new idea

SOLUTION

Magic Trick Encodage

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

Pour vérifier si `num[1]` et `num[2]` appartiennent au premier intervalle, on peut utiliser :

```
if (num[1] >= 1 and num[1] <= 13 and num[2] >=1 and num[2] <= 13)
```

En utilisant une logique similaire, vous pouvez vérifier si `num[1]` et `num[2]` appartiennent au deuxième intervalle, ou au troisième intervalle, ou au quatrième intervalle.
Vous pouvez vérifier n'importe quelle paire de nombres de cette façon.

Étape 2 : Calculer la distance minimale sur le cercle.

Soient x et y les deux nombres trouvés à l'étape 1, avec $x < y$.
Distance directe : $d1 = y - x$. **Distance inverse :** $d2 = \cdots$?
Distance minimale : $dmin = \min(d1, d2)$.

Étape 3 : Déterminer l'ordre d'émission.

Si $dmin = d1$, alors on émet d'abord x (en position 1), puis on cache y en 5-éme.
Si $dmin = d2$, alors on émet d'abord y (en position 1), puis on cache x en 5-éme.

Étape 4 : Permutation des trois nombres restants.

Après avoir choisi quel nombre cacher, il reste trois nombres à transmettre.
Maintenant, déterminez l'ordre de ces trois nombres aux positions 2, 3 et 4 en consultant la table d'encodage fournie précédemment, pour trouver la permutation correspondant à la distance $dmin$.

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

Pour vérifier si `num[1]` et `num[2]` appartiennent au premier intervalle, on peut utiliser :

```
if (num[1] >= 1 and num[1] <= 13 and num[2] >=1 and num[2] <= 13)
```

En utilisant une logique similaire, vous pouvez vérifier si `num[1]` et `num[2]` appartiennent au deuxième intervalle, ou au troisième intervalle, ou au quatrième intervalle.
Vous pouvez vérifier n'importe quelle paire de nombres de cette façon.

```
x, y = 0, 0
a,b,c = 0, 0, 0

if ((num[0] >= 1  and num[0] <= 13 and num[1] >= 1  and num[1] <= 13) or
    (num[0] >= 14 and num[0] <= 26 and num[1] >= 14 and num[1] <= 26) or
    (num[0] >= 27 and num[0] <= 39 and num[1] >= 27 and num[1] <= 39) or
    (num[0] >= 40 and num[0] <= 52 and num[1] >= 40 and num[1] <= 52)):
       x,y = num[0], num[1]
       a,b,c = num[2], num[3], num[4]

elif ((num[1] >= 1  and num[1] <= 13 and num[2] >= 1  and num[2] <= 13) or
      (num[1] >= 14 and num[1] <= 26 and num[2] >= 14 and num[2] <= 26) or
      (num[1] >= 27 and num[1] <= 39 and num[2] >= 27 and num[2] <= 39) or
      (num[1] >= 40 and num[1] <= 52 and num[2] >= 40 and num[2] <= 52)):
       x,y = num[1], num[2]
       a,b,c = num[0], num[3], num[4]

elif ((num[2] >= 1  and num[2] <= 13 and num[3] >= 1  and num[3] <= 13) or
      (num[2] >= 14 and num[2] <= 26 and num[3] >= 14 and num[3] <= 26) or
      (num[2] >= 27 and num[2] <= 39 and num[3] >= 27 and num[3] <= 39) or
      (num[2] >= 40 and num[2] <= 52 and num[3] >= 40 and num[3] <= 52)):
       x,y = num[2], num[3]
       a,b,c = num[0], num[1], num[4]

elif ((num[3] >= 1  and num[3] <= 13 and num[4] >= 1  and num[4] <= 13) or
      (num[3] >= 14 and num[3] <= 26 and num[4] >= 14 and num[4] <= 26) or
      (num[3] >= 27 and num[3] <= 39 and num[4] >= 27 and num[4] <= 39) or
      (num[3] >= 40 and num[3] <= 52 and num[4] >= 40 and num[4] <= 52)):
       x,y = num[3], num[4]
       a,b,c = num[0], num[1], num[2]
```

Étape 2 : Calculer la distance minimale sur le cercle.

Soient x et y les deux nombres trouvés à l'étape 1, avec $x < y$.

Distance directe : $d1 = y - x$. **Distance inverse :** $d2 = \cdots$?

Distance minimale : $dmin = \min(d1, d2)$.

Étape 3 : Déterminer l'ordre d'émission.

Si $dmin = d1$, alors on émet d'abord x (en position 1), puis on cache y en 5-éme.

Si $dmin = d2$, alors on émet d'abord y (en position 1), puis on cache x en 5-éme.

```
d1 = y - x
d2 = 13 - d1

if d1 <= 6:
    dmin = d1
    output[0], output[4] = x, y
else:
    dmin = d2
    output[0], output[4] = y, x
```

Étape 4 : Permutation des trois nombres restants.

Après avoir choisi quel nombre cacher, il reste trois nombres à transmettre.
Maintenant, déterminez l'ordre de ces trois nombres aux positions 2, 3 et 4 en consultant la table d'encodage fournie précédemment, pour trouver la permutation correspondant à la distance $dmin$.

```
if dmin == 1:
    output[1], output[2], output[3] = a, b, c
elif dmin == 2:
    output[1], output[2], output[3] = a, c, b
elif dmin == 3:
    output[1], output[2], output[3] = b, a, c
elif dmin == 4:
    output[1], output[2], output[3] = b, c, a
elif dmin == 5:
    output[1], output[2], output[3] = c, a, b
elif dmin == 6:
    output[1], output[2], output[3] = c, b, a
```

Premièrement : Rendez-le simple, même si cela devient long !

Ensuite : Une fois que cela fonctionne, essayer de l'améliorer.

```
x, y = 0, 0
a,b,c = 0, 0, 0

i, j = 0, 1
if ((num[i] >= 1  and num[i] <= 13 and num[j] >= 1  and num[j] <= 13) or
    (num[i] >= 14 and num[i] <= 26 and num[j] >= 14 and num[j] <= 26) or
    (num[i] >= 27 and num[i] <= 39 and num[j] >= 27 and num[j] <= 39) or
    (num[i] >= 40 and num[i] <= 52 and num[j] >= 40 and num[j] <= 52)):
       x,y = num[0], num[1]
       a,b,c = num[2], num[3], num[4]

i, j = 1, 2
if ((num[i] >= 1  and num[i] <= 13 and num[j] >= 1  and num[j] <= 13) or
    (num[i] >= 14 and num[i] <= 26 and num[j] >= 14 and num[j] <= 26) or
    (num[i] >= 27 and num[i] <= 39 and num[j] >= 27 and num[j] <= 39) or
    (num[i] >= 40 and num[i] <= 52 and num[j] >= 40 and num[j] <= 52)):
       x,y = num[1], num[2]
       a,b,c = num[0], num[3], num[4]

  .
  .
  .
```

Next: Avec un peu de mathématiques, rendez-le encore plus simple !

```
x, y = 0, 0
a,b,c = 0, 0, 0

if int( (num[0]-1) / 13 ) == int( (num[1]-1) / 13 ):
    x, y = num[0], num[1]
    a, b, c = num[2], num[3], num[4]

if int( (num[1]-1) / 13 ) == int( (num[2]-1) / 13 ):
    x, y = num[1], num[2]
    a, b, c = num[0], num[3], num[4]

if int( (num[2]-1) / 13 ) == int( (num[3]-1) / 13 ):
    x, y = num[2], num[3]
    a, b, c = num[0], num[1], num[4]

if int( (num[3]-1) / 13 ) == int( (num[4]-1) / 13 ):
    x, y = num[3], num[4]
    a, b, c = num[0], num[1], num[2]
```

new idea

Conseils

Maîtrisez d'abord l'algorithme

Prenez le temps de bien comprendre chaque étape avant d'écrire une seule ligne de code. Si un point reste flou, revenez-y, dessinez un schéma, ou expliquez-le à voix haute. Un algorithme clair dans votre tête se traduit par un code clair à l'écran.

Divisez pour mieux régner

Décomposez le problème en sous-tâches simples et gérables. Par exemple :
Lecture des entrées—Traitement principal—Affichage des résultats. Cela rend le travail moins intimidant et facilite le débogage.

Validez étape par étape

Ne passez pas à la tâche $t + 1$ avant d'avoir vérifié que la tâche t fonctionne correctement. Testez avec des exemples simples, puis des cas limites. Cette habitude vous évitera des erreurs en cascade et vous fera gagner beaucoup de temps.

Un code qui évolue pas à pas, c'est comme un puzzle qu'on assemble pièce par pièce. Chaque étape validée est une petite victoire qui vous rapproche de la solution finale.

new idea

STRING

Un nouveau type de variable pour le texte : **str**ing

```
r = "hello"
print(r)
print( type(r) )
```

→

```
hello
<class 'str'>
```

```
r = 'hello'
print(r)
```

→

```
hello
```

```
r = "hello"
print(len(r))
```

→

```
5
```

Pour accéder au i-ème caractère d'une string :

```
r = "hello"
s = r[1]
print(type(s))
print(s)
```

→

```
<class 'str'>
e
```

```
r = "hello"
s = r[1:4]
print(type(s))
print(s)
```

→

```
<class 'str'>
ell
```

```
a = "hello"
b = 'hello'

if a == b:
    print("They are equal!")
else:
    print("They are different!")
```

↓

```
They are equal!
```

```
a = "hello"
b = "Hello"

if a == b:
    print("They are equal!")
else:
    print("They are different!")
```

```
They are different!
```

```
r = "hello'
print(r)
```

```
  File "/home/nabil/abc.py", line 1
    r = "hello'
        ^
SyntaxError: unterminated string literal (detected at line 1)
```

Chaque caractère est associée à une valeur entière.
Chaque entier compris entre 1 et 127 est associé à un caractère.

```
s = "a"
x = ord(s)
print(x)
```

97

```
x = 97
s = chr(x)
print(s)
```

a

Dec	Hex	Chr
0	00	NUL
1	01	SOH
2	02	STX
3	03	ETX
4	04	EOT
5	05	ENQ
6	06	ACK
7	07	BEL
8	08	BS
9	09	HT
10	0A	LF
11	0B	VT
12	0C	FF
13	0D	CR
14	0E	SO
15	0F	SI
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

Dec	Hex	Chr
32	20	Space
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Chr
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Chr
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL

Attention : nous ne pouvons pas modifier les `strings` !

```
s = "hello"
print(s)

s[1] = 'i'
```

```
hello
Traceback (most recent call last):
  File "/home/nabil/tmp/t.py", line 4, in <module>
    s[1] = 'i'
    ~~~~
TypeError: 'str' object does not support item assignment
```

new idea

C vs Python

Une variable peut contenir différents types de valeurs

```
r = 4
print (r, type(r) )

r = 4.331
print(r, type(r) )

r = "hello"
print(r, type(r) )

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

```
4 <class 'int'>
4.331 <class 'float'>
hello <class 'str'>
True <class 'bool'>
```

C

```
#include <stdio.h>

int main()
{
    int a = 4;
    printf("%d\n", a);

    //commentaire d'une ligne

    char *t = "abc";
    printf("%s\n", t);

    /*commentaires de
    plusieurs lignes */

    return 0;
}
```

```
4
abc
```

Python

```
a = 4
print(a)

#commentaire d'une ligne

t = "abc"
print(t)

"""commentaires de
plusieurs lignes"""
```

```
4
abc
```

Les deux font la même chose

new idea

STRINGS

fonctions

split: partitionner une `string` en morceaux

```
r = "this is a string, yes it is a string."
A = r.split()
print(type(A))
print(A)
```

↓ par défault : coupé par des espaces

```
<class 'list'>
['this', 'is', 'a', 'string,', 'yes', 'it', 'is', 'a', 'string.']
```

split: partitionner une string en morceaux

```
r = "this is a string, yes it is a string."
A = r.split(",")
print(type(A))
print(A)
```

```
<class 'list'>
['this is a string', ' yes it is a string.']
```

Créez une string à partir de variables en utilisant f :

```
a = 5.0
b = "what"
c = True
s = f"this is {a}, and {b} not, that is {c}"
print(type(s))
print(s)
```

```
<class 'str'>
this is 5.0, and what not, that is True
```

Créez une `string` à partir de variables en utilisant **f** :

```
a = 5.0

print("a is: a")
print(f"a is: {a}")
```

```
a is: a
a is: 5.0
```

joindre des éléments d'une liste avec join

```
A = [ 'ab', 'ty', 'zas' ]

s = ''.join(A)

print(s)
```

```
abtyzas
```

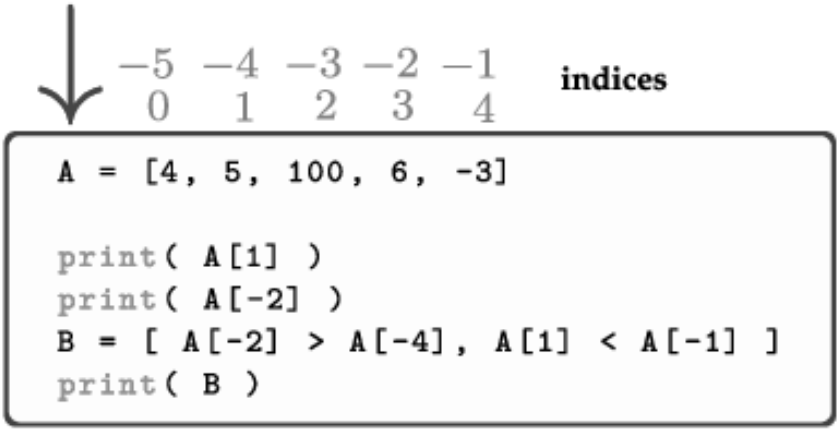
```
A = [ 'ab', 'ty', 'zas' ]

s = 'WT'.join(A)

print(s)
```

```
abWTtyWTzas
```

new idea

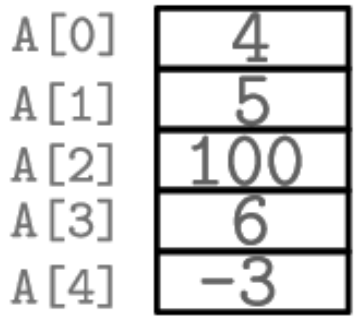
BOUCLES

liste, de taille 5




```
A = [4, 5, 100, 6, -3]

print( A[1] )
print( A[-2] )
B = [ A[-2] > A[-4], A[1] < A[-1] ]
print( B )
```

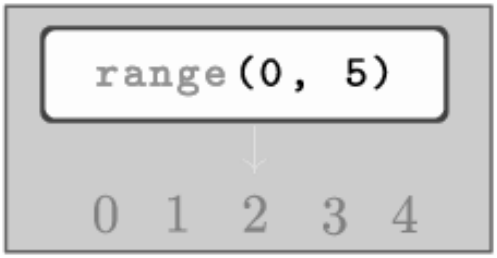
```
5
6
[True, False]
```

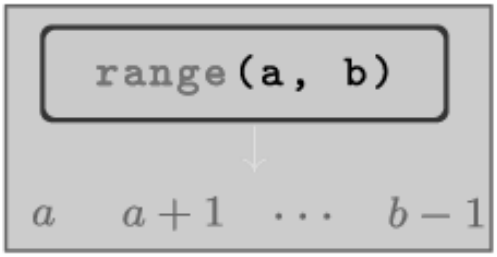


```
A = [4, 5, 100, 6, -3]

for i in range(0, 5):
    print( A[i] )
```

les deux-points et l'indentation sont formellement requis en Python.

```
4
5
100
6
-3
```






```
    0   1   2   3   4
A = [4, 5, 100, 6, -3]

for i in range(2, 4):
    print( A[i] )
```

```
100
6
```

Les deux sont équivalents!

```
A = [4, 5, 100, 6, -3]

for i in range(0, 5):
    print( A[i] )
```

```
4
5
100
6
-3
```

```
A = [4, 5, 100, 6, -3]

i = 0
print( A[i] )

i = 1
print( A[i] )

i = 2
print( A[i] )

i = 3
print( A[i] )

i = 4
print( A[i] )
```

```
A = []
for i in range(0, 5):
    A.append("hello")
print(A)
```

```
['hello', 'hello', 'hello', 'hello', 'hello']
```

```
A = []
for i in range(0, 5):
    A.append(["hello"])
print(A)
```

```
[['hello'], ['hello'], ['hello'], ['hello'], ['hello']]
```

```
for i in range(5, 7):
    for j in range(0, 3):
        print(j)
```

```
0
1
2
0
1
2
```

i = 5

$j = 0$ print(0)

$j = 1$ print(1)

$j = 2$ print(2)

i = 6

$j = 0$ print(0)

$j = 1$ print(1)

$j = 2$ print(2)

Les deux sont équivalents!

```
for i in range(0, 5):
    instructions à exécuter
```

```
i = 0
instructions à exécuter

i = 1
instructions à exécuter

i = 2
instructions à exécuter

i = 3
instructions à exécuter

i = 4
instructions à exécuter
```

```
A = [4, 5, 10, 6, -3]

for i in range(1, 5):
    A[i] = A[i] + A[i-1]
    A[i-1] = A[i] + A[i-1]
print(A)
```

→ `[13, 28, 44, 47, 22]`

```
i = 1
    A[1] = A[1] + A[0]
    A[0] = A[1] + A[0]
i = 2
    A[2] = A[2] + A[1]
    A[1] = A[2] + A[1]
i = 3
    A[3] = A[3] + A[2]
    A[2] = A[3] + A[2]
i = 4
    A[4] = A[4] + A[3]
    A[3] = A[4] + A[3]
```

A[0] 4 → 13

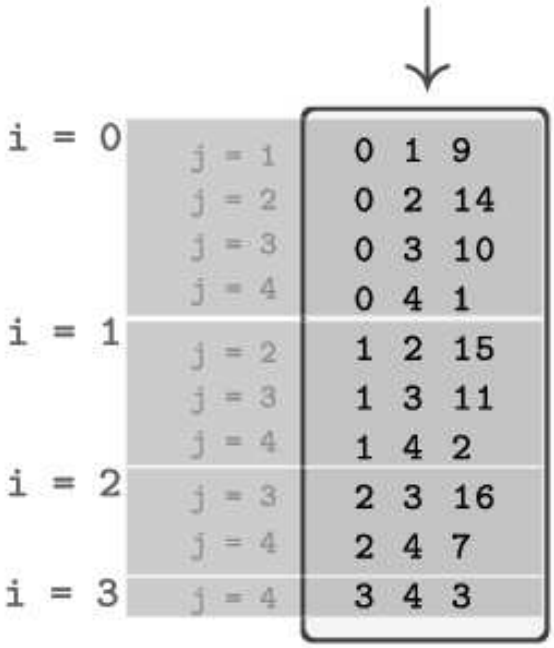
A[1] 5 → 9 → 28

A[2] 10 → 19 → 44

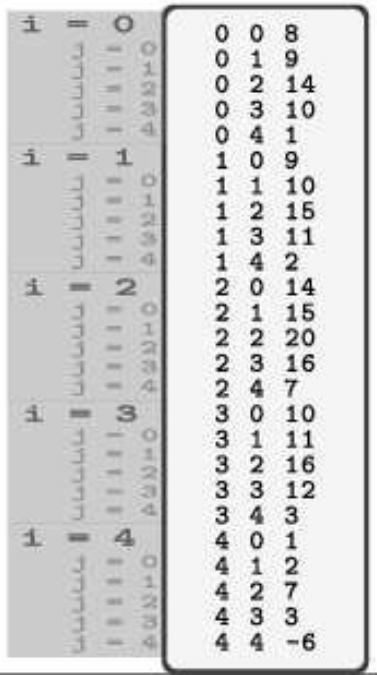
A[3] 6 → 25 → 47

A[4] −3 → 22

```
A = [4, 5, 10, 6, -3]

for i in range(0, len(A)):
    for j in range(i+1, len(A)):
        print(i, j, A[i] + A[j])
```




```
A = [4, 5, 10, 6, -3]

for i in range(0, 5):
    for j in range(0, 5):
        print(i, j, A[i] + A[j])
```



```
A = [4, 5, 10, 6, -3]

for i in range(0, 5):
    for j in range(0, 5):
        print(i, j, A[i] + A[j])
```

```
A = [4, 5, 10, 6, -3]

for i in range(0, len(A)):
    for j in range(0, len(A)):
        print(i, j, A[i] + A[j])
```

```
0 0 8
0 1 9
0 2 14
0 3 10
0 4 1
1 0 9
1 1 10
1 2 15
1 3 11
1 4 2
2 0 14
2 1 15
2 2 20
2 3 16
2 4 7
3 0 10
3 1 11
3 2 16
3 3 12
3 4 3
4 0 1
4 1 2
4 2 7
4 3 3
4 4 -6
```

new idea

UN EXEMPLE

Étant donné une liste A de 6 nombres, écrivez du code Python pour trouver les trois nombres dans A dont la somme est maximale.

```
A = [3, 43, 22, 80, 2, 4]

max = A[0]+A[1]+A[2]

if A[0]+A[1]+A[3] > max: max = A[0]+A[1]+A[3]
if A[0]+A[1]+A[4] > max: max = A[0]+A[1]+A[4]
if A[0]+A[1]+A[5] > max: max = A[0]+A[1]+A[5]
if A[0]+A[2]+A[3] > max: max = A[0]+A[2]+A[3]
if A[0]+A[2]+A[4] > max: max = A[0]+A[2]+A[4]
if A[0]+A[2]+A[5] > max: max = A[0]+A[2]+A[5]
if A[0]+A[3]+A[4] > max: max = A[0]+A[3]+A[4]
if A[0]+A[3]+A[5] > max: max = A[0]+A[3]+A[5]
if A[0]+A[4]+A[5] > max: max = A[0]+A[4]+A[5]

if A[1]+A[2]+A[3] > max: max = A[1]+A[2]+A[3]
if A[1]+A[2]+A[4] > max: max = A[1]+A[2]+A[4]
if A[1]+A[2]+A[5] > max: max = A[1]+A[2]+A[5]
if A[1]+A[3]+A[4] > max: max = A[1]+A[3]+A[4]
if A[1]+A[3]+A[5] > max: max = A[1]+A[3]+A[5]
if A[1]+A[4]+A[5] > max: max = A[1]+A[4]+A[5]

if A[2]+A[3]+A[4] > max: max = A[2]+A[3]+A[4]
if A[2]+A[3]+A[5] > max: max = A[2]+A[3]+A[5]
if A[2]+A[4]+A[5] > max: max = A[2]+A[4]+A[5]

if A[3]+A[4]+A[5] > max: max = A[3]+A[4]+A[5]

print("Maximum sum of three numbers is: ", max)
```

```
Maximum sum of three numbers is:  145
```

Étant donné une liste A de 6 nombres, écrivez du code Python pour trouver les trois nombres dans A dont la somme est maximale.

```
A = [3, 43, 22, 80, 2, 4]

max = A[0]+A[1]+A[2]

for i in range(0, len(A)):
    for j in range(i+1, len(A)):
        for k in range(j+1, len(A)):
            if A[i]+A[j]+A[k] > max:
                max = A[i]+A[j]+A[k]

print("Maximum sum of three numbers is: ", max)
```

```
Maximum sum of three numbers is:  145
```

Fonctionne pour n'importe quel A!

```
A = [3, 43, 22, 80, 2, 4, 57, 12, 67, 51, 23, 31]

max = A[0]+A[1]+A[2]

for i in range(0, len(A)):
    for j in range(i+1, len(A)):
        for k in range(j+1, len(A)):
            if A[i]+A[j]+A[k] > max:
                max = A[i]+A[j]+A[k]

print("Maximum sum of three numbers is: ", max)
```

```
Maximum sum of three numbers is:  204
```

new idea

LES INDICES DES BOUCLES

```
A = [4, 5, 100, 6, -3]

for i in range(0, 5):
    print( A[i] )
```

→

```
4
5
100
6
-3
```

```
for i in range(0, 5):
```

→

```
i = 0
i = 1
i = 2
i = 3
i = 4
```

Peut parcourir n'importe quelle liste :

```
A = [4, 5, 100, 6, -3]

for i in [2,0,-1] :
    print(i, A[i] )
```

→

```
2 100
0 4
-1 -3
```

```
B = ["hello", "again", "4.512", 4.512, 'bye', True]
for i in B :
    print( i )
```

↓

```
hello
again
4.512
4.512
bye
True
```

```
B = ["hello", "again", "4.512", 4.512, 'bye', True]
for i in B :
    print( B[i] )
```

↓

```
Traceback (most recent call last):
  File "prog.py", line 3, in <module>
    print( B[i] )
           ~~~~
TypeError: list indices must be integers or slices, not str
```

tout cela donne la même réponse !

```
A = [4, 5, 100, 6, -3]

for i in range(0, 5):
    print( A[i] )
```

↓

```
A = [4, 5, 100, 6, -3]

for x in A:
    print( x )
```

→

```
4
5
100
6
-3
```

```
B = ["hello", "again", "4.512", 4.512, 'bye']
For i in range(0,5) :
    print( B[i] )
```

↓

```
  File "/home/nabil/tmp/t.py", line 2
    For i in B :
        ^
SyntaxError: invalid syntax
```

```
for i in range(0, 4):
    print(i)
print(i)
```

La meme ?

↓

```
0
1
2
3
3
```

```
#include <stdio.h>

int main()
{
  int i = 0;
  for (i=0;i<4;i++)
    printf("%d\n", i);
  printf("%d\n", i);
}
```

↓

```
0
1
2
3
4
```

Différence entre Python et C++

En Python, lorsque la boucle se termine, c'est toujours le dernier index

new idea

BOUCLES WHILE