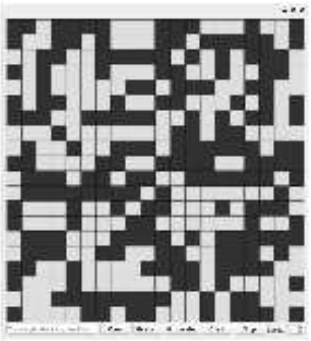
PYTHON

langage de programmation simple, mais puissant

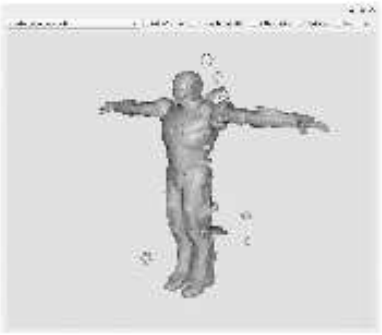
deux types de TP :

des exercices de base un problème à resoudre



Game of Life

Sudoku



3D modélisation

L'objectif est d'apprendre et de découvrir... **il faut poser des questions !**

Evaluation

il y aura deux tests

Ces tests surprises pourront tomber aussi bien en séance de TD qu'en TP, sans avertissement à l'avance.

il y aura un examen à mi-parcours

il y aura un examen finale

new idea

VARIABLES

premier programme en Python

pas besoin de spécifier le type de variable . . . Python l'attribuera automatiquement !

variable

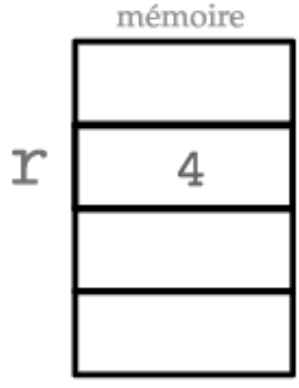
```
r = 4
```

imprimer valeur de variable r

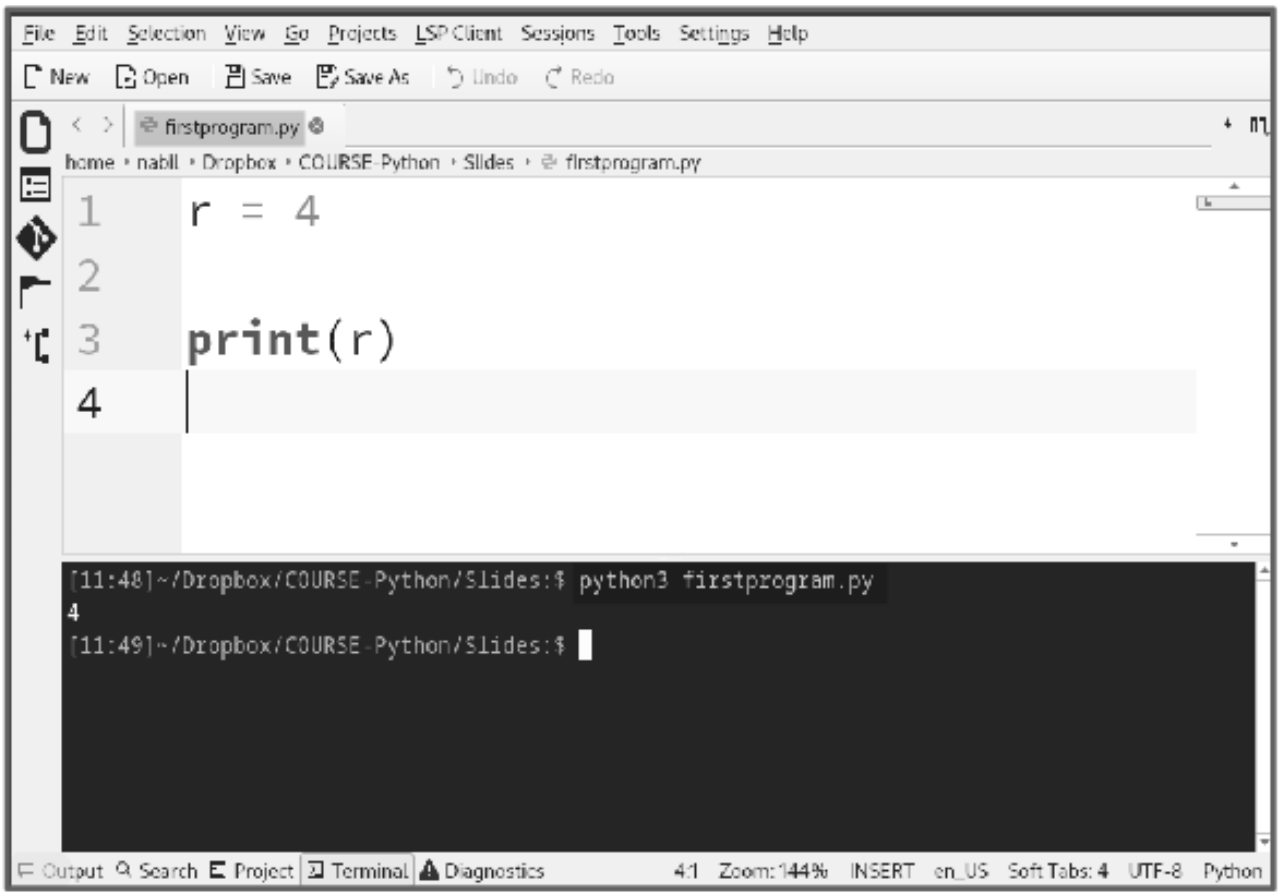
```
print(r)
```

↓

```
4
```

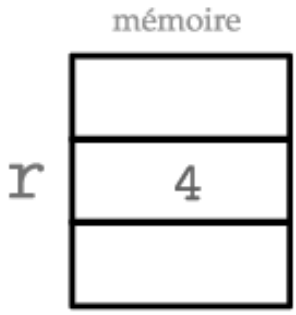


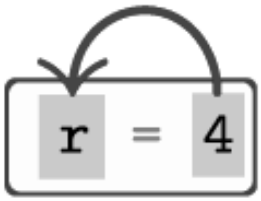
Une variable est une zone de la mémoire de l'ordinateur dans laquelle une valeur est stockée



Une ligne qui contient **deux** étapes:

1. Python alloue l'espace en mémoire, nommé `r`
2. Python assigne la valeur `4` à la variable nommé `r`





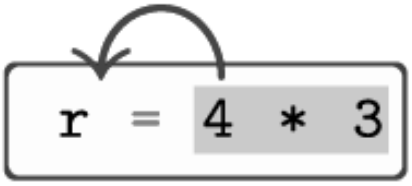
Python:

'=' est une **action**

1. évaluer l'expression de droite

2. assigne sa valeur à la variable de gauche

Attention: en mathematics, '=' signifie quelque chose de différent : que les parties gauche et droite sont égales



1. évaluer l'expression de droite

2. assigne sa valeur à la variable de gauche

```
r = 0
r = r + 10
print(r)
```

→ `10`

Python:

'=' est une **action**

1. ajouter `10` à la valeur actuelle de `r`

2. stocke le résultat dans la même variable `r`

le code est exécuté de manière séquentielle

```
r = 4

w = 11.343

y = r + w

print(r)
print(w)
print(y)
```

```
4
11.343
15.343
```

mémoire

w	11.343
r	4
y	15.343

Opérations sur les numériques

puissance

```
x = 2**5
y = x**2

print(x)
print(y)
```

```
32
1024
```

quotient

```
x = 5 // 4
y = 34 // 4

print(x)
print(y)
```

```
1
8
```

remainder

```
x = 5 % 4
y = 34 % 4

print(x)
print(y)
```

```
1
2
```

new idea

PRINT

Imprimer du texte

Le texte doit être entre guillemets

```
print("Hello!")
```

affiche une chaîne de caractères

```
Hello!
```

Imprimer du texte

```
r = 5

print("r is: ", r)
```

séparateur en Python

```
r is:  5
```

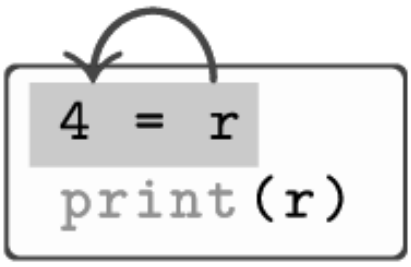
```
r = 5

print("r is: ", r, " and goodbye.")
```

```
r is:  5  and goodbye.
```

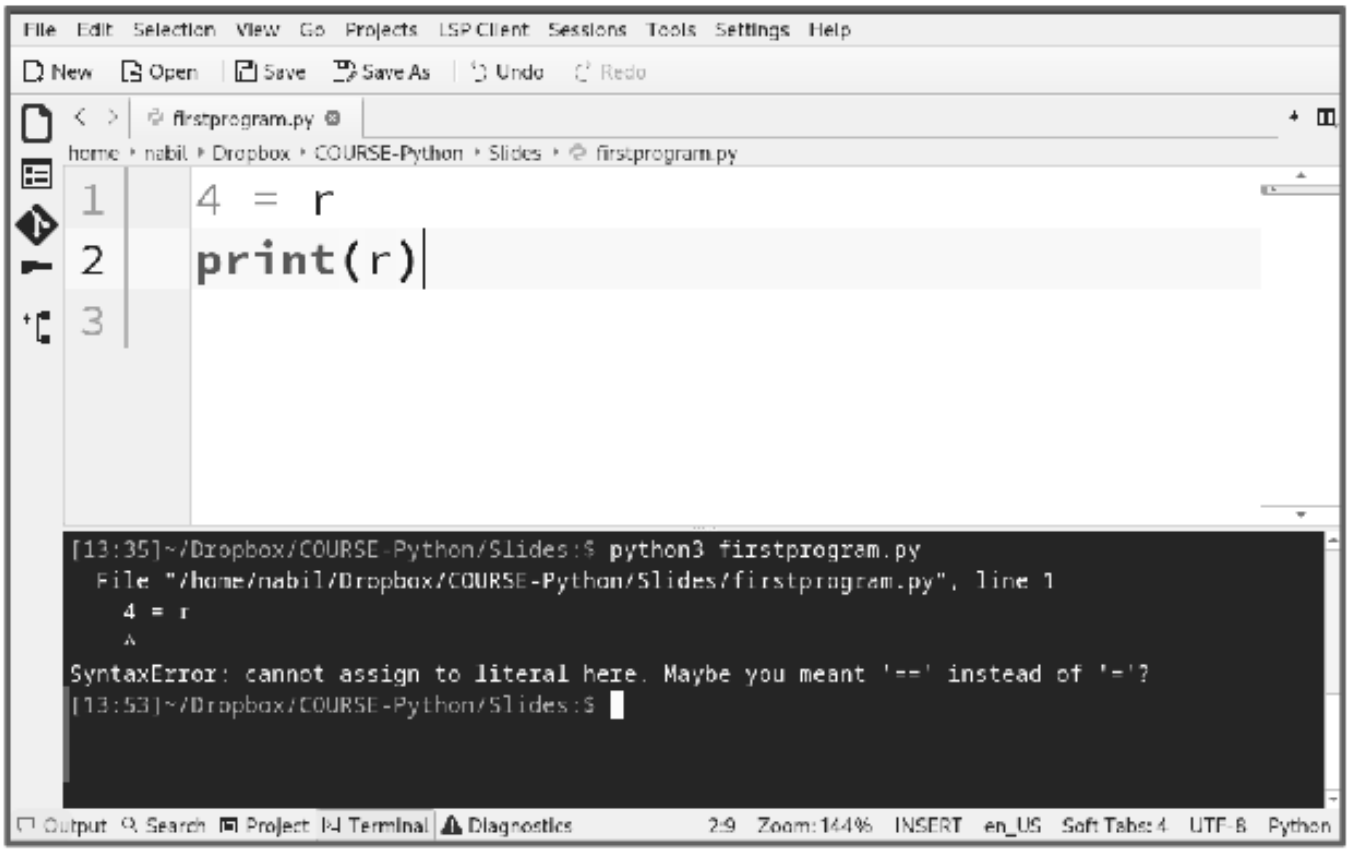
new idea

ERREURS

Problem . . . 4 n'est pas une variable!



```
File "/home/nabil/abc.py", line 1
    4 = r
    ^
SyntaxError: cannot assign to literal here.
Maybe you meant '==' instead of '='?
```



```
r

print(r)
```

```
Traceback (most recent call last):
  File "/home/nabil/abc.py", line 1, in <module>
    r
NameError: name 'r' is not defined
```

Pas d'erreur !

```
r = 4
r
print(r)
```

```
4
```

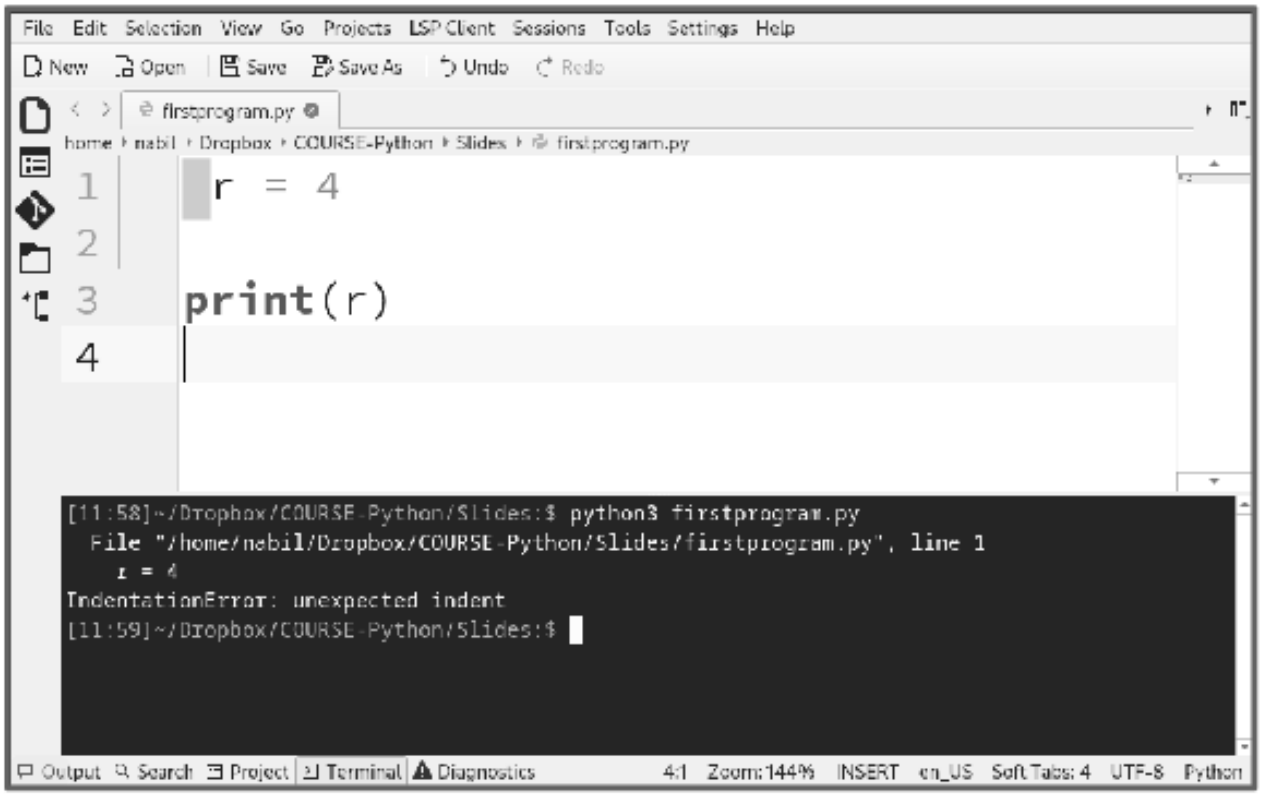
Différence énorme par rapport aux autres langues : l'indentation est essentielle

pas d'espace supplémentaire
au début de chaque ligne

```
 r = 4
print(r)
```

```
File "/home/nabil/abc.py", line 1
    r = 4
IndentationError: unexpected indent
```

l'indentation est importante !



```
r := 4

print(r)
```

```
  File "/home/nabil/abc.py", line 1
    r := 4
      ^^
SyntaxError: invalid syntax
```

le code est exécuté de manière séquentielle

```
① t = r + 10
② r = 4
③ print(r)
④ print(t)
```



```
Traceback (most recent call last):
  File "/home/nabil/abc.py", line 1, in <module>
    t = r + 10
NameError: name 'r' is not defined
```

```
r = 4
t = r + 10
print(r)
print(t)
```

```
4
14
```

```
r = r + 10
print(r)
```

```
Traceback (most recent call last):
  File "/home/nabil/abc.py", line 1, in <module>
    r = r + 10
NameError: name 'r' is not defined
```

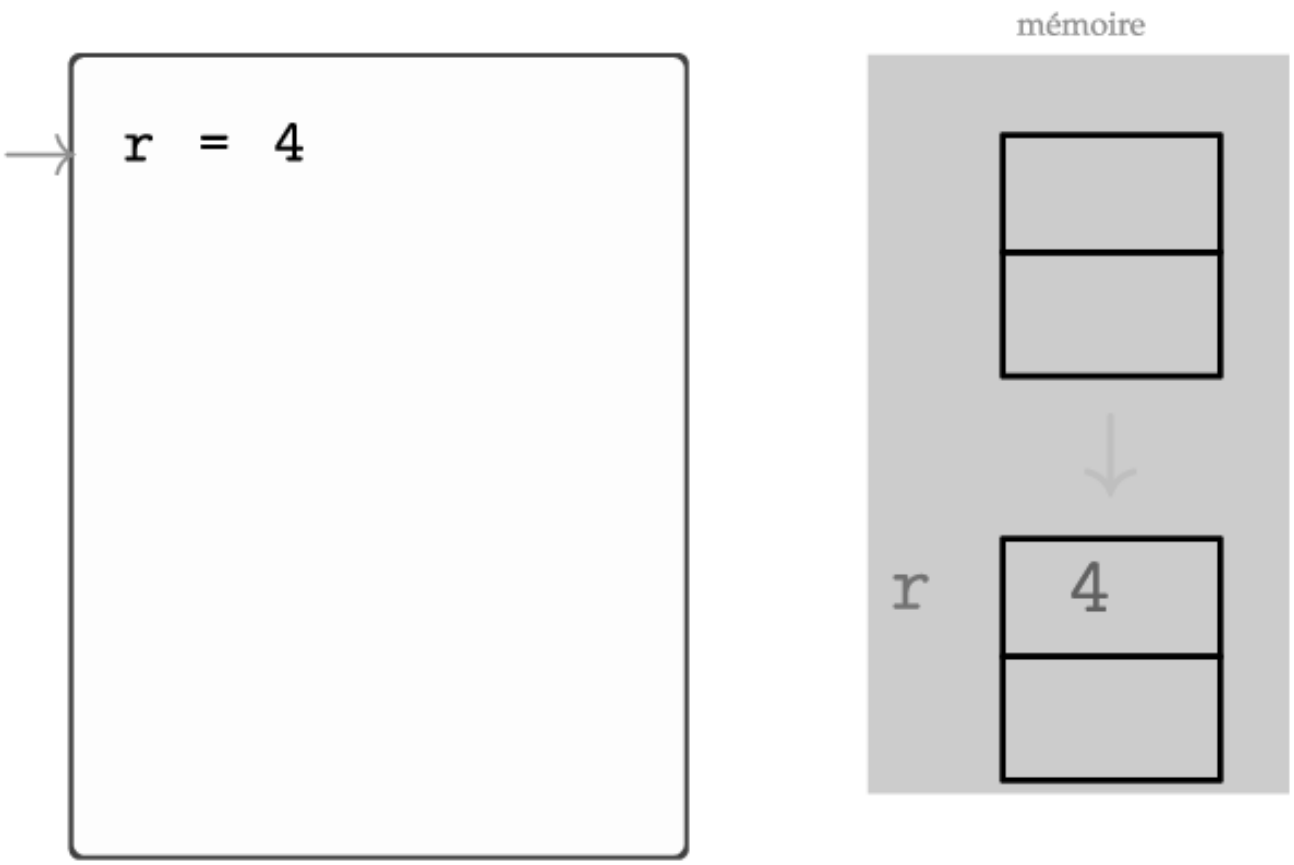
N'oubliez pas de mettre le texte entre guillemets !

```
print(Hello!)
```

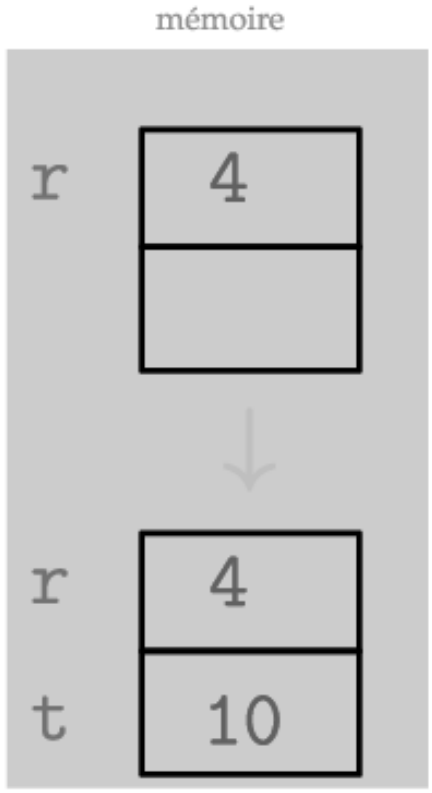
```
  File "firstprogram.py", line 1
    print(Hello!)
                ^
SyntaxError: invalid syntax
```

new idea

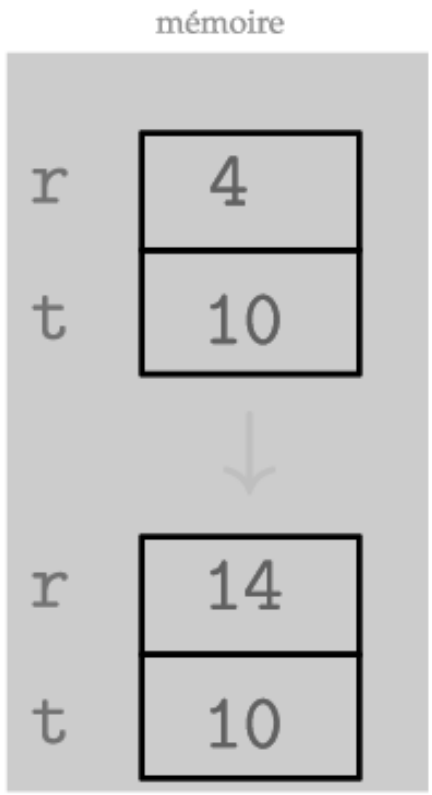
UN EXEMPLE



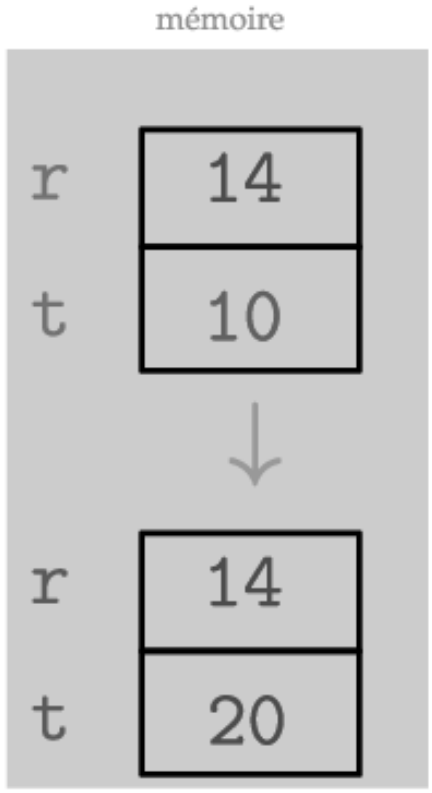
```
r = 4
t = 10
```



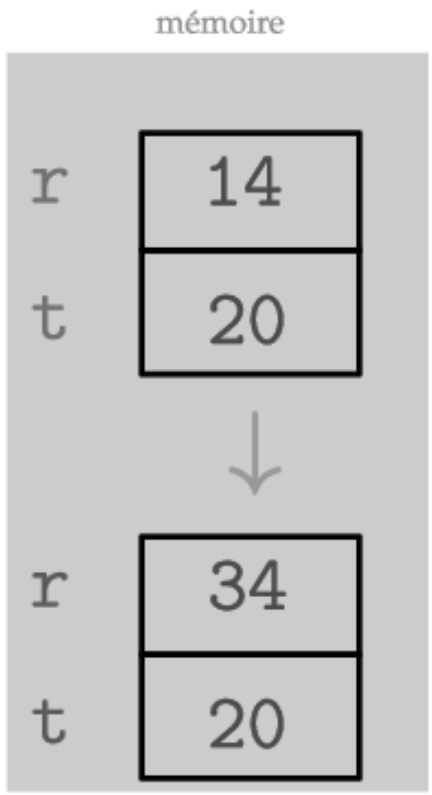

```
r = 4
t = 10
r = r + t
```



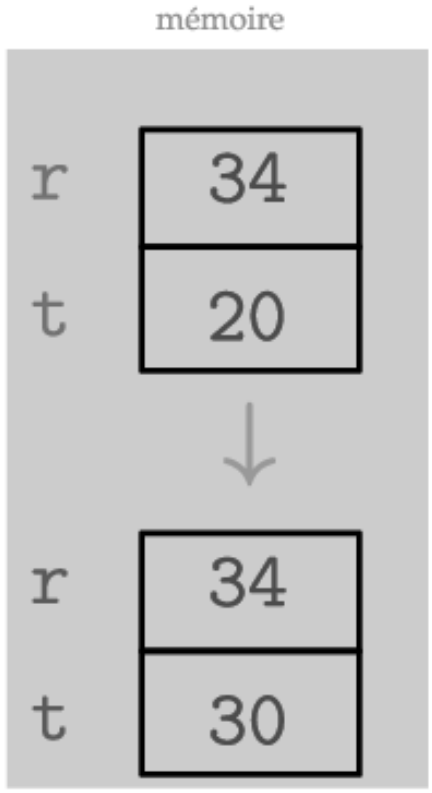
```
r = 4
t = 10
r = r + t
t = t + 10
```



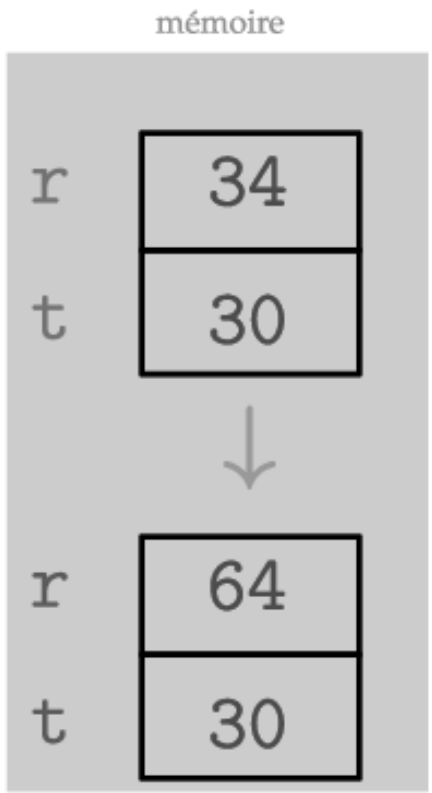

```
r = 4
t = 10
r = r + t
t = t + 10
r = r + t
```



```
r = 4
t = 10
r = r + t
t = t + 10
r = r + t
t = t + 10
```

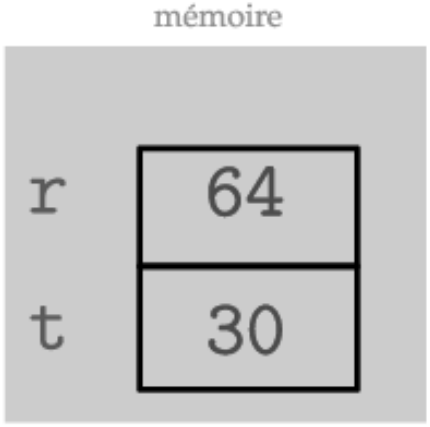



```
r = 4
t = 10
r = r + t
t = t + 10
r = r + t
t = t + 10
r = r + t
```



```
r = 4
t = 10
r = r + t
t = t + 10
r = r + t
t = t + 10
r = r + t
print(r)
print(t)
```

```
64
30
```



new idea

TYPE

Deux types de variables pour nombres : **entier**, et **decimal**

Python assigne, *automatiquement*, un type à une variable

utiliser `type(r)` à savoir le type de variable `r`

```
r = 4
```

integer

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

↓

```
<class 'int'>
```

```
r = 4.3554
```

float

```
r = 4.3554
print( type(r) )
```

↓

```
<class 'float'>
```

vous pouvez convertir entre les types

```
r = int(3.4554)
print(r)
print( type(r) )
```

↓

```
3
<class 'int'>
```

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

↓

```
4.0
<class 'float'>
```

`int` : supprime simplement la partie décimale

```
r = int(3.0004)
print(r)
```

3

```
r = int(3.6554)
print(r)
```

3

```
r = int(3.9999)
print(r)
```

3

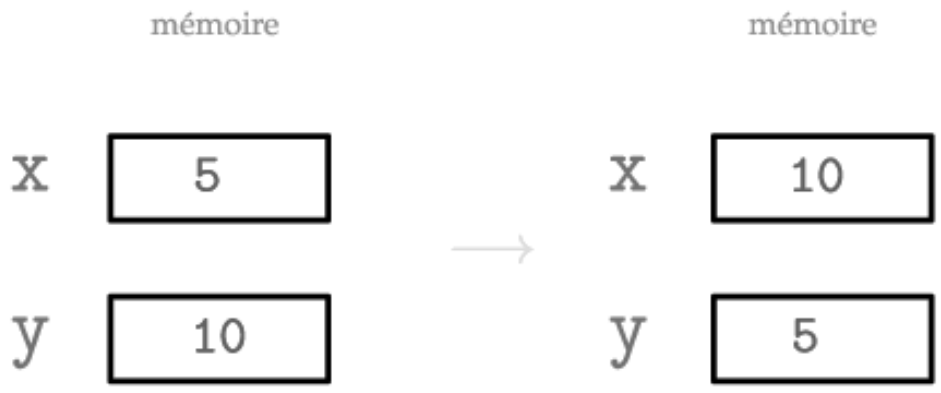
```
r = 14
t = 14 / 3
s = int(14 / 3)

print("r is: ", r)
print("t is: ", t)
print("s is: ", s)
```

```
r is:  14
t is:  4.666666666666667
s is:  4
```

new idea

UN EXERCICE

Question: Échangez les valeurs en x et y.



Question: Échangez les valeurs en x et y.

```
x = 5
y = 10
print("before x and y: ", x, y)

x = y
y = x
print("after x and y: ", x, y)
```

```
before x and y:  5 10
after x and y:  10 10
```

Question: Échangez les valeurs en x et y.

```
x = 5
y = 10
print("before x and y: ", x, y)

tmp = x
x = y
y = tmp
print("after x and y: ", x, y)
```

```
before x and y:  5 10
after x and y:  10 5
```

Python : **combiner les affectations**

```
a = 10
b = 4
print(a, b)
```

les deux sont identiques

```
a, b = 10, 4
print(a,b)
```

```
10 4
```

```
10 4
```

```
a, b, c, d = 10, 8, 4, 14
print(a, b, c, d)
```

```
10 8 4 14
```

```
r = 4
s = 13.6
print(r, type(r) )
print(s, type(s) )
t = r**2
print(t, type(t) )
u = float(r**2)
print(u, type(u) )
v = float( int(16.45) // 3 ) / int(2.343)
print(v, type(v) )
```

```
4 <class 'int'>
13.6 <class 'float'>
16 <class 'int'>
16.0 <class 'float'>
2.5 <class 'float'>
```

new idea

CONDITIONALS

A first example:

```
a = 4
b = 10

if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

```
b is larger.
```

```
a = 4
b = 10

if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

→ `b is larger.`

a > b ?

True ↓ ↓ **Sinon**

`print("a is larger.")` `print("b is larger.")`

Les espaces—l'indentation—obligatoire aprés if et else !

```
a = 40
b = 10
if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

4 espaces

↓

`a is larger.`

Au moins un espace obligatoire

Dans ce module : **4 espaces**

```
a = 40
b = 10

if a > b:
    c = a
else:
    c = b

print(c)
```

```
40
```

```
if CONDITION :
    instructions à exécuter
    si CONDITION est True
else :
    instructions à exécuter
    si CONDITION est False
```

'==' : vérifier l'égalité

```
a = 40
b = 10

if a == b:
    print("a and b are equal.")
else:
    print("a and b are not equal.")
```

```
a and b are not equal.
```

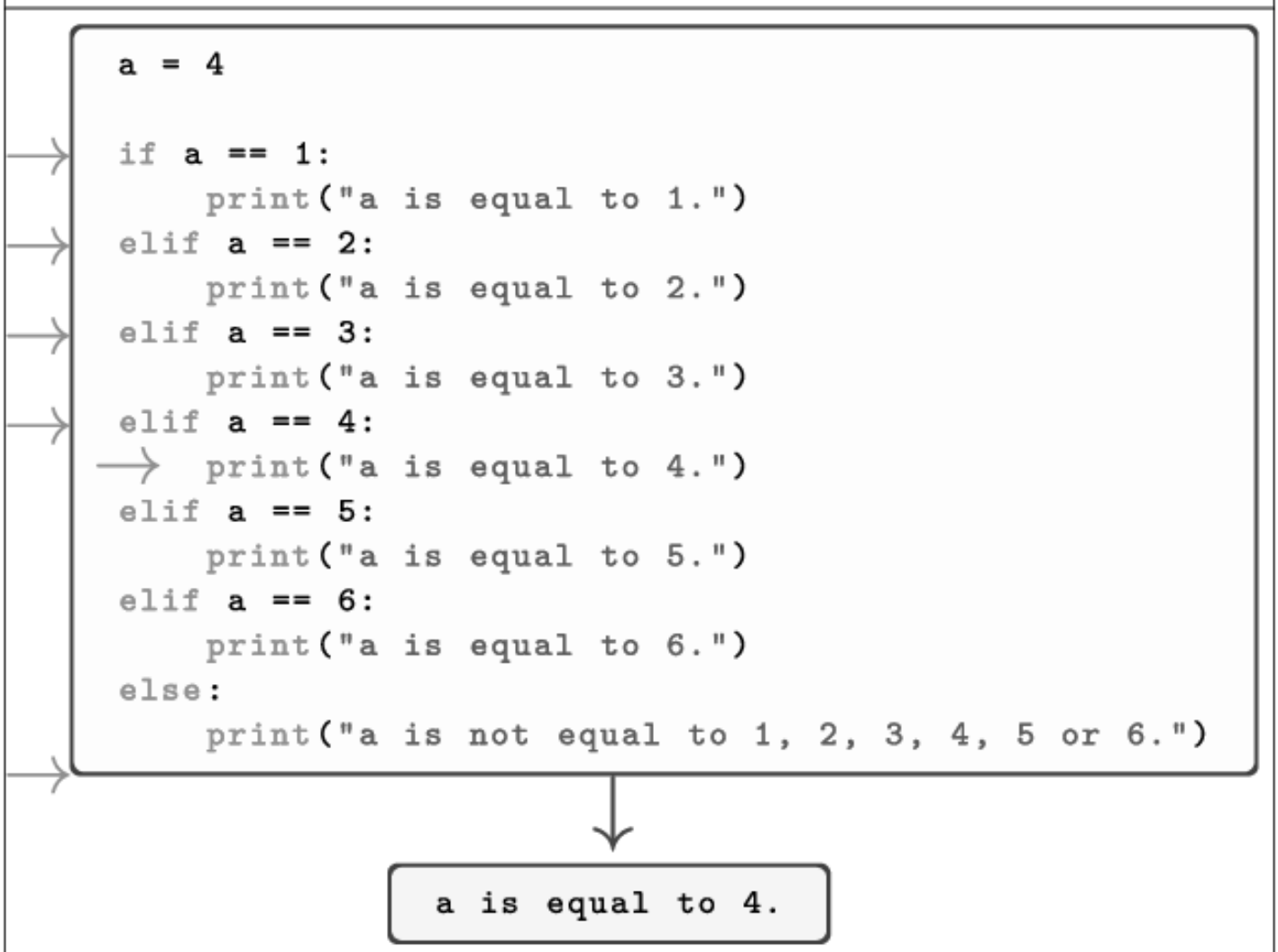
```
a = 40
b = 10

if a == b:
    print("a and b are equal.")
else:
    print("a and b are not equal.")
```

'==' vérifier l'égalité

`a == b`	**True** ssi `a` est égal à `b`
`a != b`	**True** ssi `a` n'est pas égal à `b`
`a >= b`	**True** ssi `a` supérieur ou égal à `b`
`a > b`	**True** ssi `a` supérieur à `b`

enchainer if conditions avec elif

```
a = 4

if a == 1:
    print("a is equal to 1.")
elif a == 2:
    print("a is equal to 2.")
elif a == 3:
    print("a is equal to 3.")
elif a == 4:
    print("a is equal to 4.")
elif a == 5:
    print("a is equal to 5.")
elif a == 6:
    print("a is equal to 6.")
else:
    print("a is not equal to 1, 2, 3, 4, 5 or 6.")
```



```
a = 6

if a == 1:
    print("a is equal to 1.")
elif a == 2:
    print("a is equal to 2.")
elif a == 3:
    print("a is equal to 3.")
elif a == 4:
    print("a is equal to 4.")
elif a == 5:
    print("a is equal to 5.")
elif a == 6:
    print("a is equal to 6.")
else:
    print("a is not equal to 1, 2, 3, 4, 5 or 6.")
```

```
a is equal to 6.
```

```
a = 8

if a == 1:
    print("a is equal to 1.")
elif a == 2:
    print("a is equal to 2.")
elif a == 3:
    print("a is equal to 3.")
elif a == 4:
    print("a is equal to 4.")
elif a == 5:
    print("a is equal to 5.")
elif a == 6:
    print("a is equal to 6.")
else:
    print("a is not equal to 1, 2, 3, 4, 5 or 6.")
```

```
a is not equal to 1, 2, 3, 4, 5 or 6.
```

new idea

ERREURS

l'indentation obligatoire aprés if et else !

des espaces !

```
a = 40
b = 10

if a > b:
c = a
else:
c = b

print(c)
```

```
  File "/home/nabil/abc.py", line 5
    c = a
    ^
IndentationError: expected an indented block
after 'if' statement on line 4
```

```
a = 40
b = 10

if a >= b
    print("a is larger.")
else:
    print("b is larger.")
```

```
File "/home/nabil/abc.py", line 4
    if a >= b
             ^
SyntaxError: expected ':'
```

Pas d'erreur !

... mais dans ce module : il faut utiliser 4 espaces

```
a = 40
b = 10

if a > b:
 c = a
else:
 c = b

print(c)
```

```
40
```

new idea

BOOL

un nouveau `type` de variable : **bool**ean

Ne peut prendre que deux valeurs : True ou False

```
t = True
```

```
t = True

print(t)
print( type(t) )
```

```
True
<class 'bool'>
```

```
t = False
```

```
t = False

print(t)
print( type(t) )
```

```
False
<class 'bool'>
```

Le résultat d'une comparaison est un Booléen !

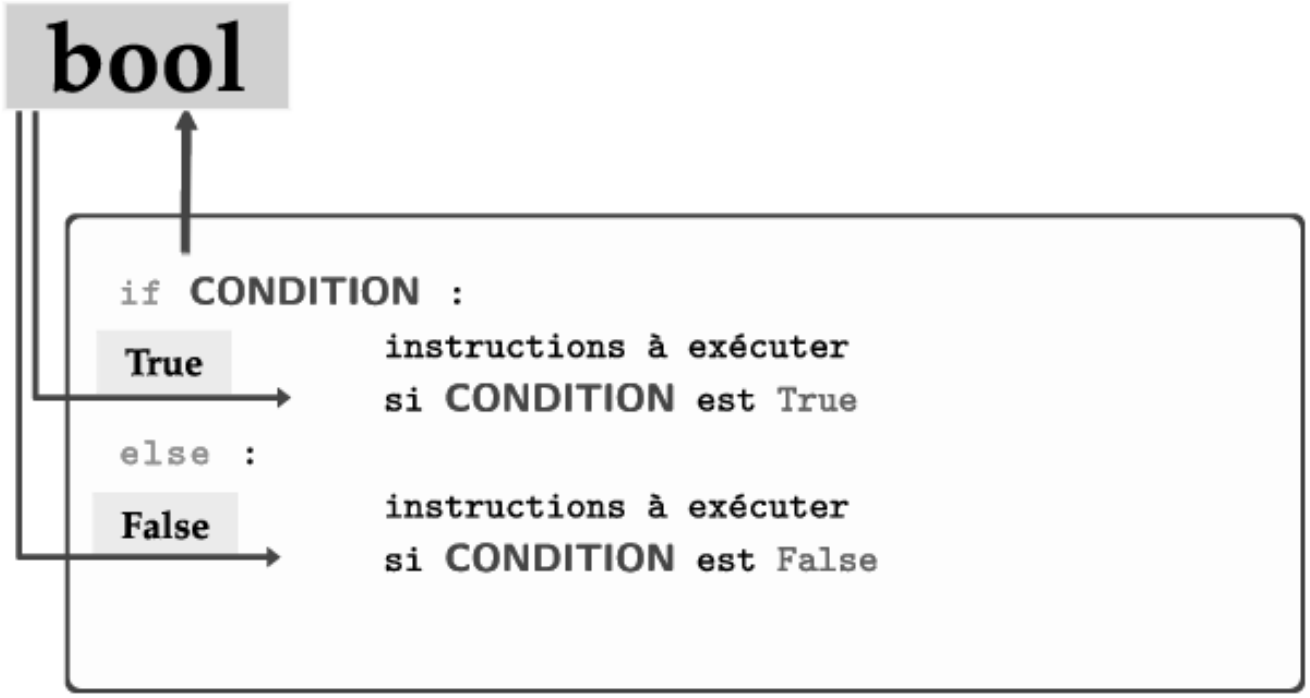
```
a = 10
b = 5

c = (a > b)
print("c is: ", c)

d = (b > a)
print("d is: ", d)
```

```
c is:  True
d is:  False
```

```
a = 10
b = 10.239

c = ( int(b) >= a )
print("c is: ", c)

d = ( a >=  int(b) )
print("d is: ", d)
```

```
c is:  True
d is:  True
```

En fait,



les deux sont exactement identiques

```
a = 4
b = 10

if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

```
b is larger.
```

```
a = 4
b = 10

c = (a > b)

if c:
    print("a is larger.")
else:
    print("b is larger.")
```

```
b is larger.
```

```
if CONDITION 1 :
    instructions à exécuter si CONDITION 1 est True

elif CONDITION 2 :
    instructions à exécuter si CONDITION 1 est
     False et
                                CONDITION 2 est True
elif CONDITION 3 :
    instructions à exécuter si CONDITIONS 1 2 sont
     False et
                                CONDITION 3 est True
  .
  .
  .
elif CONDITION 9 :
    instructions à exécuter si CONDITIONS 1 ··· 8
     sont False et
                                CONDITION 9 est True
else :
    instructions à exécuter si CONDITIONS 1 ··· 9
     sont False
```

new idea

OPÉRATEURS BOOLÉENS

utiliser and et or pour combiner conditions

```
a = 30
b = 10
c = 40

if a >= b and a >= c:
    print("a is maximum.")
elif b >= a and b >= c:
    print("b is maximum.")
else:
    print("c is maximum.")
```

```
c is maximum.
```

```
if CONDITION1 and CONDITION2:
            ...
else :
            ...
```

CONDITION1	CONDITION2	CONDITION1 and CONDITION2
True	True	→ True
True	False	→ False
False	True	→ False
False	False	→ False

```
if CONDITION1 or CONDITION2:
                ...
else :
                ...
```

CONDITION1	CONDITION2	CONDITION1 or CONDITION2
True	True	→ True
True	False	→ True
False	True	→ True
False	False	→ False

```
if not CONDITION :
                ...
else :
                ...
```

CONDITION	not CONDITION
True	→ False
False	→ True

```
a = 10
b = 20
c = 30
d = 42

if (a < b and a > c) or (c > b and d < b):
    print("First.")
elif (a > b or c > a) and (d < c or d < b):
    print("Second.")
else:
    print("Third.")
```

```
Third.
```

tout cela donne la même réponse !

```
a = 4
b = 10

if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

```
a = 4
b = 10

c = (a > b)

if c:
    print("a is larger.")
else:
    print("b is larger.")
```

```
a = 4
b = 10

c = (a > b)

if not c:
    print("b is larger.")
else:
    print("a is larger.")
```

```
a = 4
b = 10

if not a <= b:
    print("a is larger.")
else:
    print("b is larger.")
```

le code est exécuté de manière séquentielle

```
a = 40
b = 10
c = 40

if a >= b and a >= c:
    print("a is maximum.")
elif b >= a and b >= c:
    print("b is maximum.")
elif c >= a and c >= b:
    print("c is maximum.")
```

```
a is maximum.
```

N'entre que la première fois la condition est vraie.

Même s'il existe plusieurs conditions vraies

```
a = 42

if a > 10:
    print("First!")
elif a > 20:
    print("Second!")
elif a > 30:
    print("Third!")
elif a > 40:
    print("Fourth!")
```

```
First!
```

```
a = 25

if a > 10:
    print("First!")
elif a > 20:
    print("Second!")
elif a > 30:
    print("Third!")
elif a > 40:
    print("Fourth!")
```

```
First!
```

les trois ifs sont indépendantes !

```
a = 40
b = 10
c = 40

if a >= b and a >= c:
    print("a is maximum.")

if b >= a and b >= c:
    print("b is maximum.")

if c >= a and c >= b:
    print("c is maximum.")
```

```
a is maximum.
c is maximum.
```

```
c = 20

print("First.")
if c >= 30:
    print("Second.")
    if c >= 10:
        print("Third.")
```

```
First.
```

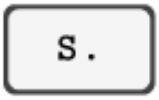
```
c = 50

print("First.")
if c >= 30:
    print("Second.")
    if c >= 10:
        print("Third.")
```

```
First.
Second.
Third.
```

L'indentation change la resultat !

```
c = 20

if c > 30:
    print("F.")
    if c > 10:
        print("S.")
```

```
c = 20

if c > 30:
    print("F.")
if c > 10:
    print("S.")
```



new idea

TRIER 3 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 nombres dans l'ordre croissant sont : ", t1, t2, t3)
```

new idea

COMMENTAIRES

insérer commentaire d'une ligne, avec le symbole #

```
a = 4
b = 10

#Cette ligne commentaire est ignoree par Python.

if a > b:
    print("a is larger.")
#cette ligne aussi !
else:
#et celui-ci !
    print("b is larger.")
```

les deux sont exactement identiques !

```
a = 4
b = 10
if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

```
b is larger.
```

```
a = 4
b = 10
#Il s'agit d'un commentaire de plusieurs lignes.
Tout ce paragraphe est ignoree par python.
Meme ici.
if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

```
File "prog.py", line 4
    Tout ce paragraphe est ignoree par python.
         ^^
SyntaxError: invalid syntax
```

Commentaire de plusieurs lignes avec les symboles """

```
a = 4
b = 10

"""Il s'agit d'un commentaire de plusieurs lignes.
Tout ce paragraphe est ignoree par python."""

if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

les deux sont exactement identiques !

```
a = 4
b = 10
if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

```
b is larger.
```

```
a = 4
b = 10
"""Il s'agit d'un commentaire de plusieurs lignes.
Tout ce paragraphe est ignoree par python.
Meme ici.
"""
if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

```
a = 4
b = 10
if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```

tous identiques !

```
a = 4
b = 10
#Il s'agit d'un commentaire de plusieurs lignes.
#Tout ce paragraphe est ignoree par python.
#Meme ici.
if a > b:
    print("a is larger.")
else:
    print("b is larger.")
```