new idea

TRIER 4 NOMBRES

Triez les trois nombres `num1,num2,num3` et mettez-les dans les variables `t1,t2,t3`.

```
num1 = int(input("Number 1: "))
num2 = int(input("Number 2: "))
num3 = int(input("Number 3: "))

t1 = 0
t2 = 0
t3 = 0
```

Trier 3 nombres

```
num1 = int(input("Number 1: "))
num2 = int(input("Number 2: "))
num3 = int(input("Number 3: "))

t1 = 0
t2 = 0
t3 = 0

if num1 <= num2 and num1 <= num3:
    t1 = num1
    if num2 <= num3:
        t2 = num2
        t3 = num3
    else:
        t2 = num3
        t3 = num2

elif num2 <= num1 and num2 <= num3:
    t1 = num2
    if num1 <= num3:
        t2 = num1
        t3 = num3
    else:
        t2 = num3
        t3 = num1

else:
    t1 = num3
    if num2 <= num1:
        t2 = num2
        t3 = num1
    else:
        t2 = num1
        t3 = num2

print("Les 3 numeros dans l'ordre croissant sont : ", t1, t2, t3)
```

Triez les quatres nombres `num1,num2,num3,num4` et mettez-les dans les variables `t1,t2,t3,t4`.

```
num1 = int(input("Number 1: "))
num2 = int(input("Number 2: "))
num3 = int(input("Number 3: "))
num4 = int(input("Number 4: "))

t1 = 0
t2 = 0
t3 = 0
t4 = 0
```

Trier 4 nombres

```
num1 = int(input("Number 1: "))
num2 = int(input("Number 2: "))
num3 = int(input("Number 3: "))
num4 = int(input("Number 4: "))

t1 = 0
t2 = 0
t3 = 0
t4 = 0

if num1 >= num2 and num1 >= num3 and num1 >= num4:
    t4 = num1
    num1 = num4
elif num2 >= num1 and num2 >= num3 and num2 >= num4:
    t4 = num2
    num2 = num4
elif num3 >= num1 and num3 >= num2 and num3 >= num4:
    t4 = num3
    num3 = num4
elif num4 >= num1 and num4 >= num2 and num4 >= num3:
    t4 = num4
```

Le variable `t4` contient le maximum
Les trois nombres restant sont maintenant dans les variables
`num1, num2, num3`
Et maintenant ... il faut simplement trier `num1, num2, num3` ... comme avant!

Trier 5 nombres ?

1. Mettez le maximum dans le variable `t5`

2. Mettez les nombres restant dans les variables

   `num1, num2, num3, num4`

3. Et maintenant ... il faut simplement trier

   `num1, num2, num3, num4`

   ... comme avant!

new idea

SOLUTION

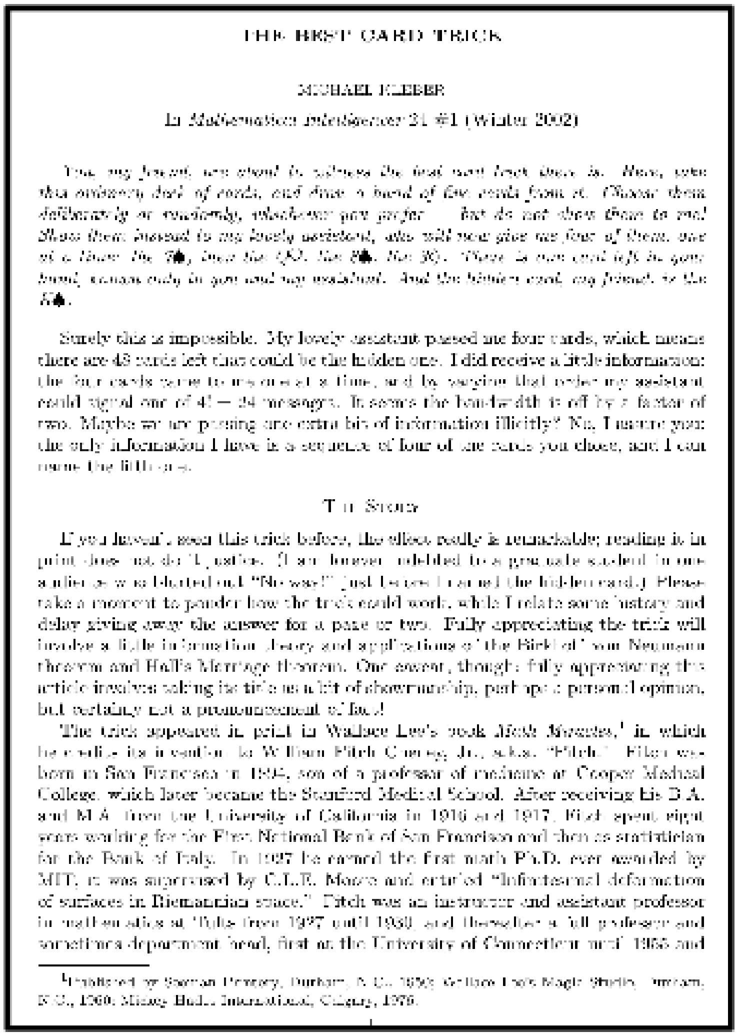
5trie

```
t1, t2, t3, t4, t5 = 0, 0, 0, 0, 0
if num1 <= num2 and num1 <= num3 and num1 <= num4 and num1 <= num5:
    t1 = num1
elif num2 <= num1 and num2 <= num3 and num2 <= num4 and num2 <= num5:
    t1 = num2
    num2 = num1
elif num3 <= num1 and num3 <= num2 and num3 <= num4 and num3 <= num5:
    t1 = num3
    num3 = num1
elif num4 <= num1 and num4 <= num2 and num4 <= num3 and num4 <= num5:
    t1 = num4
    num4 = num1
else:
    t1 = num5
    num5 = num1
if num2 <= num3 and num2 <= num4 and num2 <= num5:
    t2 = num2
elif num3 <= num2 and num3 <= num4 and num3 <= num5:
    t2 = num3
    num3 = num2
elif num4 <= num2 and num4 <= num3 and num4 <= num5:
    t2 = num4
    num4 = num2
else:
    t2 = num5
    num5 = num2
if num3 <= num4 and num3 <= num5:
    t3 = num3
elif num4 <= num3 and num4 <= num5:
    t3 = num4
    num4 = num3
else:
    t3 = num5
    num5 = num3
if num4 <= num5:
    t4 = num4
    t5 = num5
else:
    t4 = num5
    t5 = num4
print("Dans l'ordre croissant : ", t1, t2, t3, t4, t5)
```

new idea

PUZZLE :

MAGIC TRICK


THE BEST CARD TRICK

MICHAEL KLEBER

In *Mathematical Intelligencer* 24 #1 (Winter 2002)

You, my friend, are about to witness the best card trick there is. Here, take this ordinary deck of cards, and draw a hand of five cards from it. Choose them deliberately or randomly, whichever you prefer — but do not show them to me! Show them instead to my lovely assistant, who will now give me four of them: one at a time: the 7♠, then the Q♡, the 8♣, the 3♢. There is one card left in your hand, known only to you and my assistant. And the hidden card, my friend, is the K♠.

Surely this is impossible. My lovely assistant passed me four cards, which means there are 48 cards left that could be the hidden one. I did receive a little information: the four cards came to me one at a time, and by varying that order my assistant could signal one of $4! = 24$ messages. It seems the bandwidth is off by a factor of two. Maybe we are passing one extra bit of information illicitly? No, I assure you: the only information I have is a sequence of four of the cards you chose, and I can name the fifth one.

THE STORY

If you haven't seen this trick before, the effect really is remarkable; reading it in print does not do it justice. (I am forever indebted to a graduate student in one audience who blurted out "No way!" just before I named the hidden card.) Please take a moment to ponder how the trick could work, while I relate some history and delay giving away the answer for a page or two. Fully appreciating the trick will involve a little information theory and applications of the Birkhoff-von Neumann theorem and Hall's Marriage theorem. One caveat, though: fully appreciating this article involves taking its title as a bit of showmanship, perhaps a personal opinion, but certainly not a pronouncement of fact!

The trick appeared in print in Wallace Lee's book *Math Miracles*,[1] in which he credits its invention to William Fitch Cheney, Jr., a.k.a. "Fitch." Fitch was born in San Francisco in 1894, son of a professor of medicine at Cooper Medical College, which later became the Stanford Medical School. After receiving his B.A. and M.A. from the University of California in 1916 and 1917, Fitch spent eight years working for the First National Bank of San Francisco and then as statistician for the Bank of Italy. In 1927 he earned the first math Ph.D. ever awarded by MIT; it was supervised by C.L.E. Moore and entitled "Infinitesimal deformation of surfaces in Riemannian space." Fitch was an instructor and assistant professor in mathematics at Tufts from 1927 until 1930, and thereafter a full professor and sometimes department head, first at the University of Connecticut until 1955 and

[1] Published by Seeman Printery, Durham, N.C., 1950; Wallace Lee's Magic Studio, Durham, N.C., 1960; Mickey Hades International, Calgary, 1976.

1

ALGORITHME

Supposons que les cinq nombres soient stockés dans : `num1, num2, num3, num4, num5`.

Etape 1: Identifier les deux nombres situés dans le même intervalle circulaire.

indice: Les nombres étant triés, il suffit de trouver parmi les paires `(num1, num2), (num2, num3), (num3, num4) ou (num4, num5).`

Etape 2: Calculer la distance minimale (sur le cercle) entre les deux nombres trouvés à l'étape 1.

indice: la somme des distances dans les deux sens est toujours égale à 13.

Etape 3: Déterminer lequel de ces deux nombres doit être placé en premier, et lequel doit être caché.

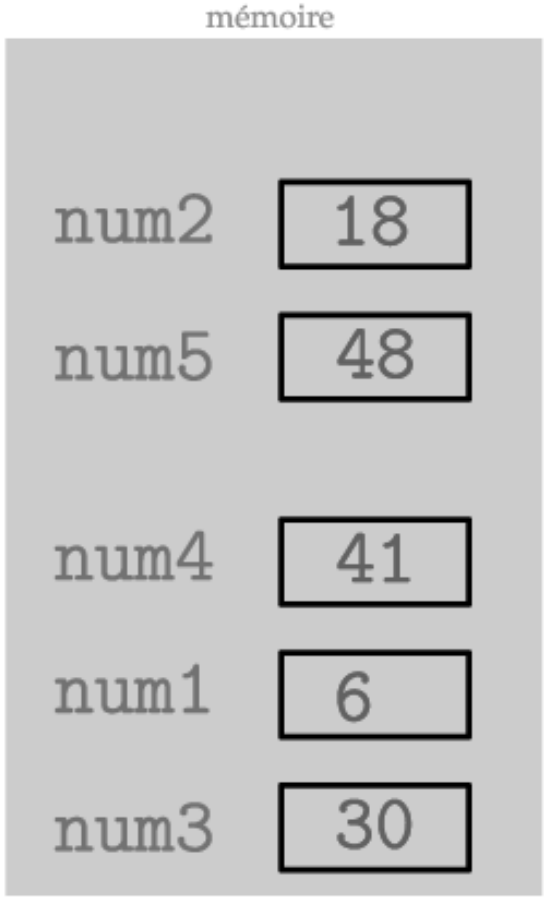
Etape 4: En fonction de la distance minimale calculée, déterminer la permutation à appliquer aux trois nombres restants.

new idea

LISTES

nous n'avons aucun contrôle sur l'emplacement dans le memoire

```
num1 = 6
num2 = 18
num3 = 30
num4 = 41
num5 = 48
```



Un nouveau type de variable : **liste**

Une liste est une structure de données qui contient une série de valeurs.

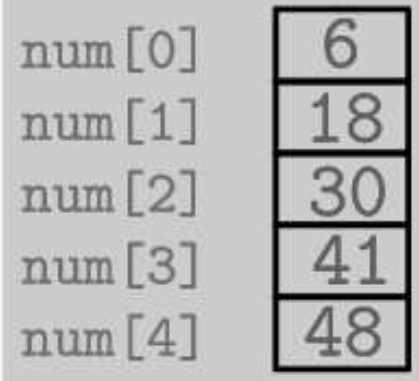
indice de la liste

```
          0    1    2    3    4
num = [6, 18, 30, 41, 48]
print(num)
print( type(num) )
```

↓

```
[6, 18, 30, 41, 48]
<class 'list'>
```

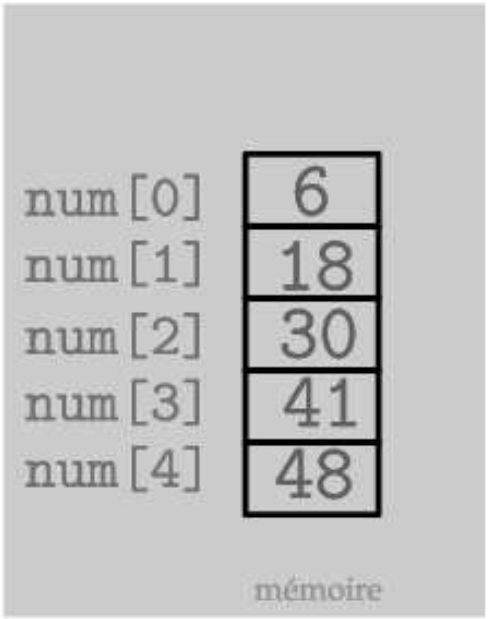
mémoire

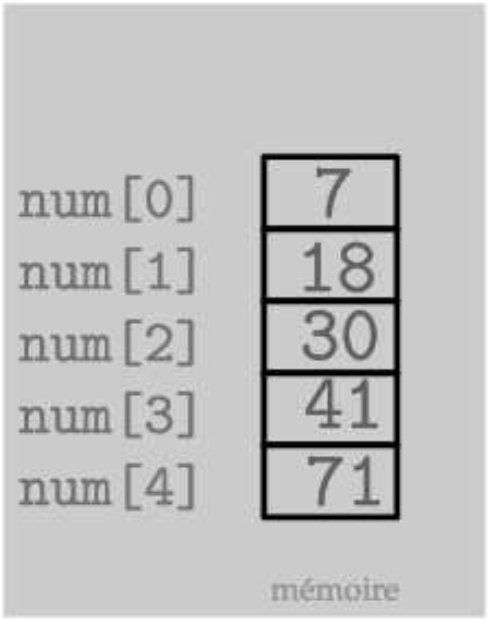



```
num = [6, 18, 30, 41, 48]
print("first num: ", num)

num[0] = num[0] + 1
num[4] = num[2] + num[3]
print("second num: ", num)
```

→

```
first num:  [6, 18, 30, 41, 48]
second num:  [7, 18, 30, 41, 71]
```





Une liste peut contenir plusieurs types

```
A = [4, 8.323, 'hello', "again"]
print(type(A))
print(type(A[0]))
print(type(A[1]))
print(type(A[2]))
print(type(A[3]))
```

```
<class 'list'>
<class 'int'>
<class 'float'>
<class 'str'>
<class 'str'>
```

Une liste peut contenir plusieurs types

```
A = [4, 8.323, 'hello']
print(A)
print( type(A) )
```

```
[4, 8.323, 'hello']
<class 'list'>
```

une liste vide

```
A = [ ]
print(A)
print(type(A))
```

```
[]
<class 'list'>
```

```
r = 0
print(r, type(r))

r = [0]
print(r, type(r))
print(r[0], type(r[0]))
```

```
0 <class 'int'>
[0] <class 'list'>
0 <class 'int'>
```

Facile de vérifier si un élément est dans la liste, avec mot-clé in :

```
A = [4, 8.323, 'hello', "again"]

if 'hello' in A:
    print("Found it!")
else:
    print("Not found!")
```

```
Found it!
```

Facile de vérifier si un élément est dans la liste, avec mot-clé in :

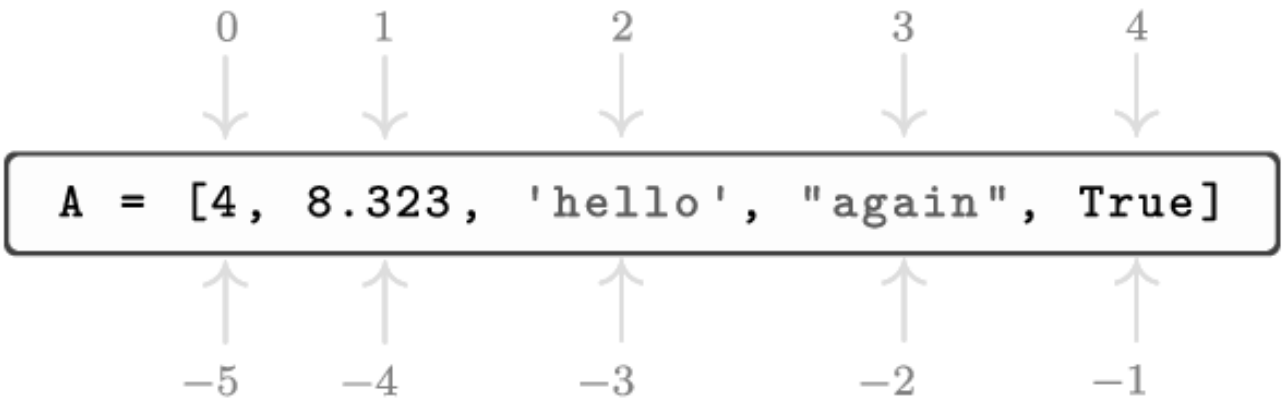
```
A = [4, 8.323, 'hello', "again"]

if 'hello' not in A:
    print("Found it!")
else:
    print("Not found!")
```

```
Not found!
```

new idea

LISTE INDICES

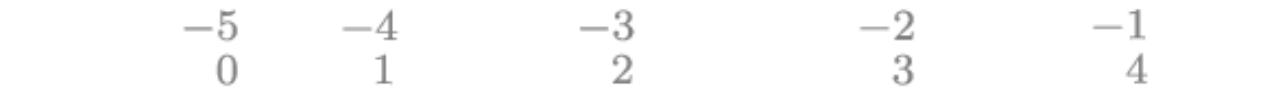
indice de la liste



Leur principal avantage :

vous pouvez accéder au dernier élément d'une liste à l'aide de l'indice -1 sans pour autant connaître la longueur de cette liste.

indice de la liste




```
A = [4, 8.323, 'hello', "again", True]
print( A[1] )
print( A[-2])
print( A[4] )
print( A[-1] )
```

```
8.323
again
True
True
```

tranche

−5	−4	−3	−2	−1
0	1	2	3	4

```
A = [4, 8.323, 'hello', "again", True]
print( A[1:3] )
print( type( A[1:3] ) )
```

```
[8.323, 'hello']
<class 'list'>
```

`A[a:b]` = `[ A[a], A[a+1], ..., A[b-1] ]`

une liste

tranche

−5	−4	−3	−2	−1
0	1	2	3	4

```
A = [4, 8.323, 'hello', "again", True]
print(A[1:])
print(A[:3])
```

```
[8.323, 'hello', 'again', True]
[4, 8.323, 'hello']
```

−5	−4	−3	−2	−1
0	1	2	3	4

```
A = [4, 8.323, 'hello', "again", True]
print( A[1:3] )
print( A[0:0] )
print( A[1:-1] )
print( A[-1:1] )
```

```
[8.323, 'hello']
[]
[8.323, 'hello', 'again']
[]
```

0	1	2	3	4	5	6

```
A = [3, 43, 'hello', 'goodbye', 3.413, 89, -3]

print( A[7] )
```

```
Traceback (most recent call last):
  File "/home/nabil/abc.py", line 4, in <module>
    print( A[7] )
IndexError: list index out of range
```

```
A = [ [1,2,3], [33,"hi", "def"], [3.41, "abc", 34] ]
```

A ⟶ liste

premier élément de A : A[0] ⟶ liste

deuxiéme élément de A : A[1] ⟶ liste

troisiéme élément de A : A[2] ⟶ liste

print(A[2]) → `[3.41, 'abc', 34]`

print(A[1][1]) → `hi`

print(A[0][1]) → `2`

new idea

LISTE OPÉRATIONS

Pour calculer la taille d'une liste : `len`

```
A = [4, 8.323]
print(A)
print(len(A))
```

```
[4, 8.323]
2
```

```
A = [4, "is", "even"]
print(A)
print(len(A))
```

```
[4, 'is', 'even']
3
```

Pour calculer la taille d'une liste : `len`

```
A = [4, 8.323, 'hello', "again"]
print(A)
print(len(A))
```

```
[4, 8.323, 'hello', 'again']
4
```

Pour ajouter un élément à la fin d'une liste : append

```
A = [4, 8.323, 'hello', "again"]
A.append("and again")
print(A)
```

```
[4, 8.323, 'hello', 'again', 'and again']
```

Pour ajouter un élément à la fin d'une liste : append

```
A = [ ]
print( type(A) )
A.append("and again")
A.append("and twice")
A.append("and thrice")
print(A)
```

```
<class 'list'>
['and again', 'and twice', 'and thrice']
```

Pour ajouter un élément à l'indice i : `insert`

```
A = [42, 62, 87, "hello"]

A.insert(3, "and again")

print(A)
```

```
[42, 62, 87, 'and again', 'hello']
```

Pour supprimer un élément à l'indice i: `pop`

```
      0   1       2          3       4       5
A = [4, 63, "hello", 4.134, True, "bye"]

print(A)

x = A.pop(4)

print(A)
print(x)
```

```
[4, 63, 'hello', 4.134, True, 'bye']
[4, 63, 'hello', 4.134, 'bye']
True
```

Pour supprimer un élément à l'indice i: `pop`

```
    0   1       2           3       4       5
A = [4, 63, "hello", 4.134, True, "bye"]

print(A)

A.pop()

print(A)
```

```
[4, 63, 'hello', 4.134, True, 'bye']
[4, 63, 'hello', 4.134, True]
```

Pour supprimer une tranche des éléments : `del`

```
    0   1       2           3       4       5
A = [4, 63, "hello", 4.134, True, "bye"]

print(A)

del A[2:4]

print(A)
```

```
[4, 63, 'hello', 4.134, True, 'bye']
[4, 63, True, 'bye']
```

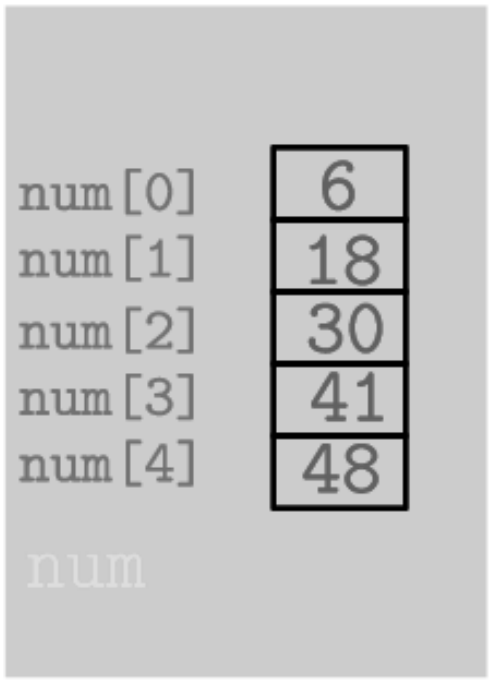
new idea

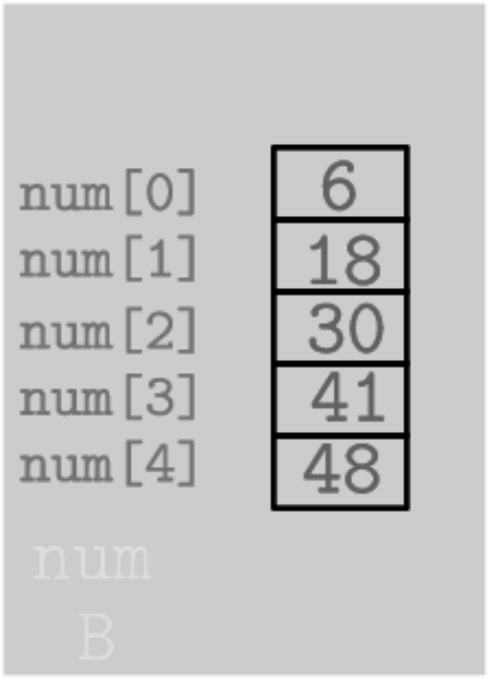
LISTE: OPÉRATEURS =, ==

```
num = [6, 18, 30, 41, 48]
B = num
```

Un autre nom de la **même** liste !

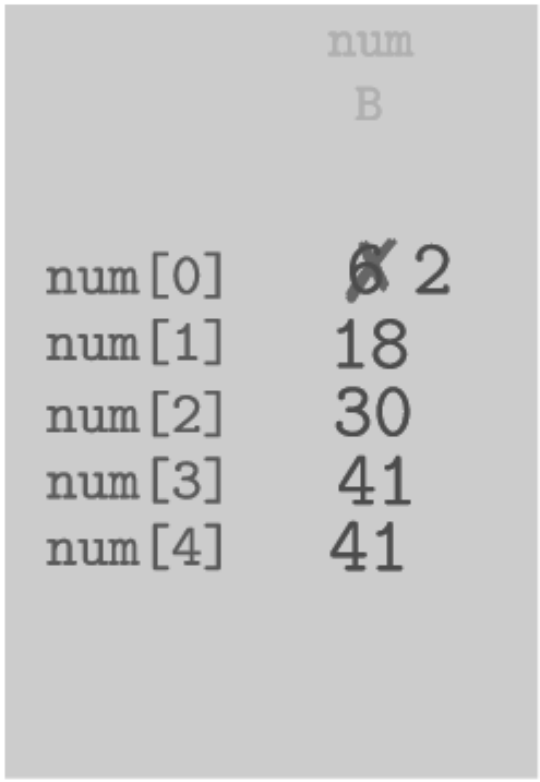
ATTENTION!





```
num = [6, 18, 30, 41, 48]
B = num
print("first B is: ", B)

num[0] = 2
print("second B is: ", B)
```

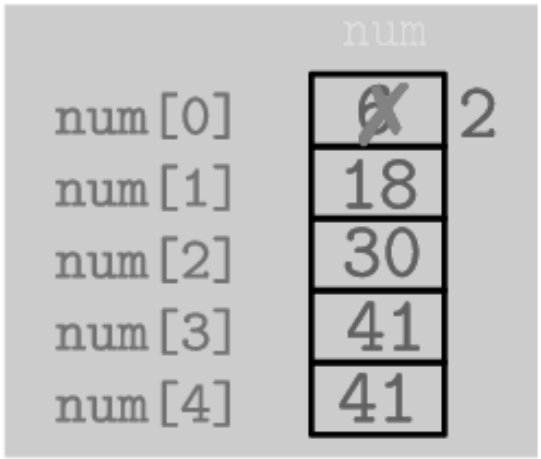
```
first B is:  [6, 18, 30, 41, 48]
second B is:  [2, 18, 30, 41, 48]
```

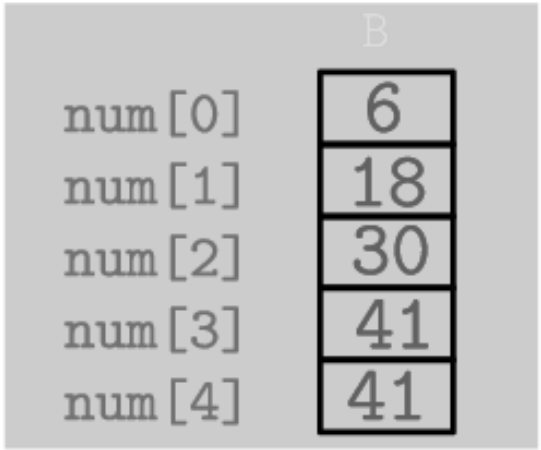


Pour faire une copie distincte, utilisez `list()`

```
num = [6, 18, 30, 41, 48]
B = list(num)
print("first B is: ", B)

num[0] = 2
print("second B is: ", B)
```

```
first B is:  [6, 18, 30, 41, 48]
second B is:  [6, 18, 30, 41, 48]
```





Pour vérifier si deux listes sont égales :

```
A = [1, 5]
B = [1, 5]

if A == B:
    print("Equal!")
else:
    print("No!")
```

Equal!

```
A = [1, 5]
B = [5, 1]

if A == B:
    print("Equal!")
else:
    print("No!")
```

No!

Attention: L'ordre est important

new idea

LISTE INITIALISATION

Initialisez une liste avec de plusieurs éléments :

```
A = [0] * 10
print(type(A))
print(A)
```



```
<class 'list'>
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
A = ['hello'] * 4
print(type(A))
print(A)
```

```
<class 'list'>
['hello', 'hello', 'hello', 'hello']
```

```
A = [ ] * 10
print(type(A))
print(A)
```

```
<class 'list'>
[]
```

Initialisez une liste avec de plusieurs éléments :

```
A = [ ['hello'] ] * 4
print(type(A))
print(A)
```

```
<class 'list'>
[['hello'], ['hello'], ['hello'], ['hello']]
```

Initialisez une liste avec de plusieurs éléments :

```
A = [2] * 4
print(A)

A[2] = 9
print(A)
```

```
[2,2,2,2]
[2,2,9,2]
```

```
A = [0,0,0,0]

x = 2
A[0] = x
A[1] = x
A[2] = x
A[3] = x

print(A)

A[2] = 9
print(A)
```

```
[2,2,2,2]
[2,2,9,2]
```

Initialisez une liste avec de plusieurs éléments :

```
A = [ ['a'] ] * 3

print(A)

A[0].append('b')

print(A)
```

ATTENTION!

```
[['a'], ['a'], ['a']]
[['a', 'b'], ['a', 'b'], ['a', 'b']]
```

Why?

```
A = [0,0,0]

x = ['a']
A[0] = x
A[1] = x
A[2] = x

print(A)

A[0].append('b')

print(A)
```

Un autre nom de la **même** liste !

```
[['a'], ['a'], ['a']]
[['a', 'b'], ['a', 'b'], ['a', 'b']]
```

Initialisez une liste avec de plusieurs éléments :

```
A = [ ['hi']*3 ] * 2
print(type(A))
print(A)

A[0][0] = 'hello'
print(A)
```

```
<class 'list'>
[['hi', 'hi', 'hi'], ['hi', 'hi', 'hi']]
[['hello', 'hi', 'hi'], ['hello', 'hi', 'hi']]
```

Attention: c'est la même liste !