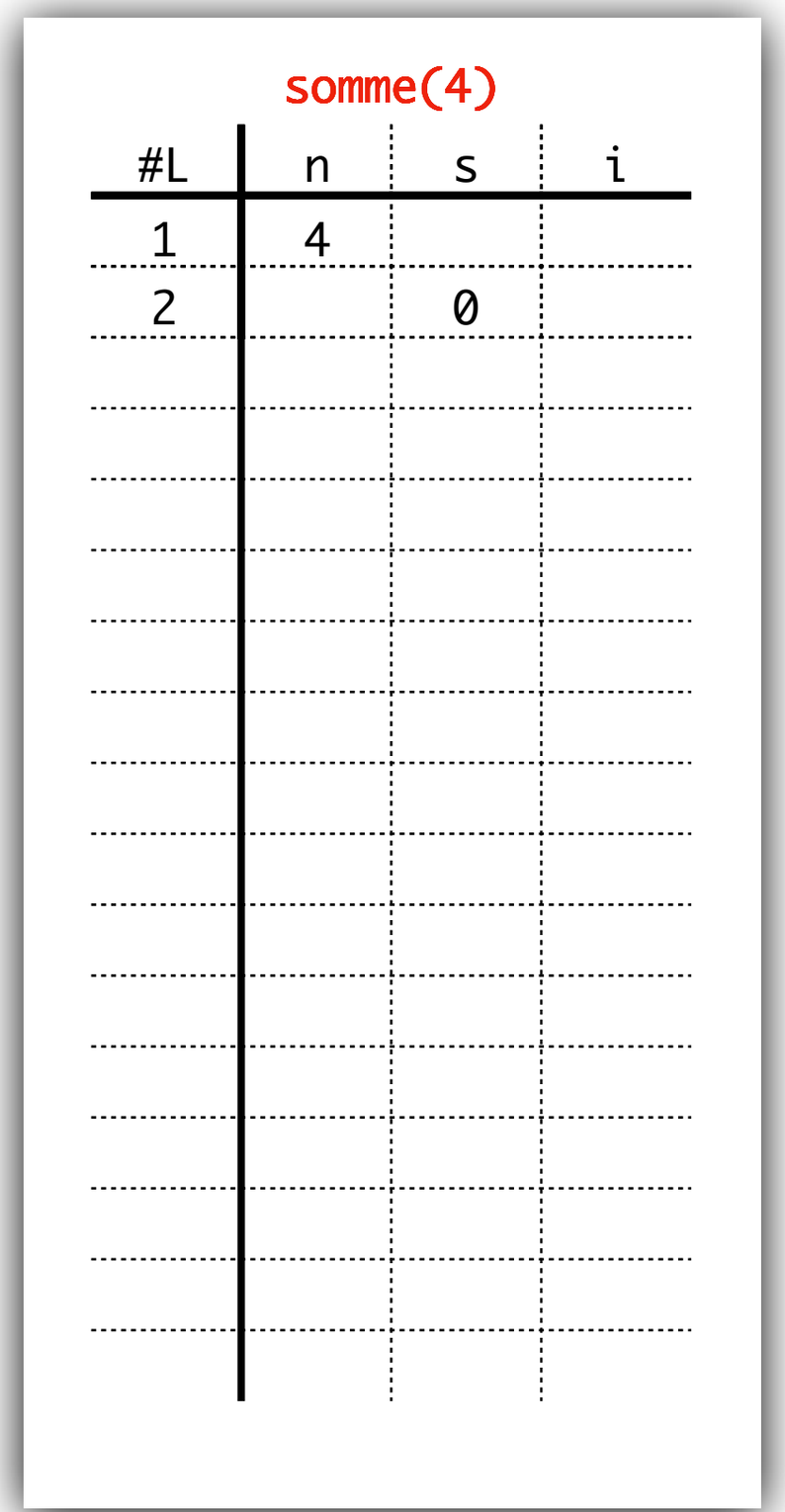
```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

somme(4)

#L	n	s	i
1	4		
2		0	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```



Somme de 1 à 4

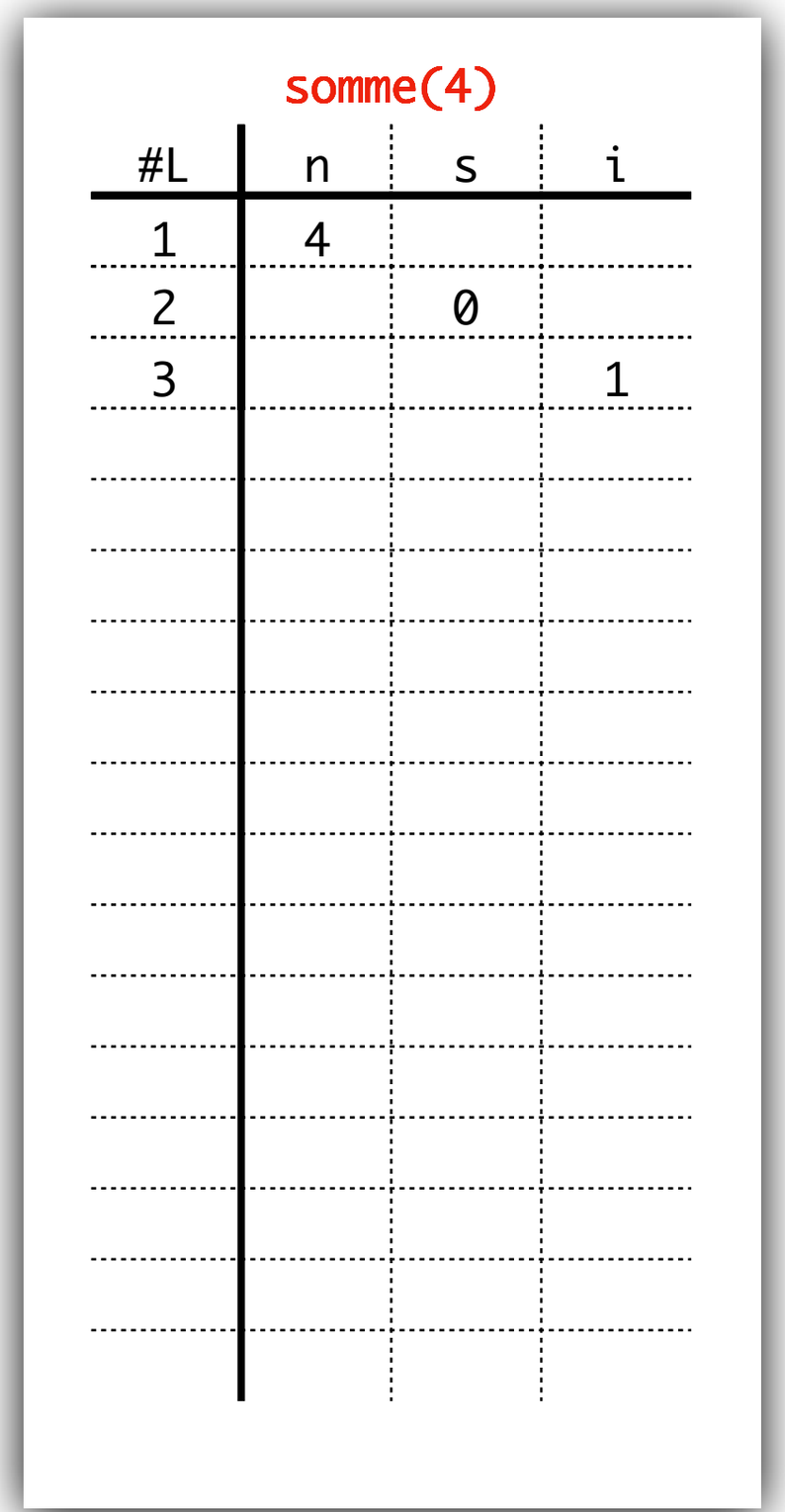
```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

somme(4)

#L	n	s	i
1	4		
2		0	
3			1

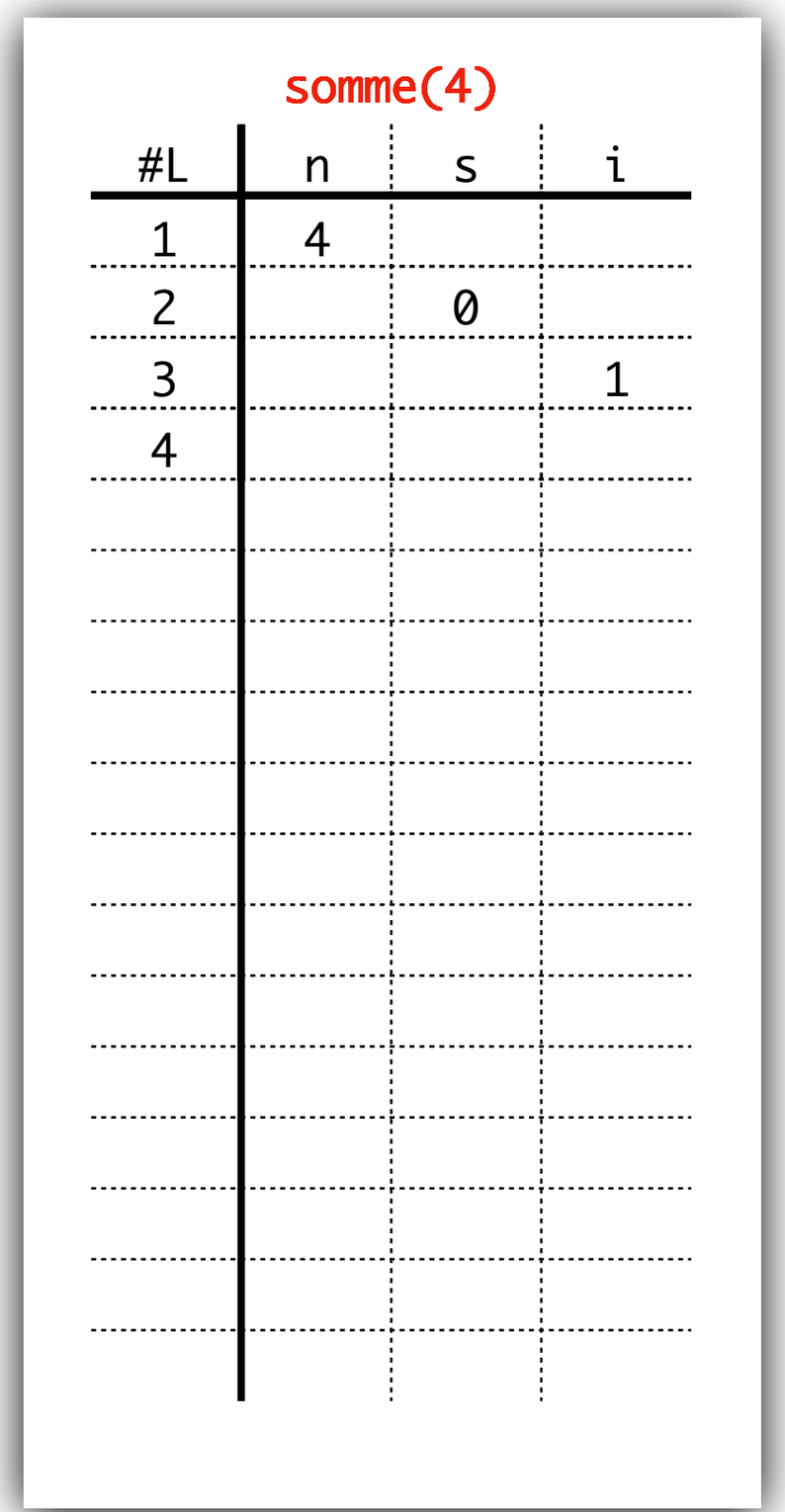
Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```



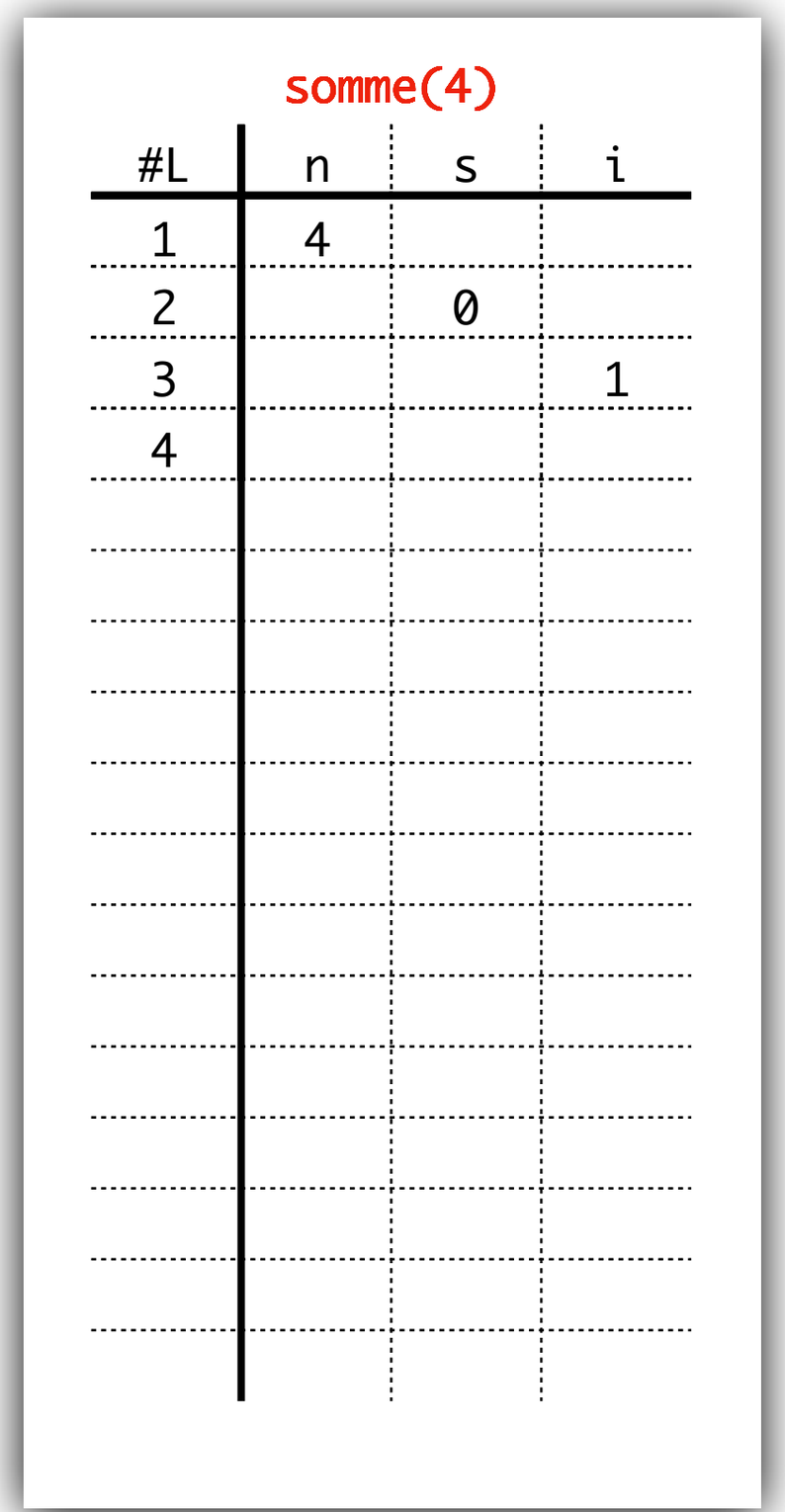
Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```



Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```



Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 6

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 6

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```


somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```


somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```


somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 6

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 4

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 4

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```


somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```


somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 6

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 6

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 4

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 4

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```


somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 6

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 6

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	
6			5

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉 4

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	
6			5

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	
6			5
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	
6			5
4			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

👉

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	
6			5
4			
7			

Somme de 1 à 4

```
1  def somme(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6          i = i + 1
7      return s
```

somme(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
6			2
4			
5		3	
6			3
4			
5		6	
6			4
4			
5		10	
6			5
4			
7			

résultat : 10

Exercice 7 du TD2

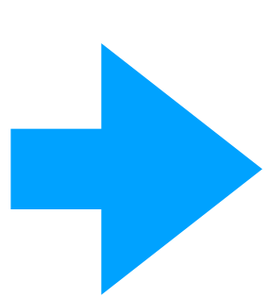
Programmes qui ne terminent pas

Somme de 1 à n, erronée

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

Somme de 1 à n, erronée

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```



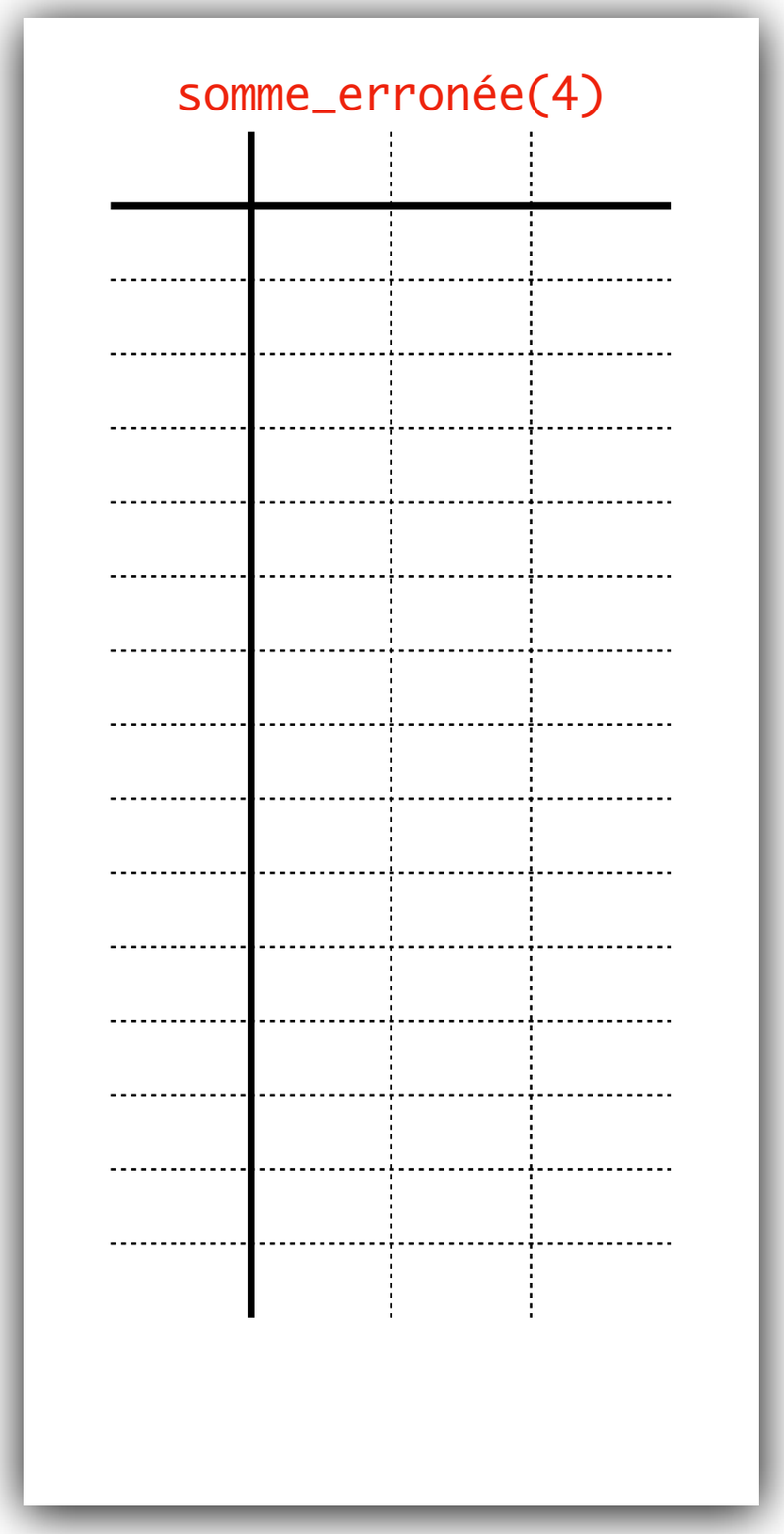
```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```



Exécution de la somme erronée

```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

somme_erronée(4)

#L	n	s	i

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		

Exécution de la somme erronée

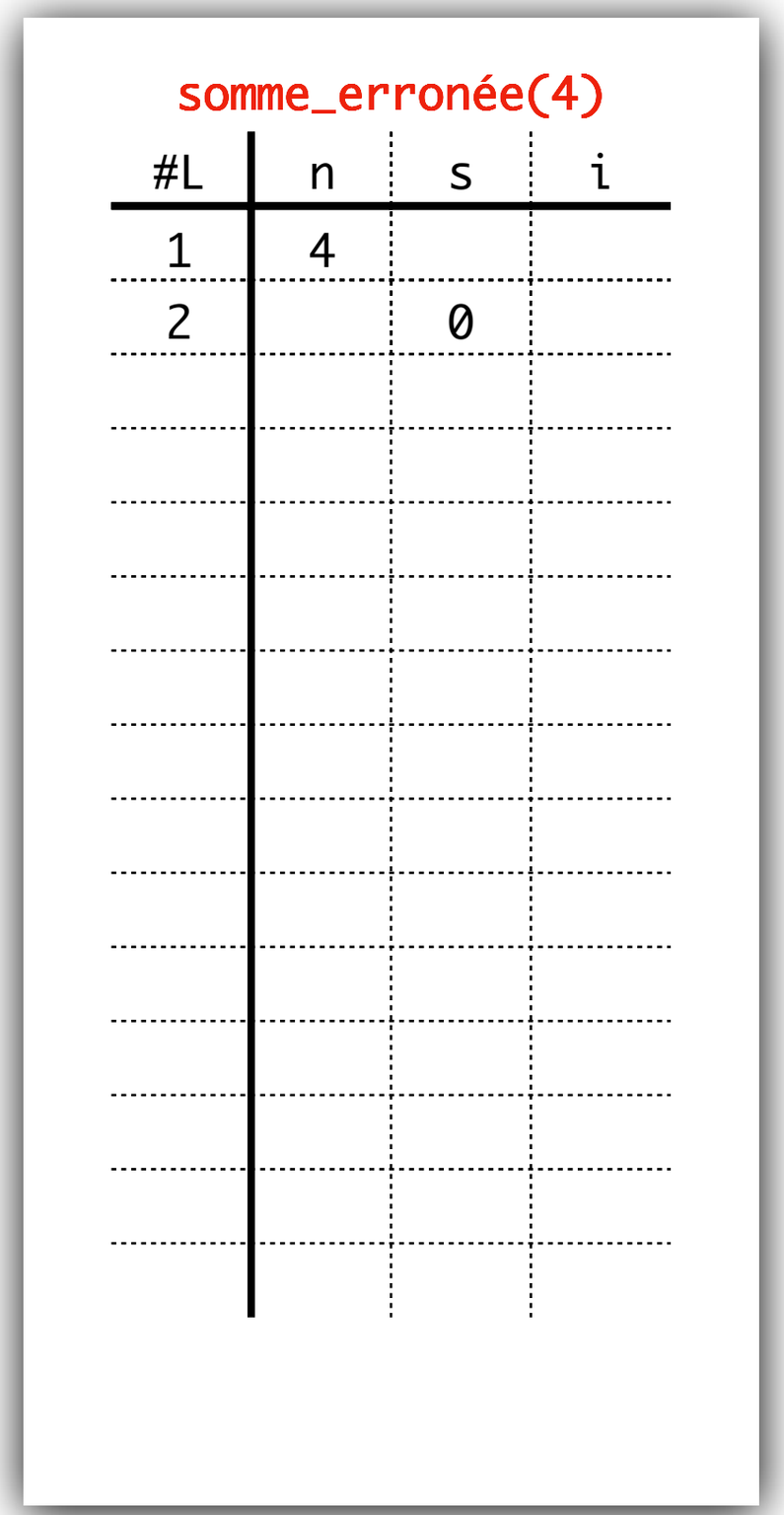
```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```



Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

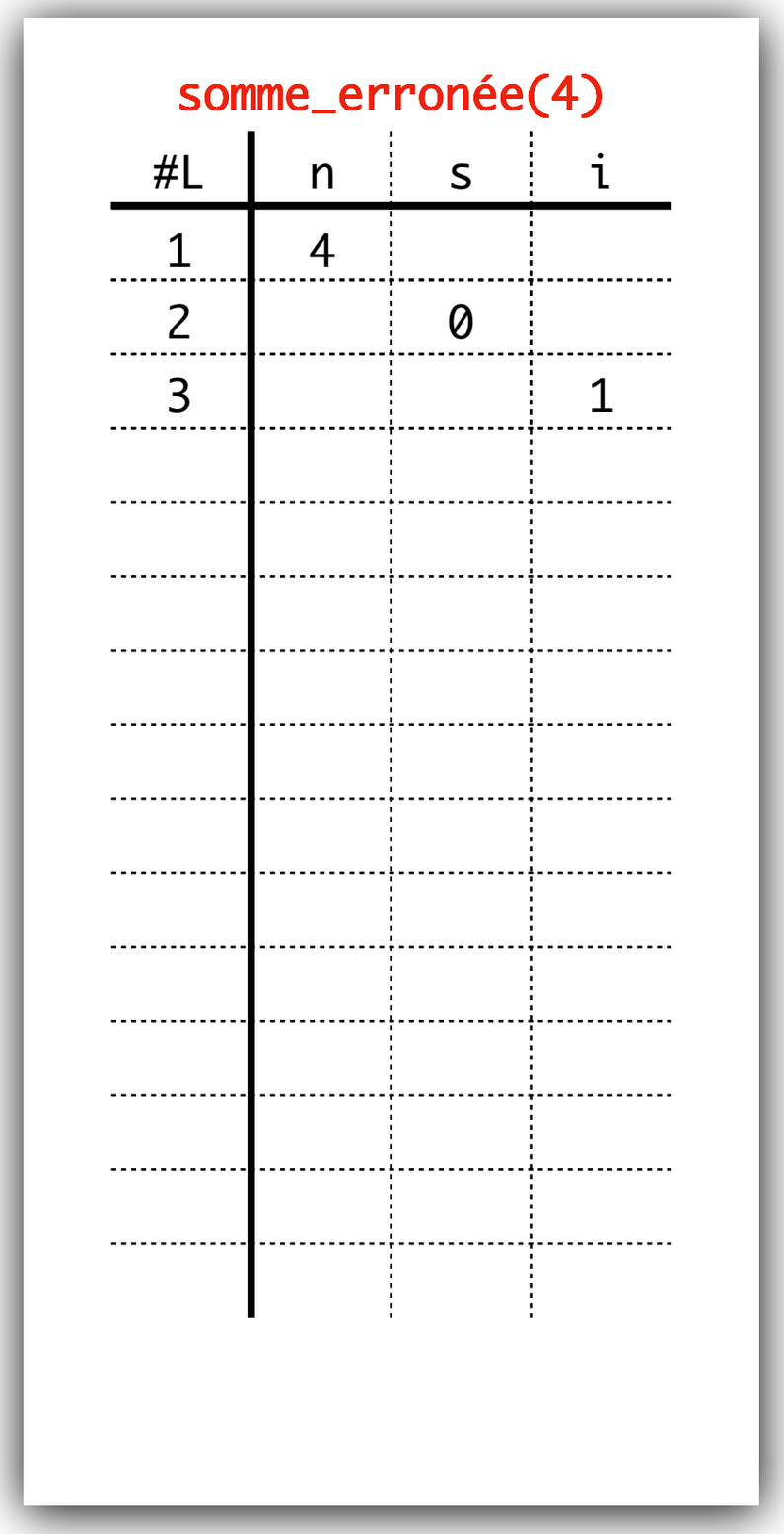
👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1

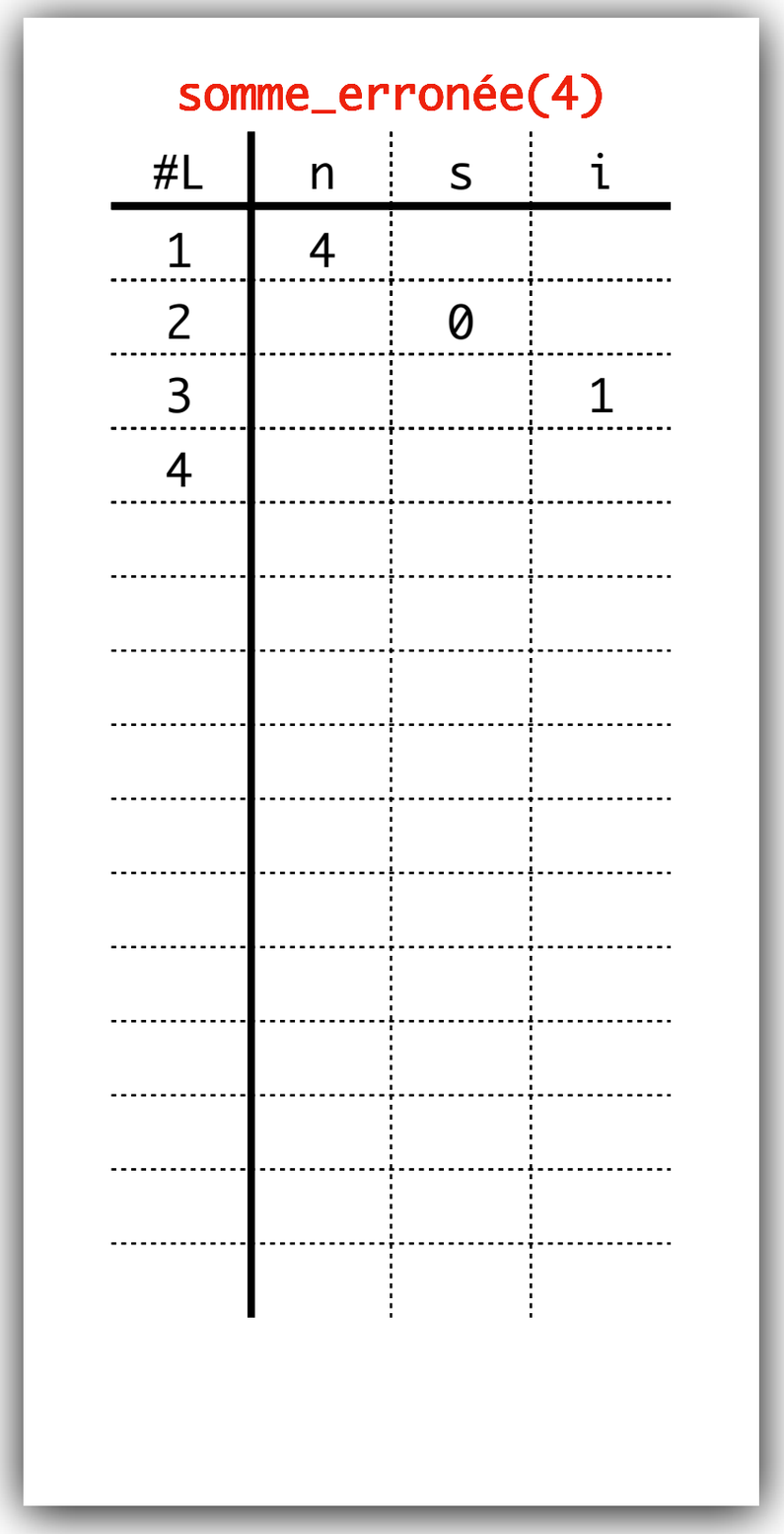
Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```



Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```



Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

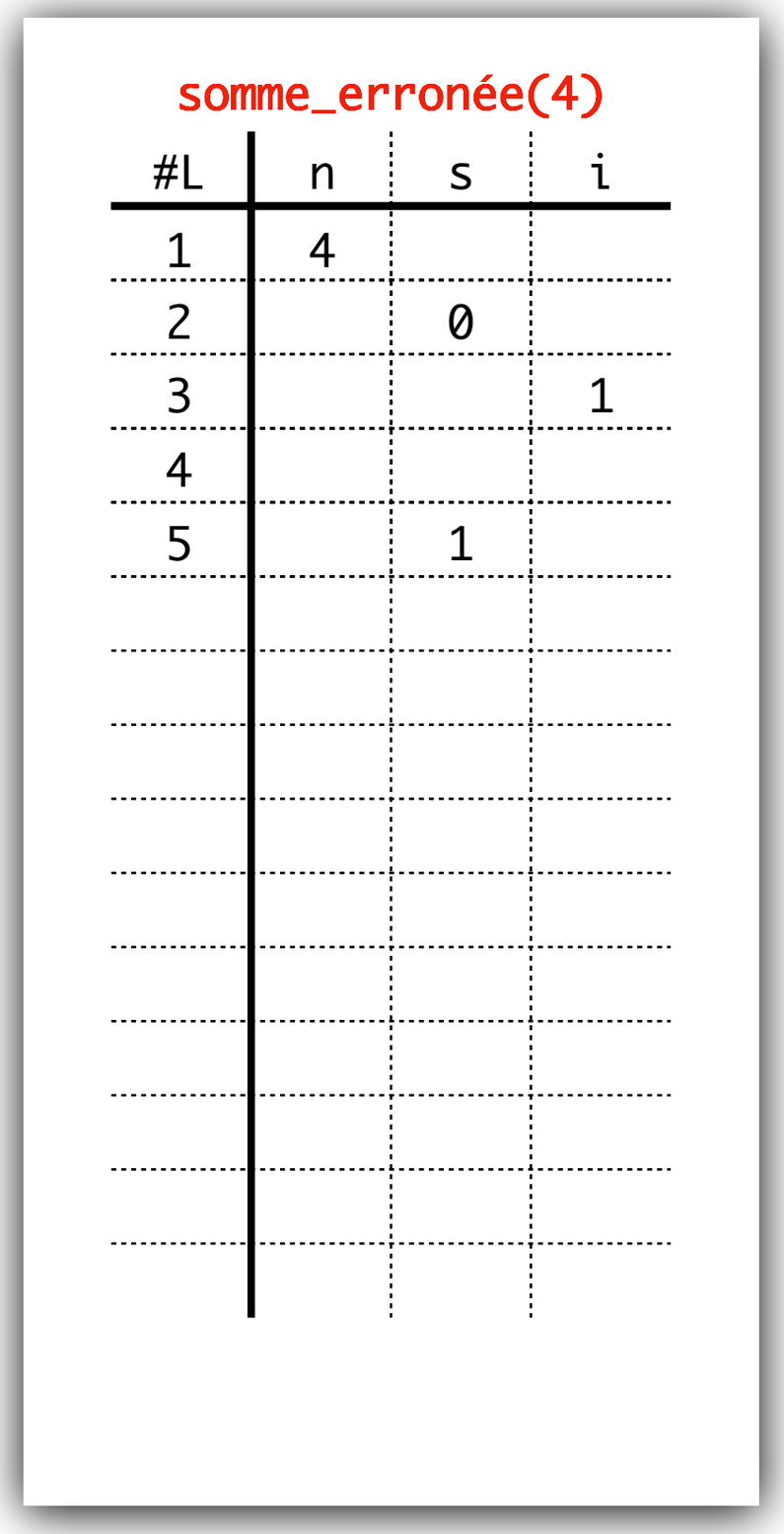
👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```



Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉 5

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```


somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉 4

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉 4

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			
5		5	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉 4

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			
5		5	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			
5		5	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			
5		5	
4			

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

👉

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			
5		5	
4			
5		6	

Exécution de la somme erronée

```
1  def somme_erronée(n):
2      s = 0
3      i = 1
4      while i <= n:
5          s = s + i
6      return s
```

somme_erronée(4)

#L	n	s	i
1	4		
2		0	
3			1
4			
5		1	
4			
5		2	
4			
5		3	
4			
5		4	
4			
5		5	
4			
5		6	

∞

Importance de la terminaison

- Un programme qui ne termine pas ne donne pas de résultat et exécute un nombre infini d'instructions

- On exige que nos algorithmes terminent toujours

- Il y a certains programmes utiles qui sont conçus pour ne pas terminer en conditions normales (un serveur web, un système d'exploitation…) mais ici on ne les considère pas comme des algorithmes

Montrer la terminaison

Terminaison d'un programme

- Pour le moment, la seule raison qu'on connaît qui puisse causer la non terminaison d'un programme est une boucle `while` qui ne termine pas

- Pour qu'une boucle while termine il est nécéssaire que sa condition devienne, à un moment donné, fausse

Terminaison de somme

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

Terminaison de `somme`

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

pour terminer,
cette condition doit
devenir fausse

Terminaison de `somme`

`i` a initialement la valeur 1

pour terminer, cette condition doit devenir fausse

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

Terminaison de `somme`

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

`i` a initialement la valeur 1

pour terminer, cette condition doit devenir fausse

`i` augmente strictement à chaque tour de boucle

Terminaison de `somme`

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

`i` a initialement la valeur 1

pour terminer, cette condition doit devenir fausse

alors que `n` ne change pas à l'intérieur de la boucle

`i` augmente strictement à chaque tour de boucle

Terminaison de `somme`

```
def somme(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
        i = i + 1
    return s
```

`i` a initialement la valeur 1

pour terminer, cette condition doit devenir fausse

alors que `n` ne change pas à l'intérieur de la boucle

`i` augmente strictement à chaque tour de boucle

donc tôt ou tard `i` va dépasser `n`

Exercice 8 du TD2

Montrer la non-terminaison

Non-terminaison de `somme_erronée` si $n \geq 1$

```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

Non-terminaison de `somme_erronée` si $n \geq 1$

pour terminer,
cette condition doit
devenir fausse

```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

Non-terminaison de `somme_erronée` si $n \geq 1$

pour terminer,
cette condition doit
devenir fausse

`i` a initialement
la valeur `1`

```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

Non-terminaison de `somme_erronée` si $n \geq 1$

pour terminer,
cette condition doit
devenir fausse

`i` a initialement
la valeur `1`

```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

`i` ne change pas
à l'intérieur
de la boucle

Non-terminaison de `somme_erronée` si $n \geq 1$

pour terminer,
cette condition doit
devenir fausse

`i` a initialement
la valeur `1`

```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

`n` ne change pas
non plus

`i` ne change pas
à l'intérieur
de la boucle

Non-terminaison de `somme_erronée` si $n \geq 1$

pour terminer,
cette condition doit
devenir fausse

`i` a initialement
la valeur `1`

```
def somme_erronée(n):
    s = 0
    i = 1
    while i <= n:
        s = s + i
    return s
```

`n` ne change pas
non plus

`i` ne change pas
à l'intérieur
de la boucle

donc la condition
reste toujours vraie

Exercice 9 du TD2

Itération sur un intervalle borné avec la boucle `for`

Puissances de 2

```
def power2(n):
    p = 1
    for i in range(n):
        p = 2 * p
    return p
```

Somme de 0 à n avec la boucle for

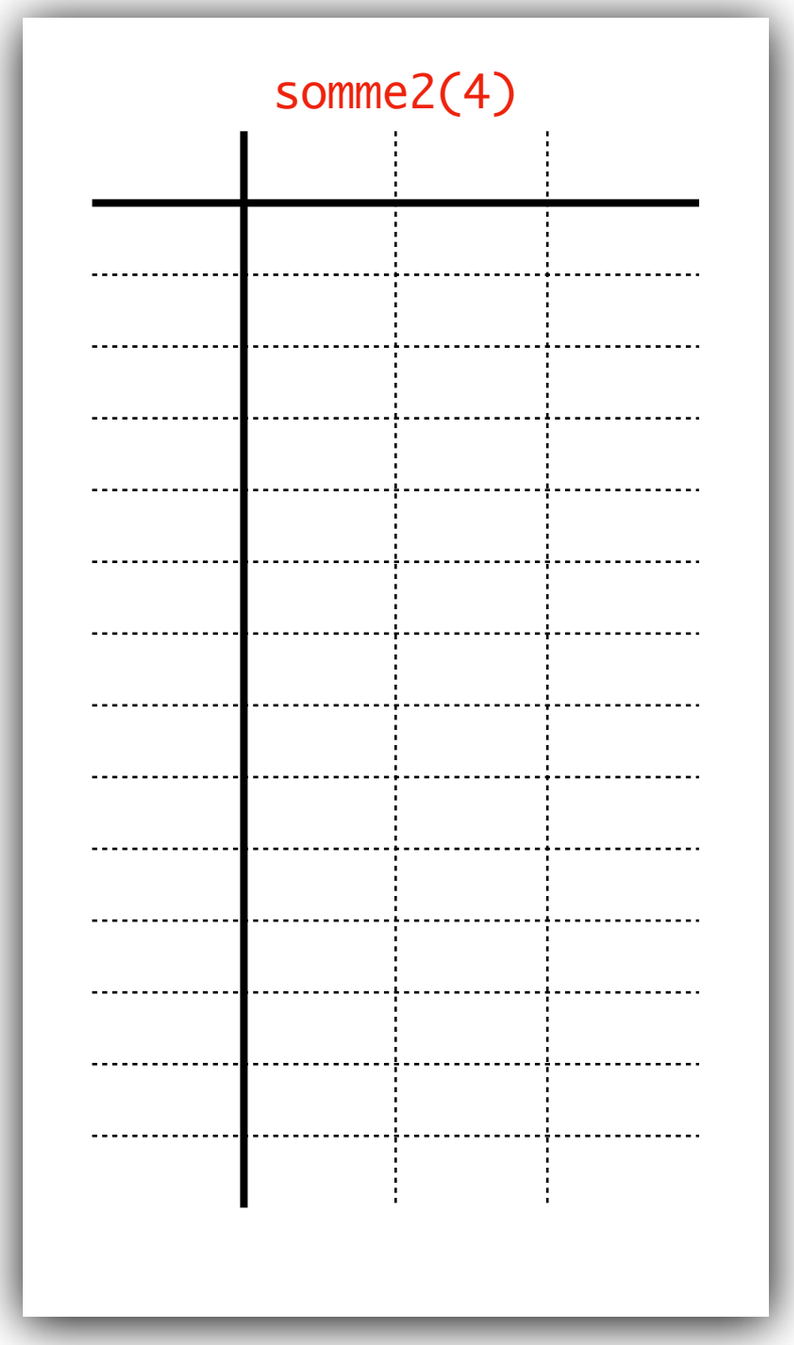
```
def somme2(n):
    s = 0
    for i in range(n + 1):
        s = s + i
    return s
```

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

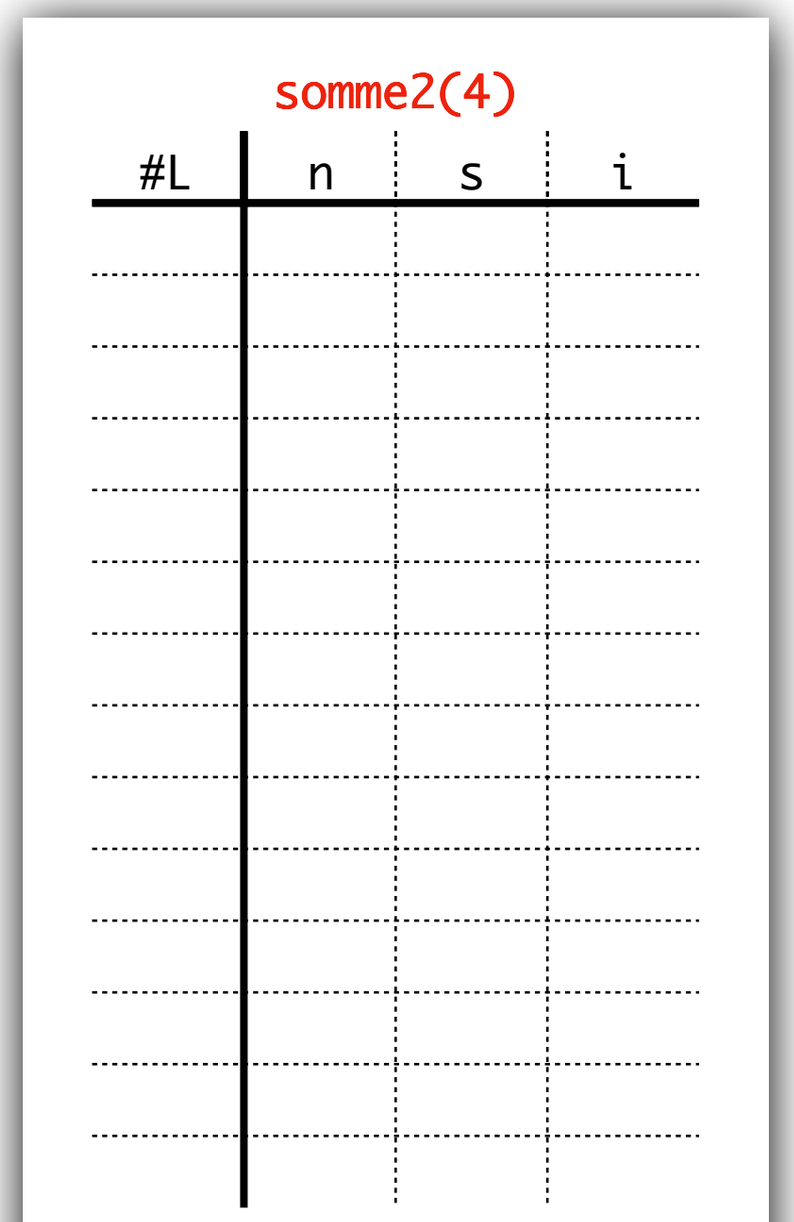
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



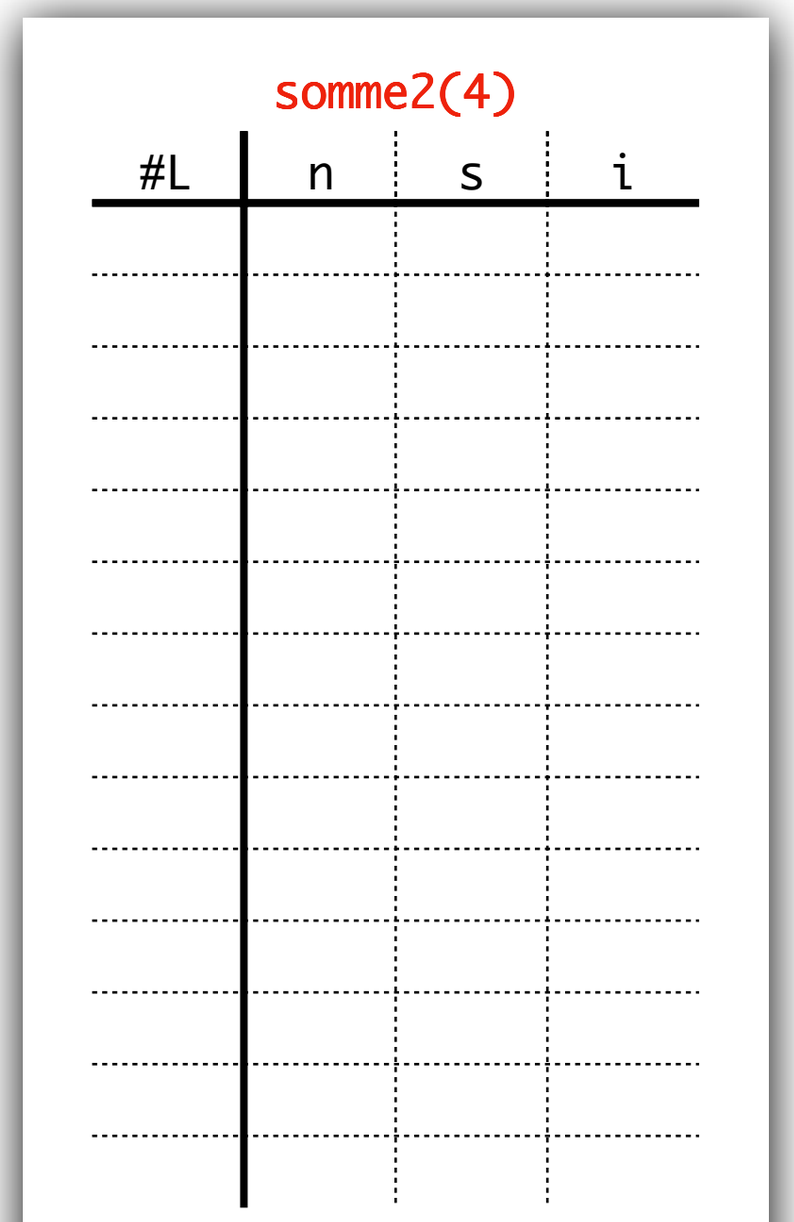
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



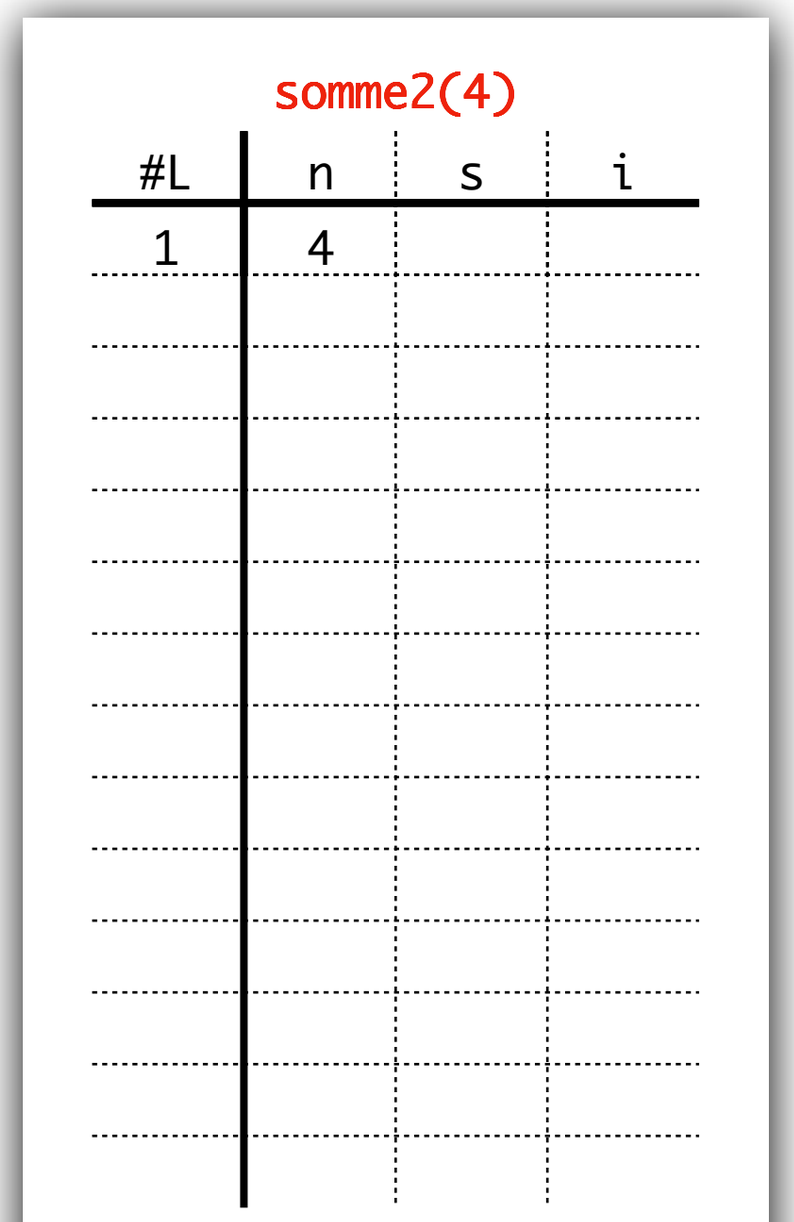
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



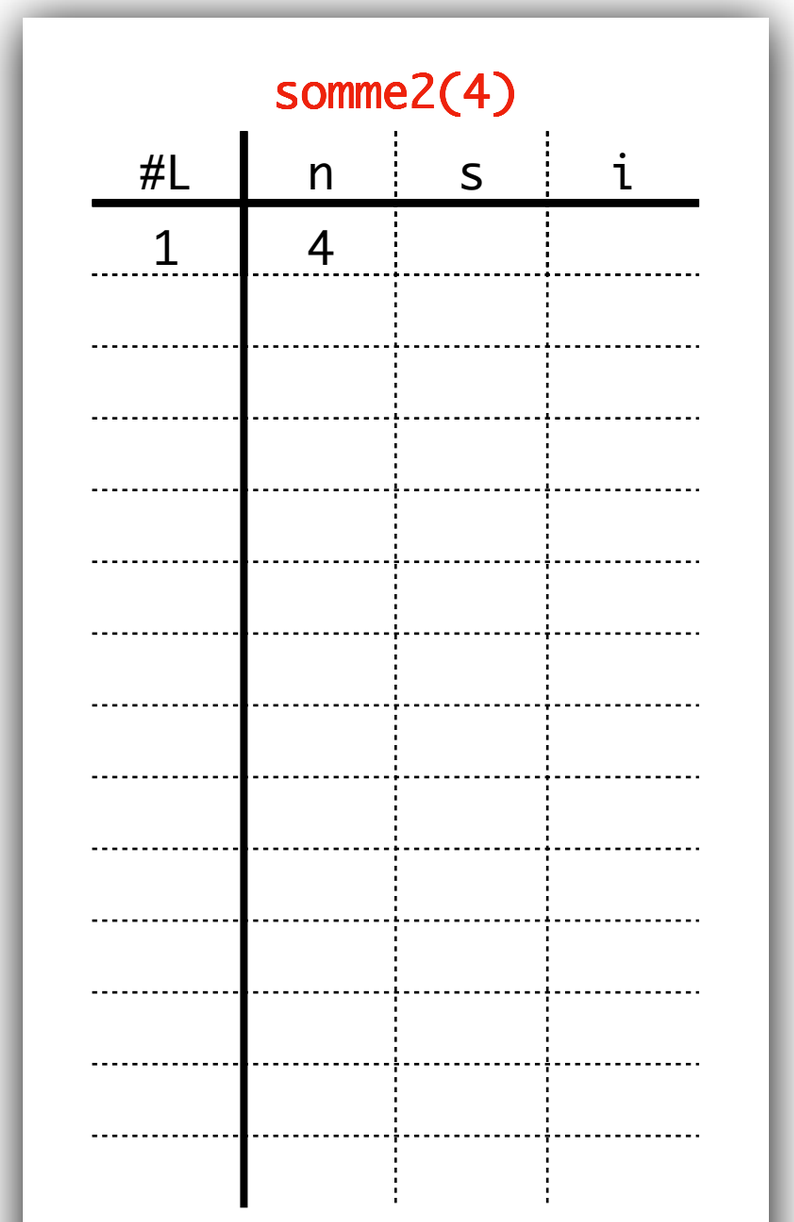
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



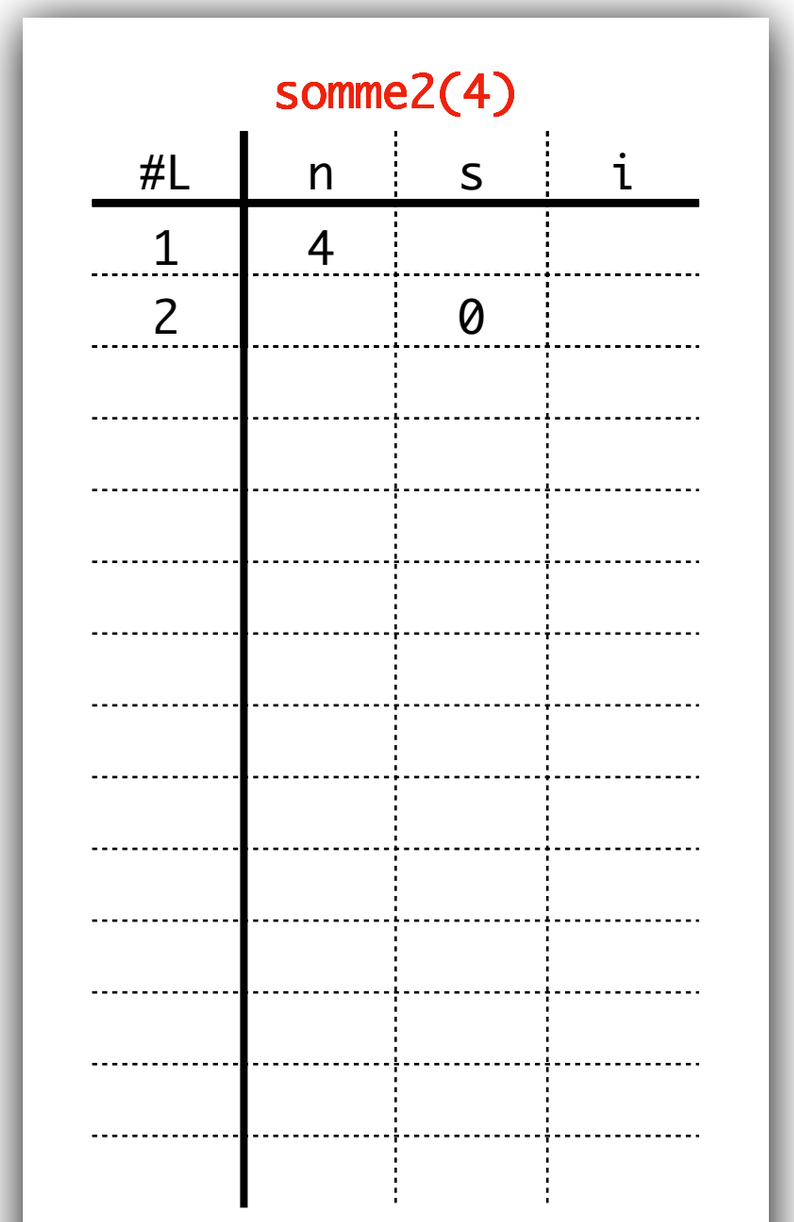
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



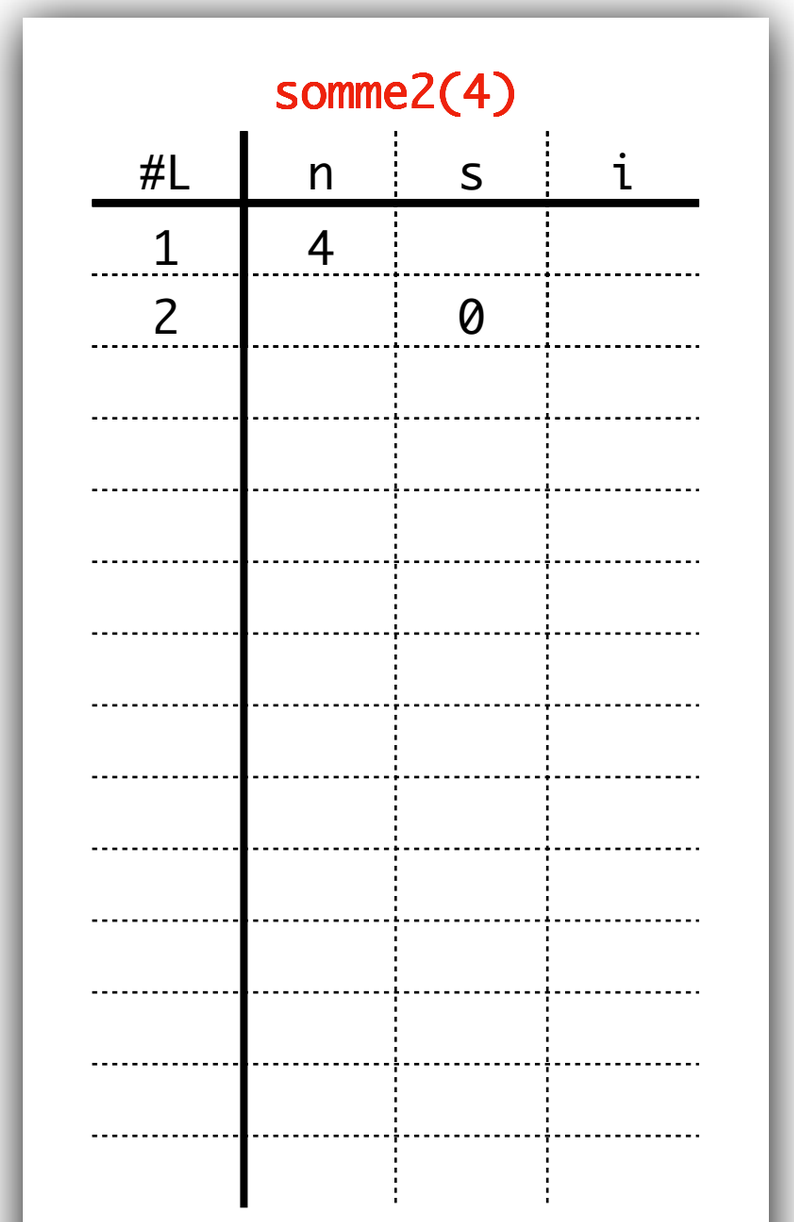
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



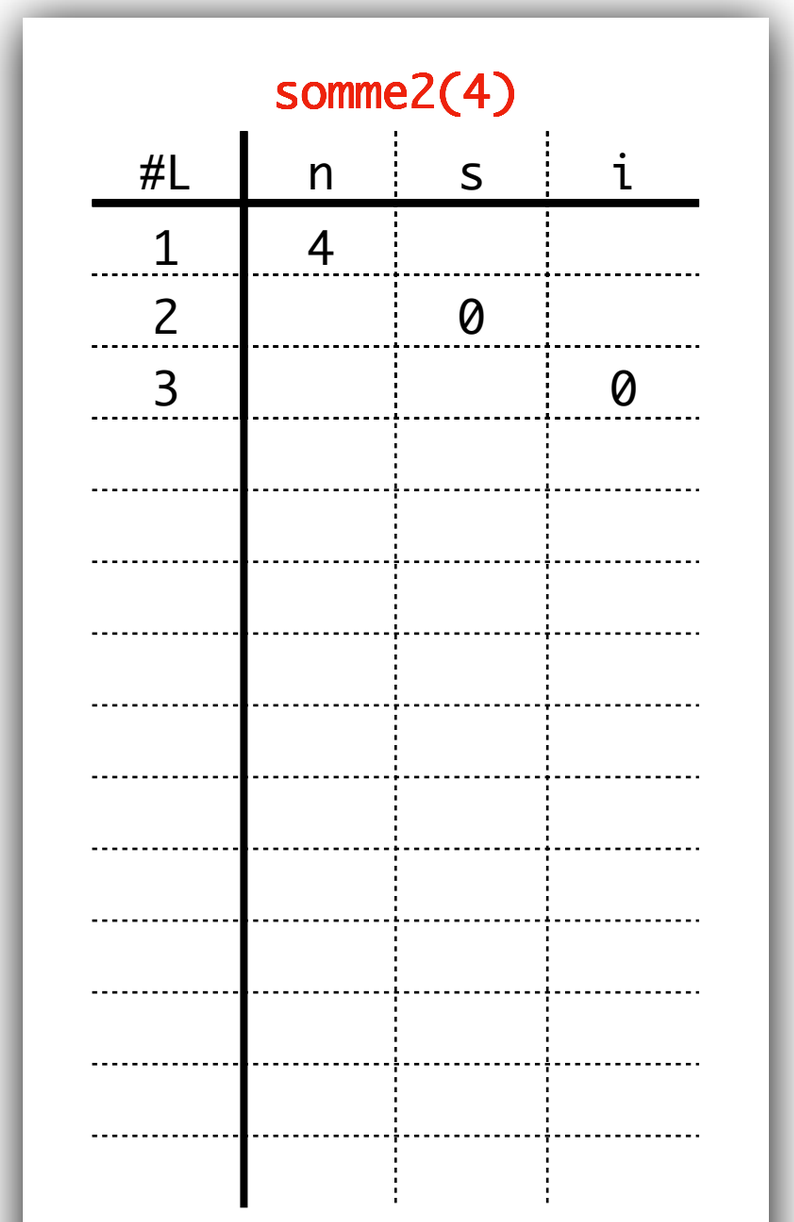
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



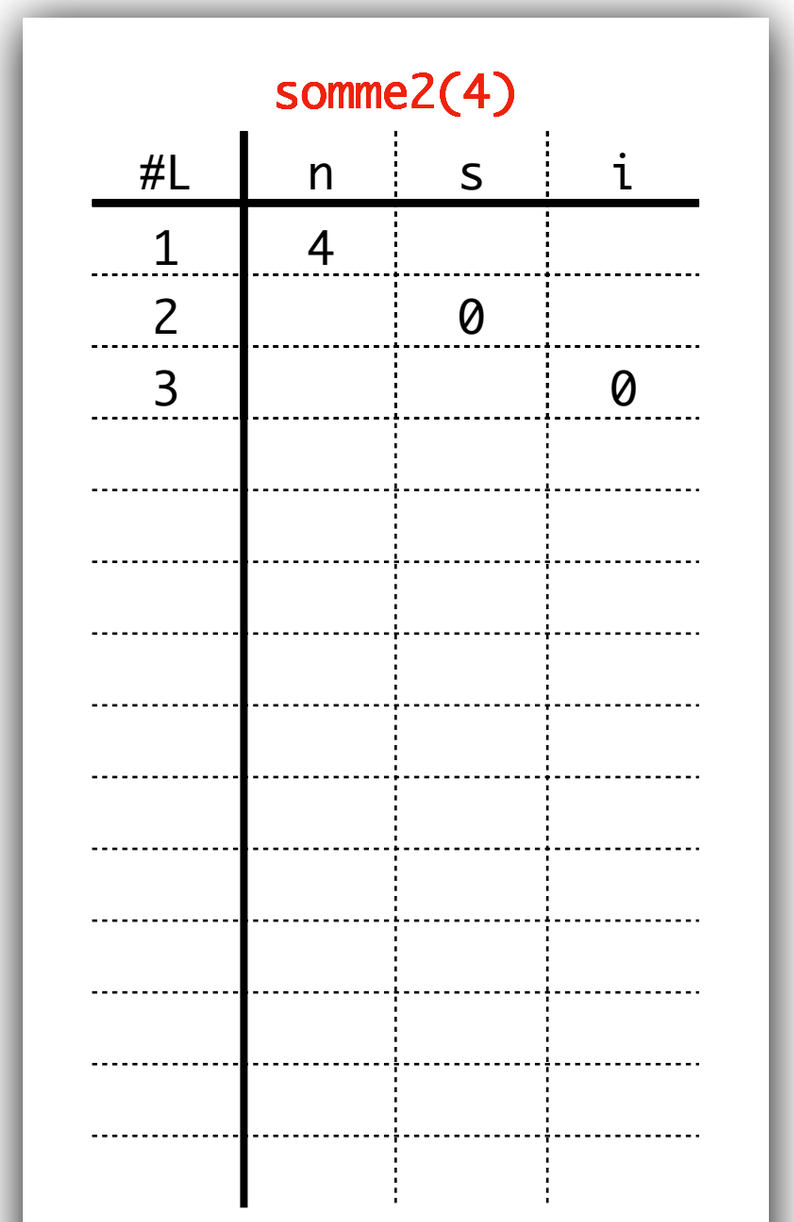
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



Somme de 0 à 4 avec `for`

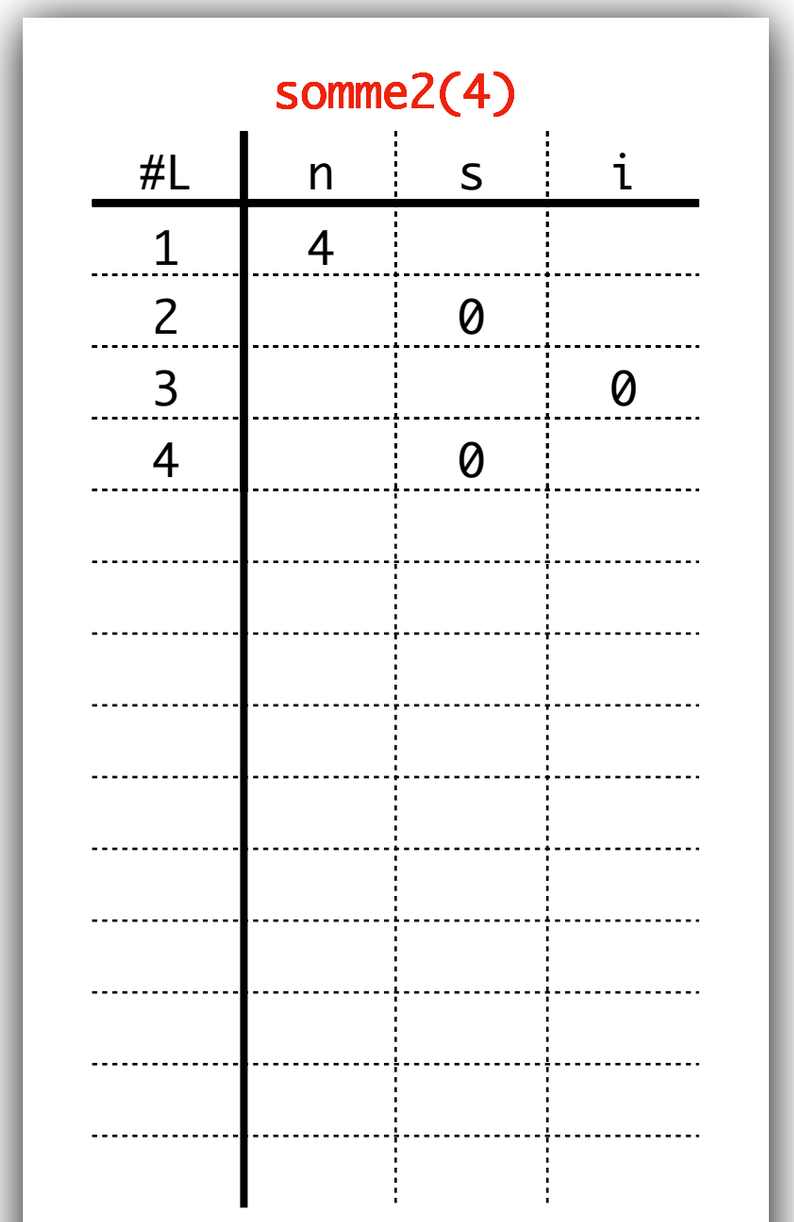
```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
4		0	

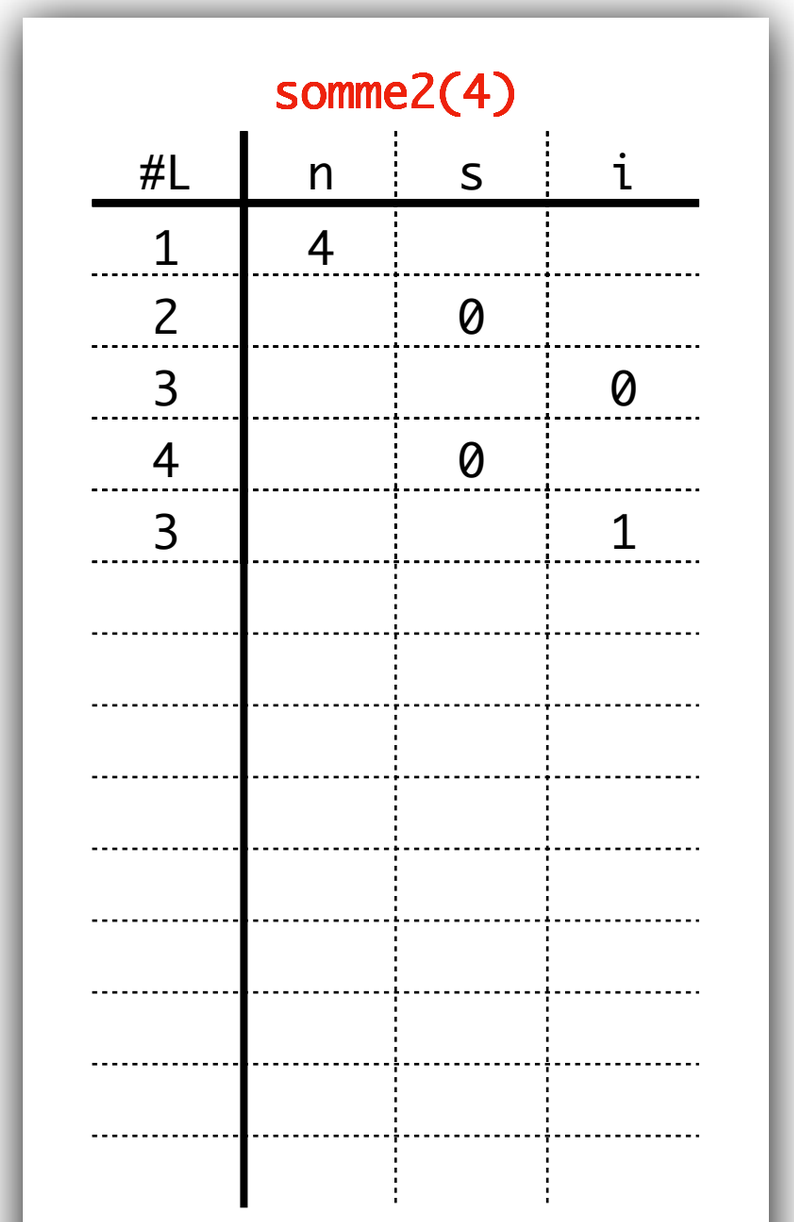
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



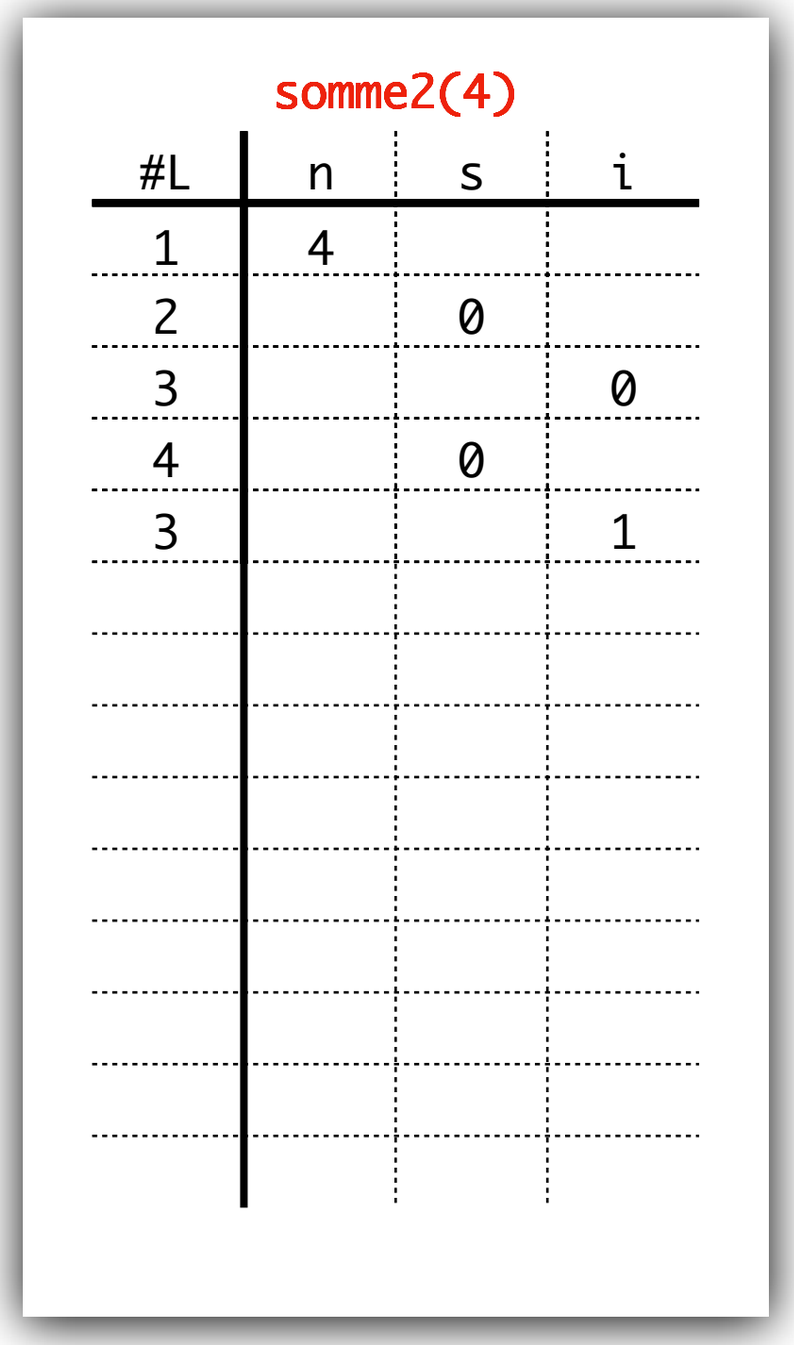
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



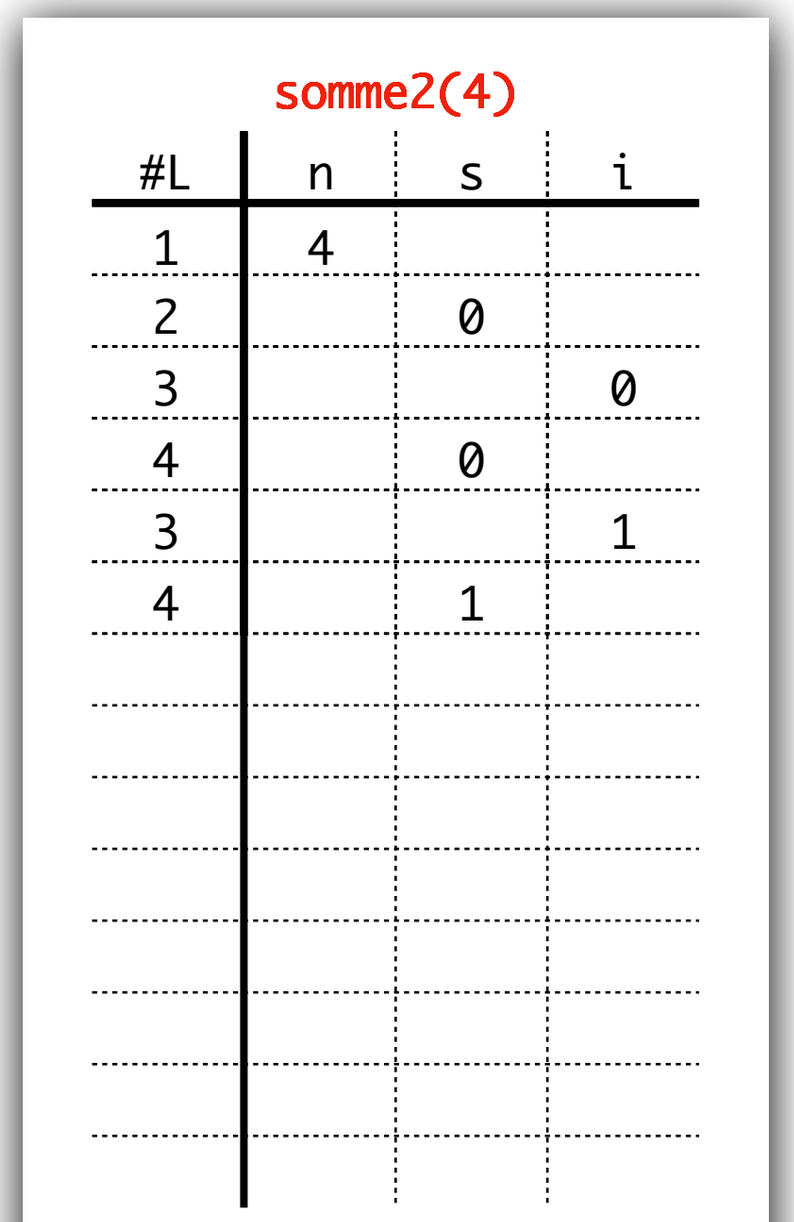
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



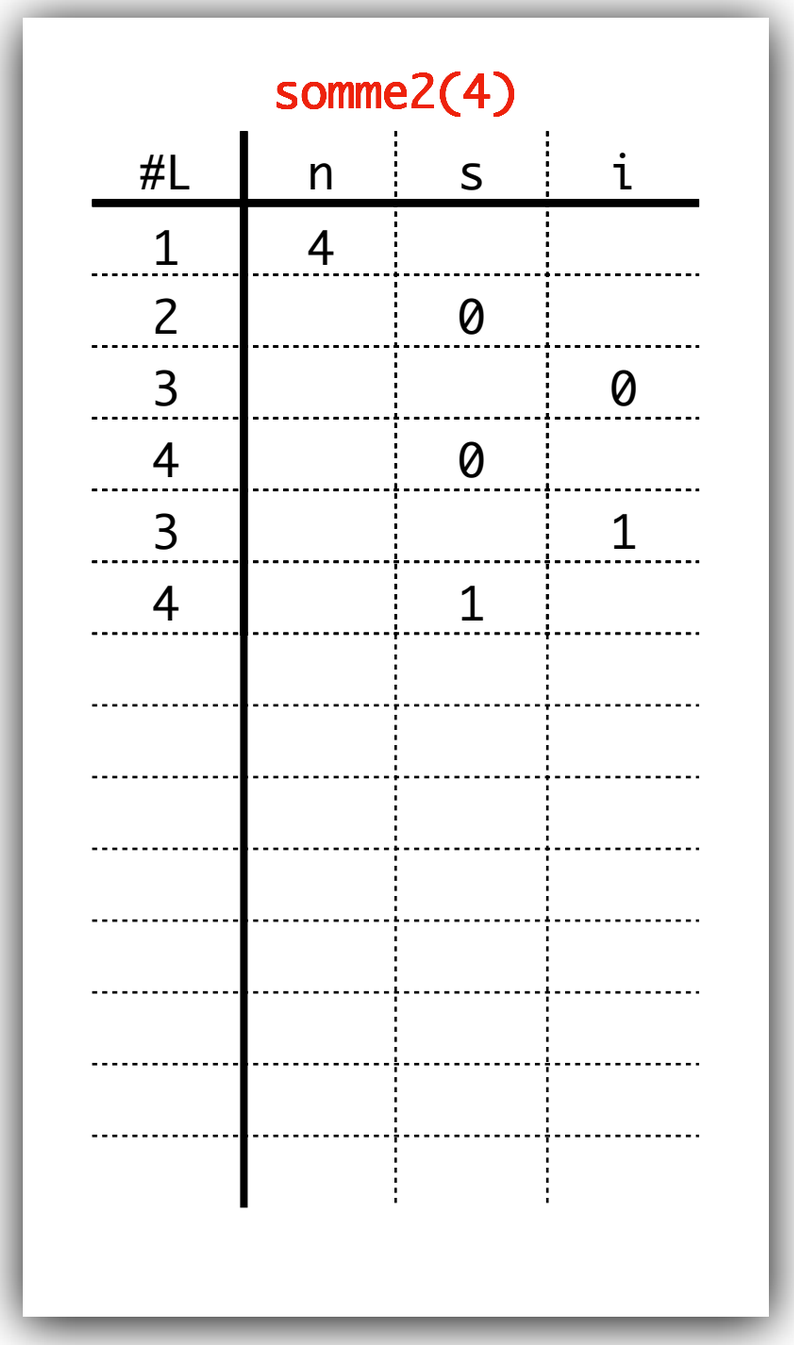
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



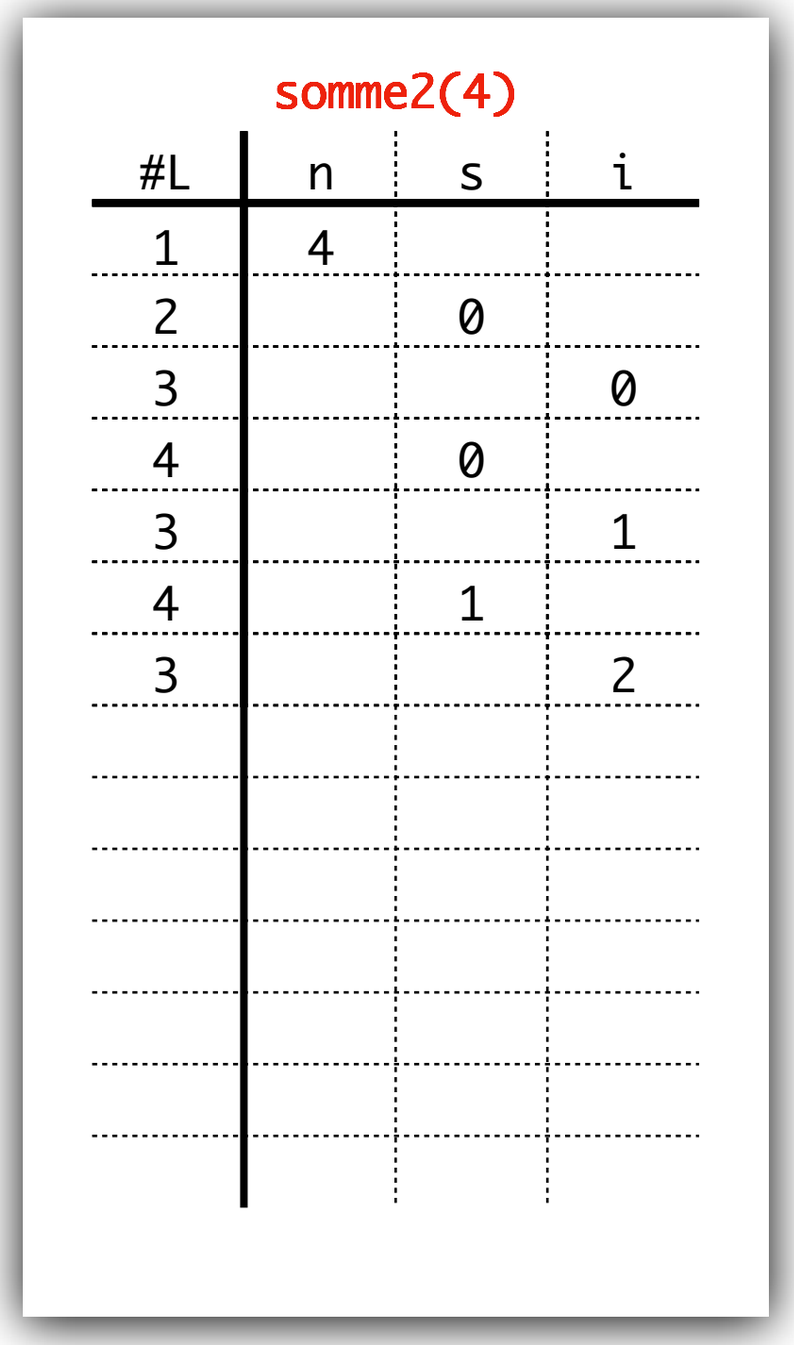
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



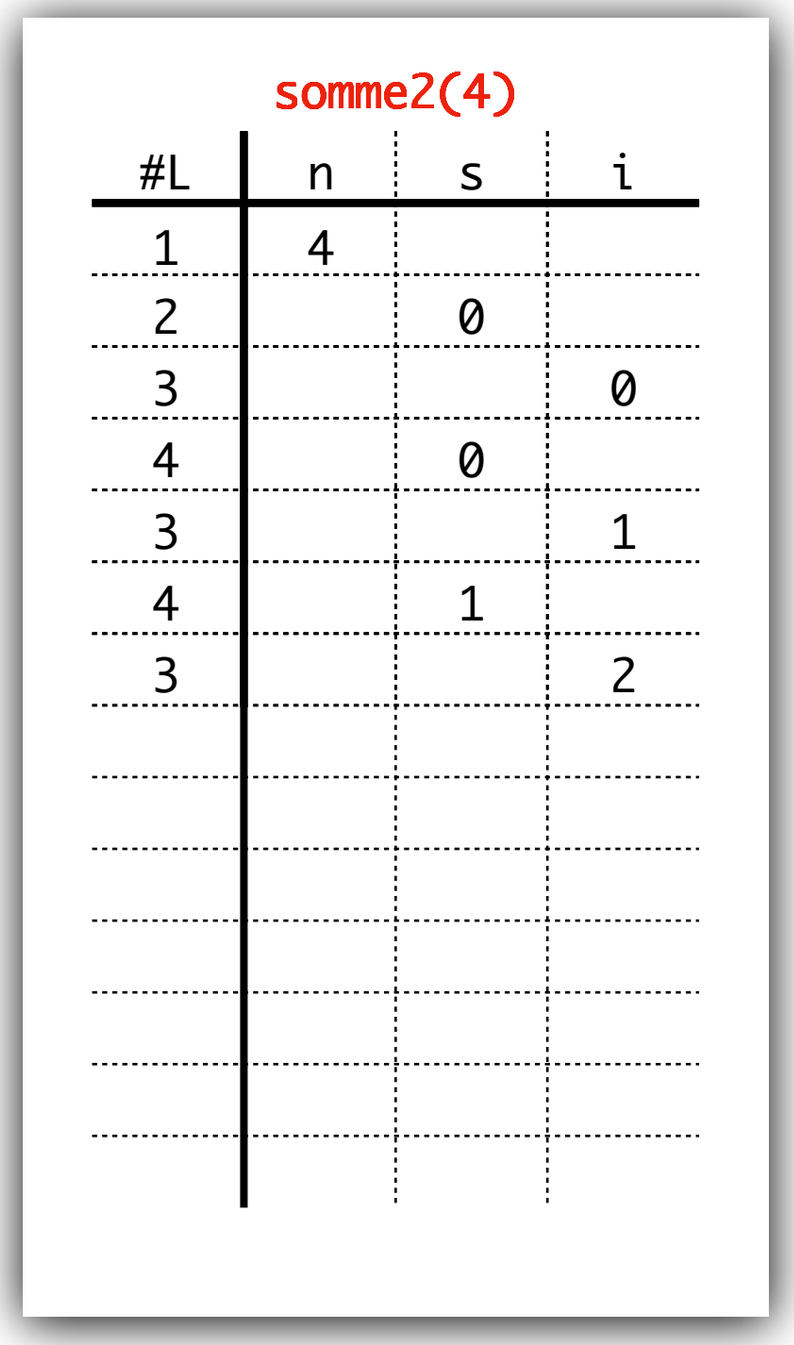
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



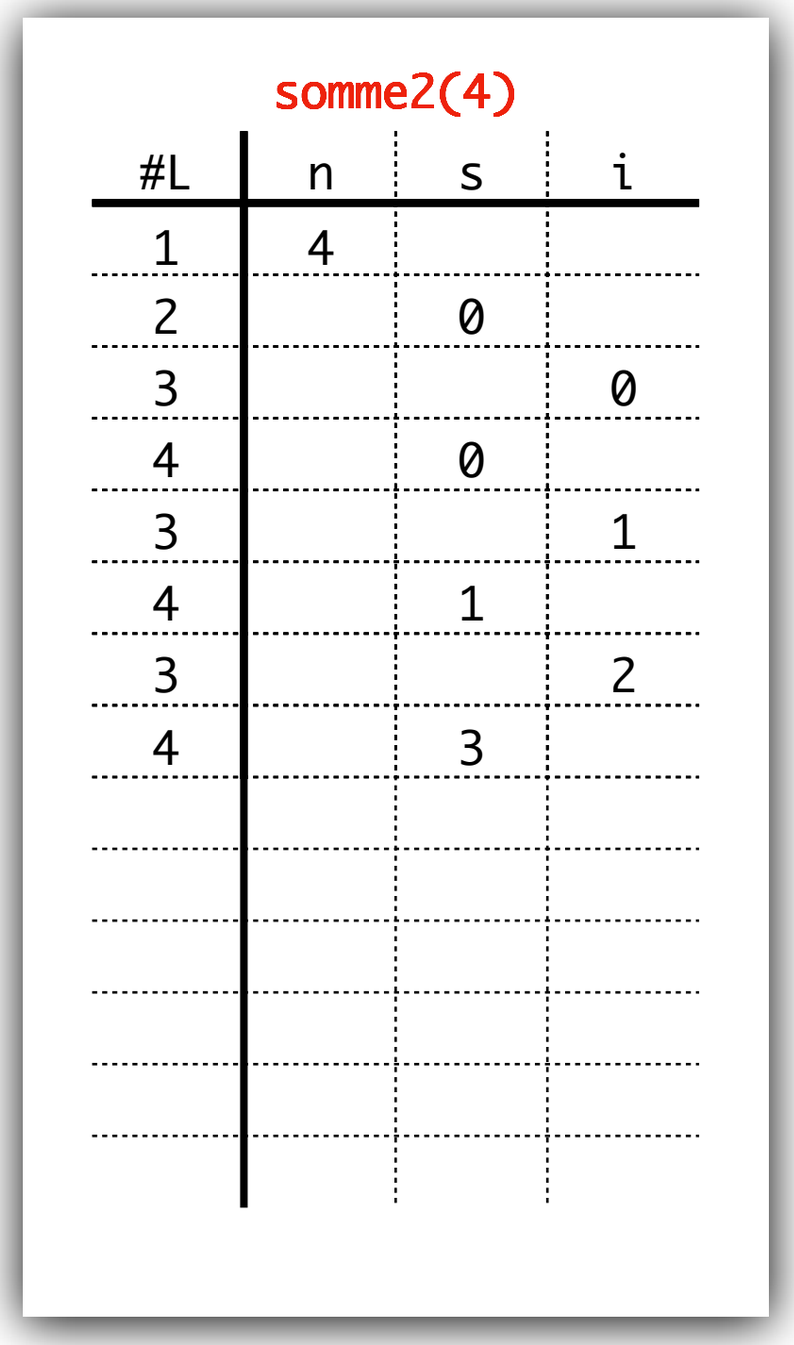
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



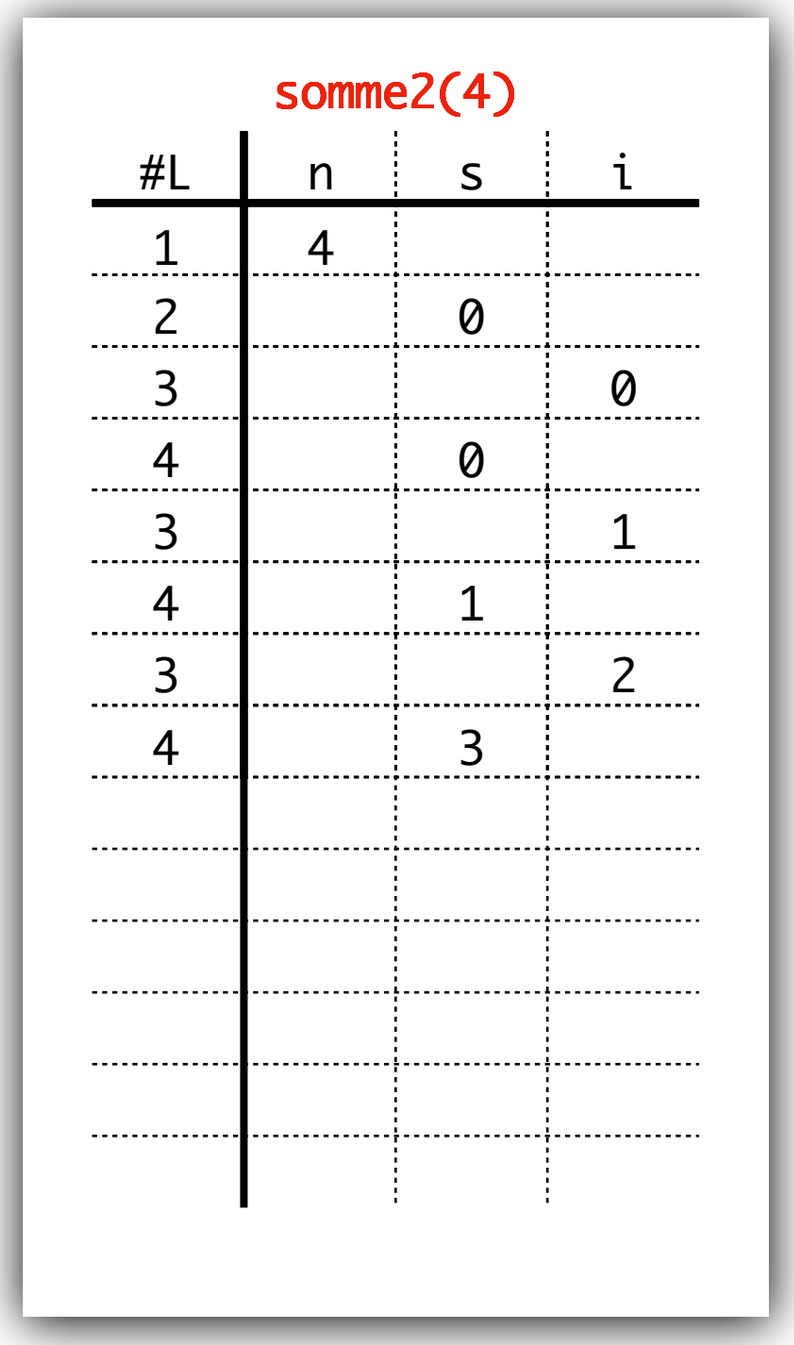
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



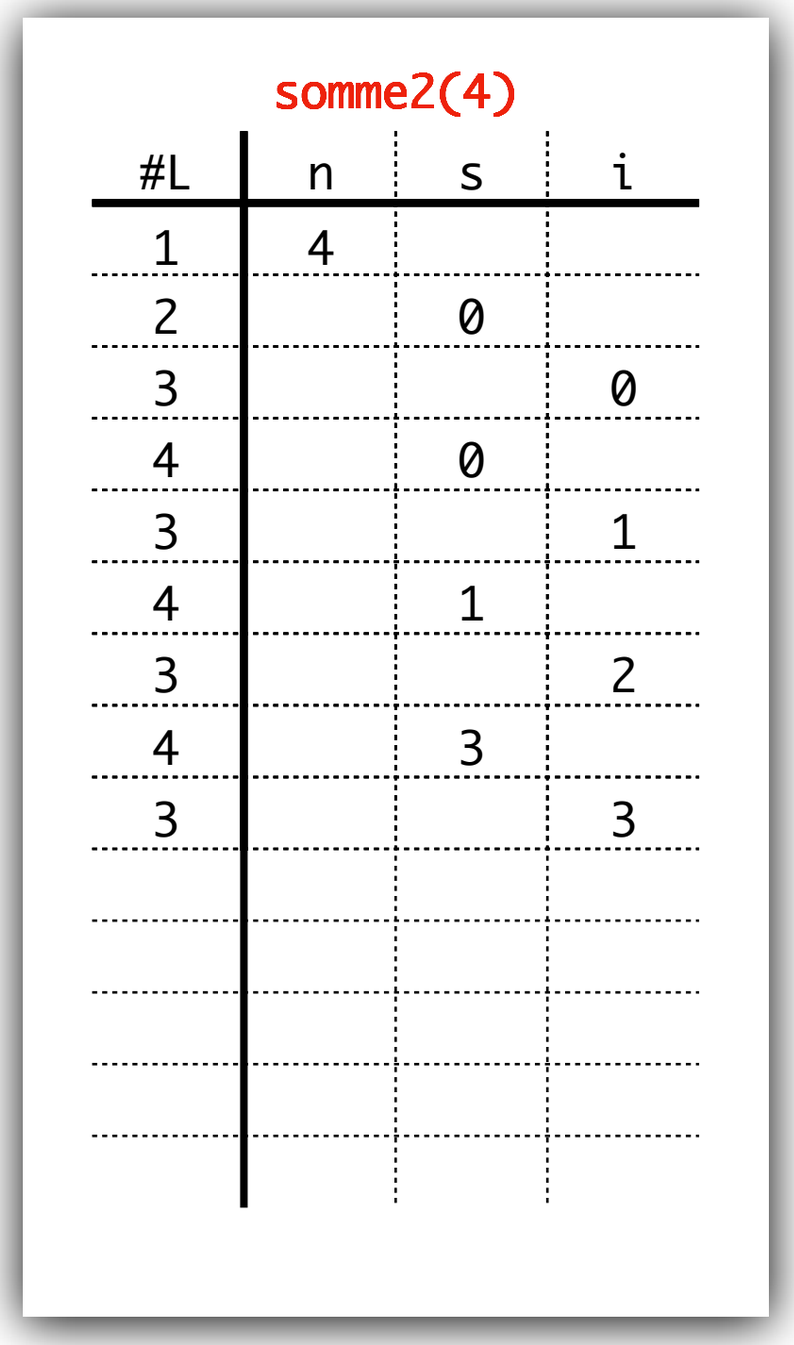
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



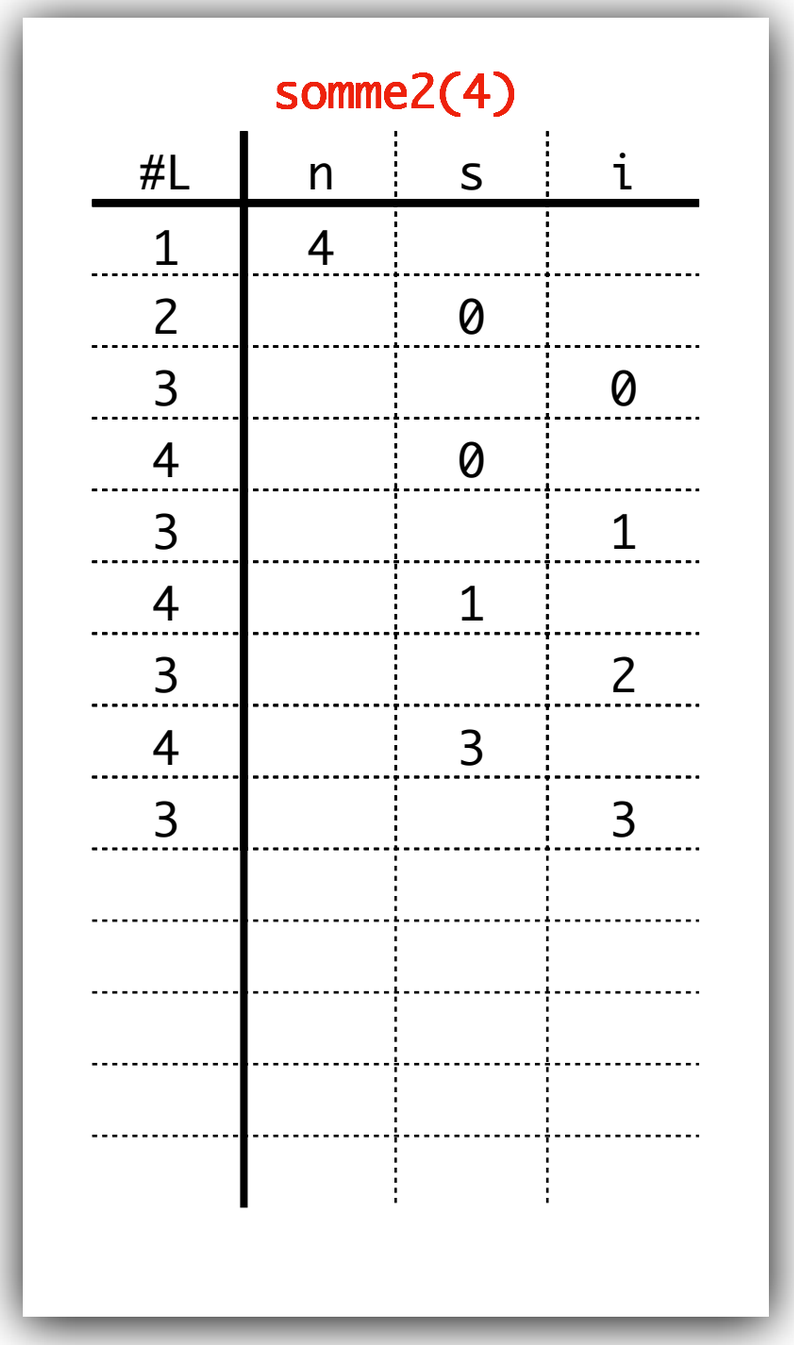
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



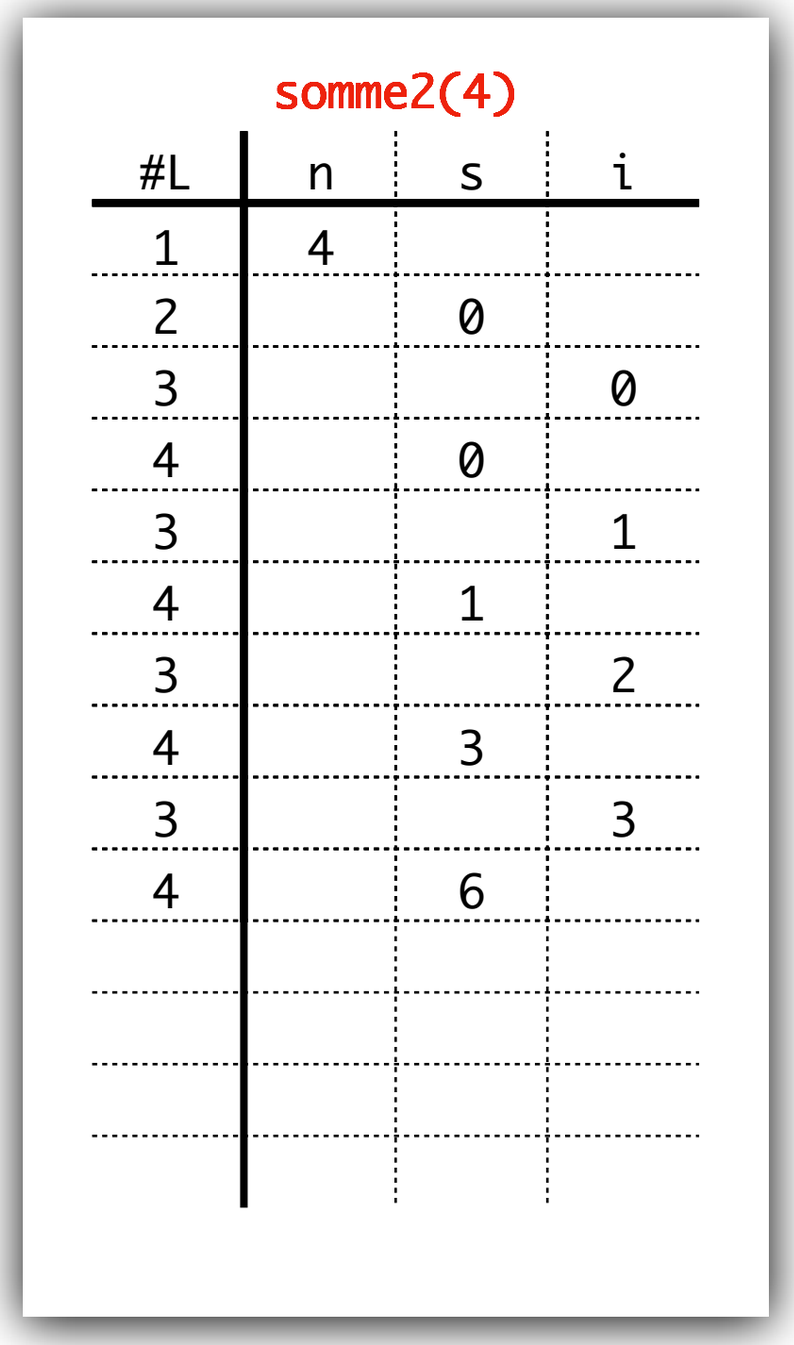
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



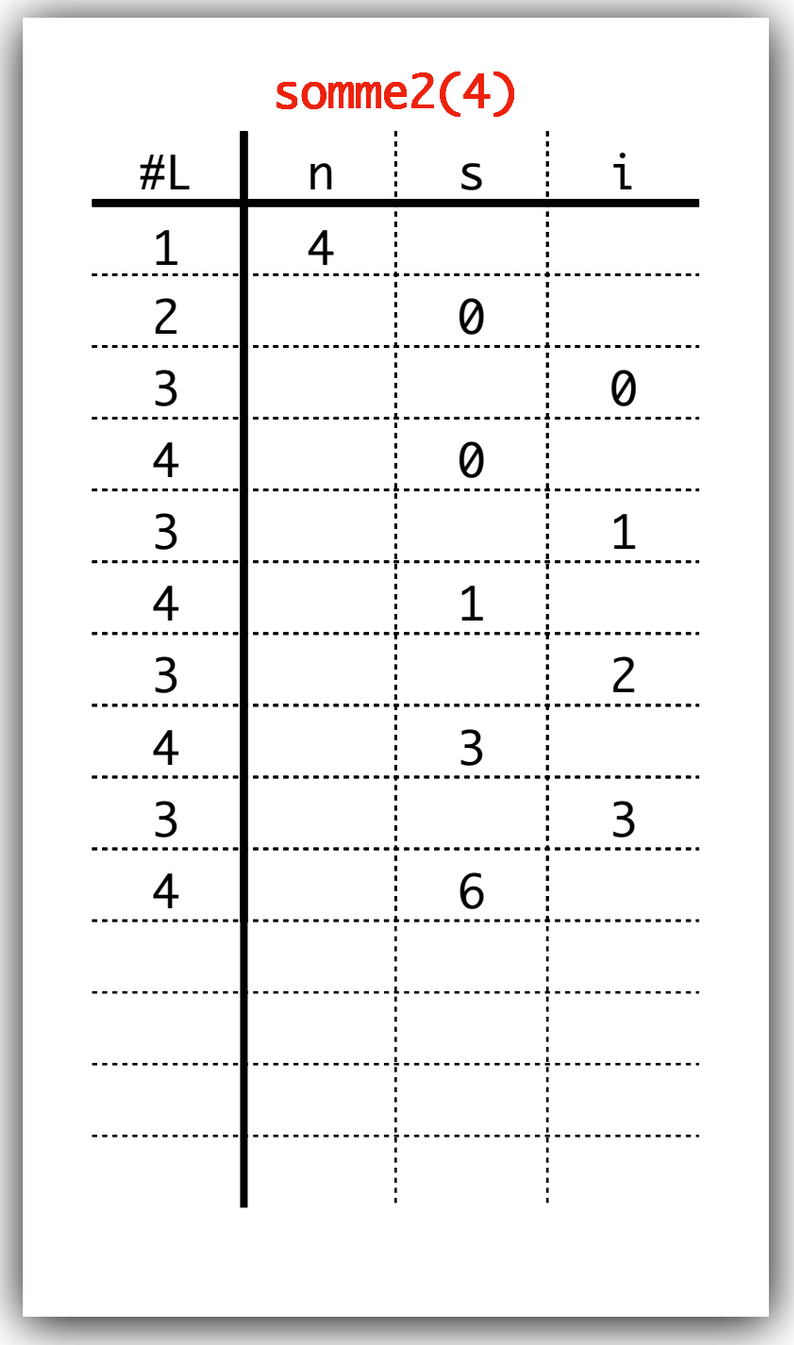
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



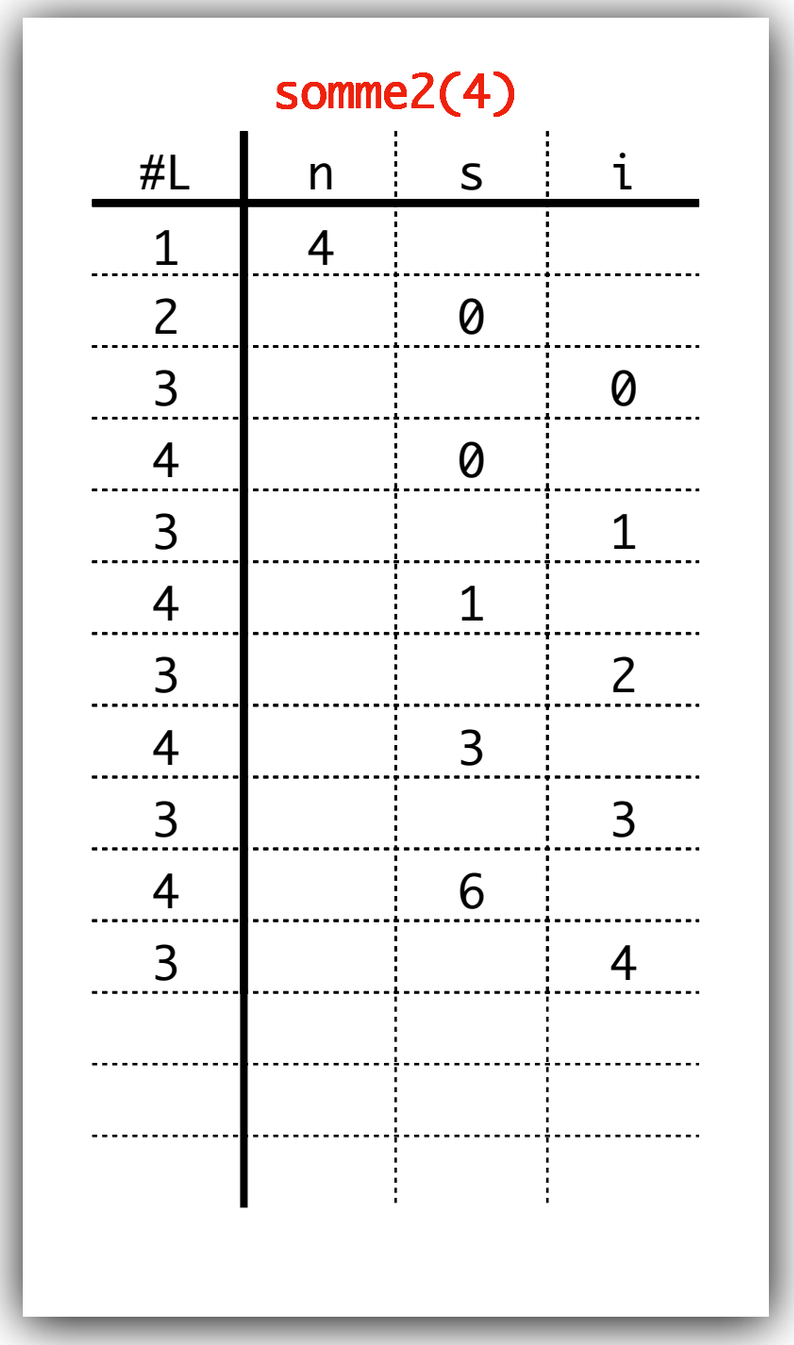
Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

Somme de 0 à 4 avec `for`

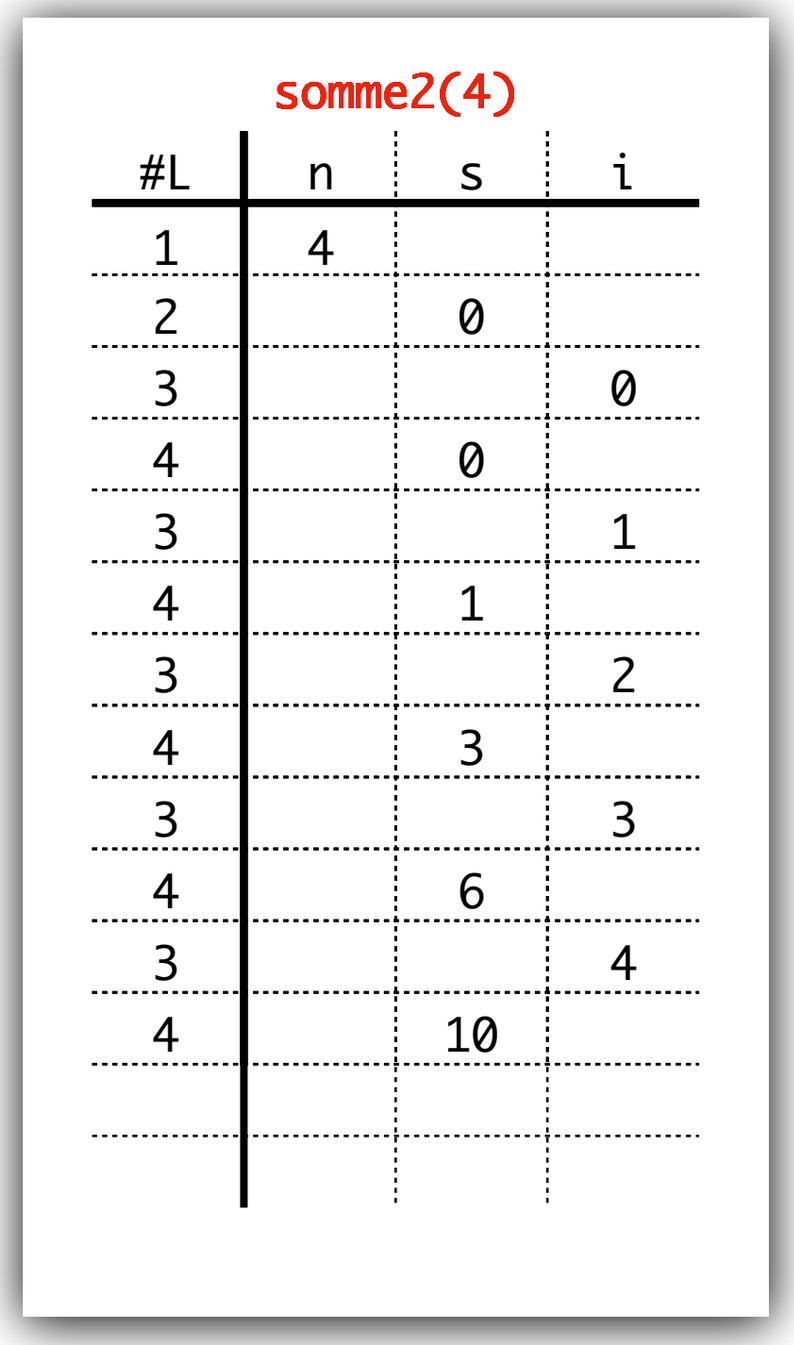
```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```


somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

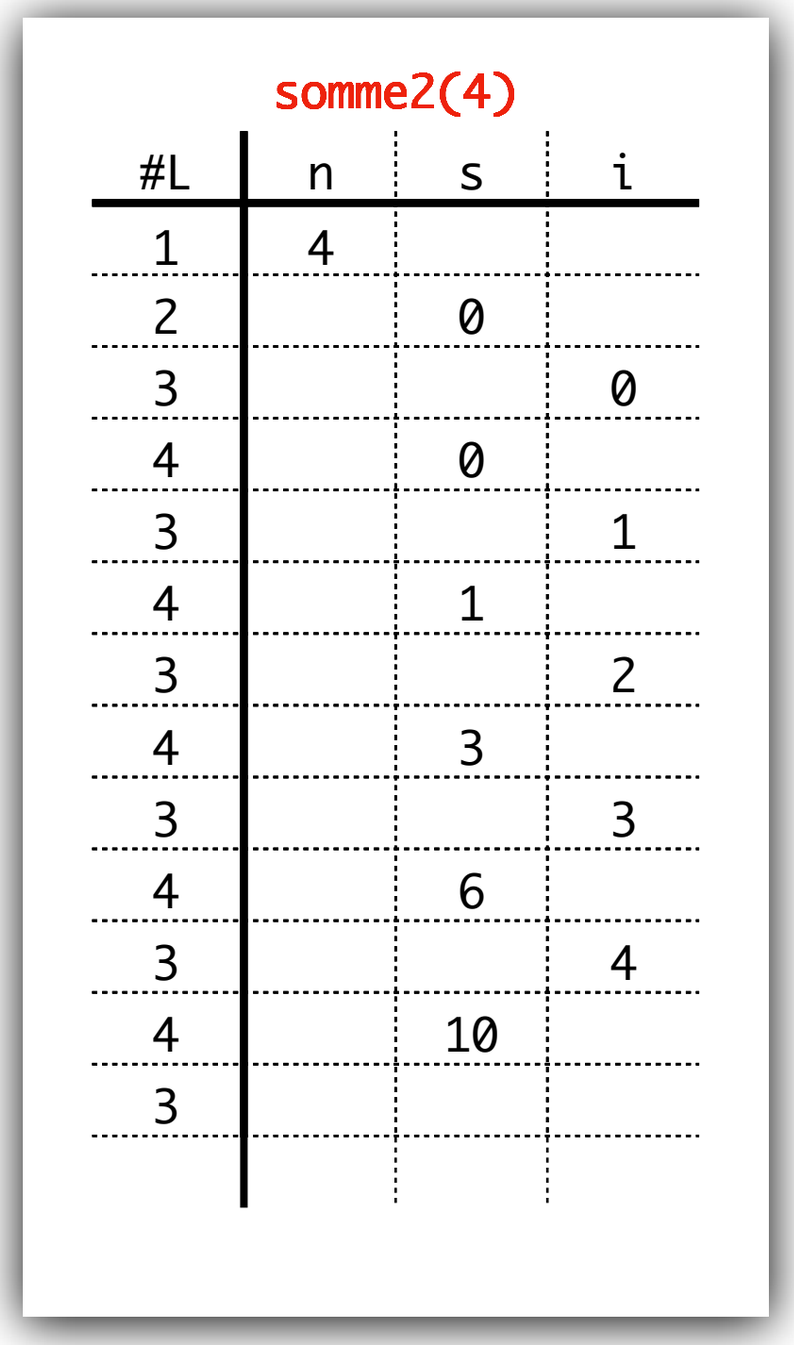


somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

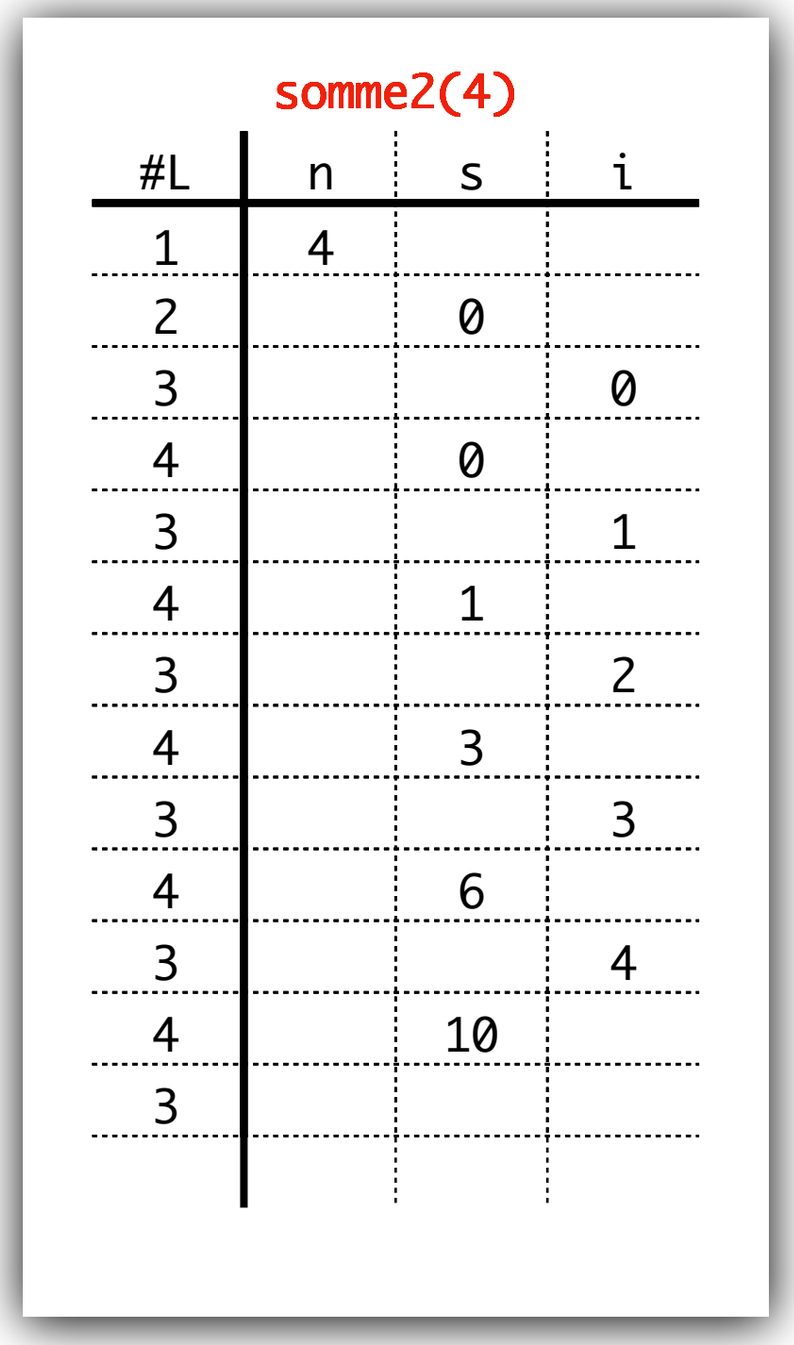


somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	
3			

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

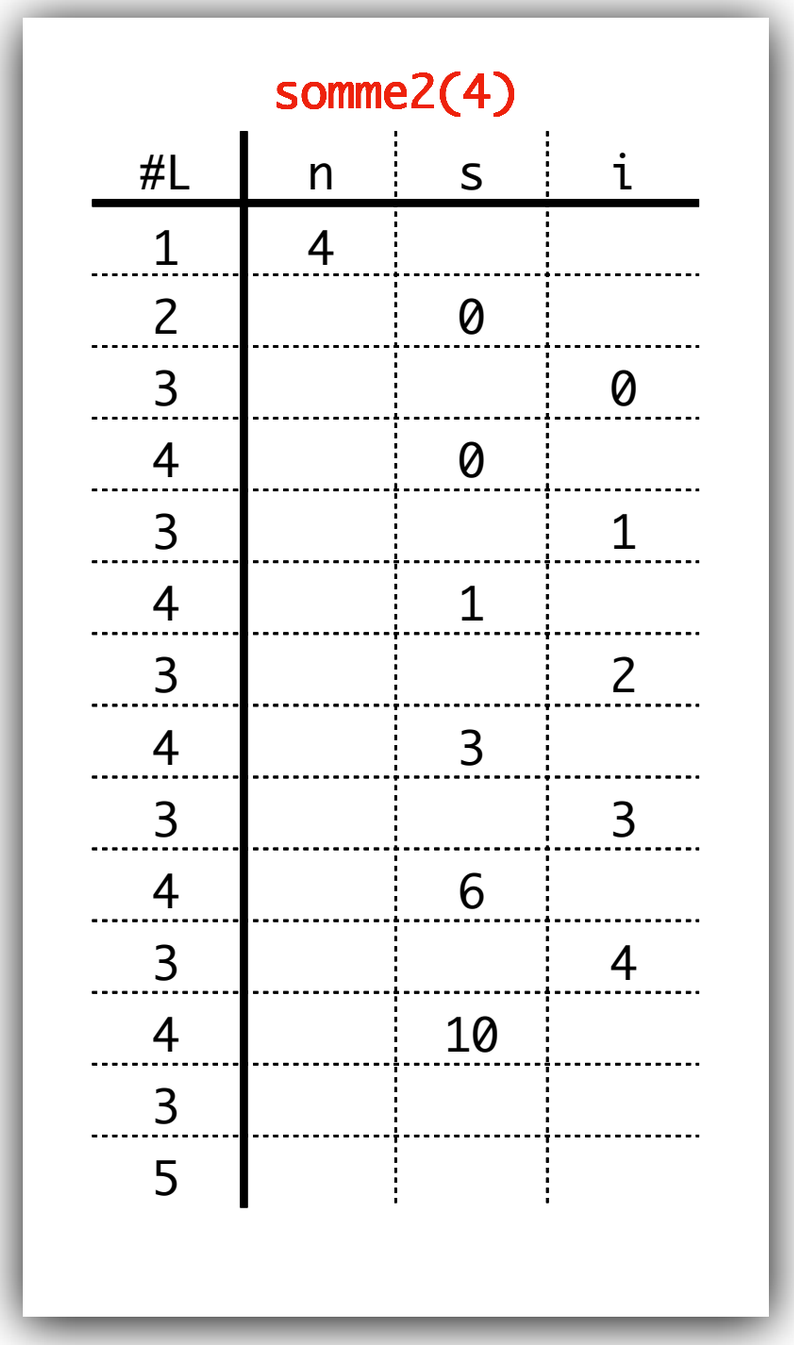


somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	
3			

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```



somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	
3			
5			

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	
3			
5			

résultat : 10

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

remarque :
`i` n'arrive
jamais à 5

somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	
3			
5			

résultat : 10

Somme de 0 à 4 avec `for`

```
1  def somme2(n):
2      s = 0
3      for i in range(n + 1):
4          s = s + i
5      return s
```

somme2(4)

#L	n	s	i
1	4		
2		0	
3			0
4		0	
3			1
4		1	
3			2
4		3	
3			3
4		6	
3			4
4		10	
3			
5			

résultat : 10

Exercice 10 du TD2

Algorithmes qui renvoient vrai ou faux

Parité des entiers naturels

```
def pair(n):
    return n % 2 == 0
```

Parité des entiers naturels

```
def pair(n):
    return n % 2 == 0
```

```
def impair(n):
    return n % 2 != 0
```

Parité des entiers naturels

```
def pair(n):
    return n % 2 == 0
```

```
def impair(n):
    return n % 2 != 0


def impair_bis(n):
    return not pair(n)
```