# Informatique 1
# Introduction à la programmation

## Mission 4 : introduction

Kim Mens  Siegfried Nijssen  Charles Pecheur

# Les séquences

- Généralisent les chaînes de caractères, les listes
- Représentent des éléments ordonnés

```
"Charles"
```

| "C" | "h" | "a" | "r" | "l" | "e" | "s" |
|---|---|---|---|---|---|---|

```
[3, 4, 5, 6, 7, 8, 9]
```

| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|

- Des **opérations** permettent de manipuler les séquences
- On va voir:
  - Les opérations en commun entre toutes les séquences
  - Les opérations particulières pour certaines séquences

# Parcourir une séquence

Exemple: compter le nombre de 'A' dans une chaîne

```
def count_A ( s ):
    """
    Pre:  Une chaîne de caractères s
    Post: Retourne le nombre de caractères 'A' dans s
    """
```

```
>>> count_A ( "BANANA" )
3
>>> count_A ( "BaNANA" )
2
```

# Parcourir une séquence

Exemple: compter le nombre de 'A' dans une chaîne

```
def count_A ( s ):
    """ Pre:  Une chaîne de caractères s
    Post: Retourne le nombre de caractères 'A' dans s """
    cnt = 0
    for c in s:
```

(1) Déterminer la **séquence** : **éléments**

```
>>> count_A ( "BANANA" )
3
>>> count_A ( "BaNANA" )
2
```

# Parcourir une séquence

Exemple: compter le nombre de 'A' dans une chaîne

```
def count_A ( s ):
    """ Pre:  Une chaîne de caractères s
    Post: Retourne le nombre de caractères 'A' dans s """
    cnt = 0
    for c in s:
        if c == 'A':
```

(1) Déterminer la **séquence** : **éléments**
(2) Écrire la **condition** de traitement

```
>>> count_A ( "BANANA" )
3
>>> count_A ( "BaNANA" )
2
```

# Parcourir une séquence

Exemple: compter le nombre de 'A' dans une chaîne

```
def count_A ( s ):
    """ Pre:  Une chaîne de caractères s
    Post: Retourne le nombre de caractères 'A' dans s """
    cnt = 0
    for c in s:
        if c == 'A':
            cnt += 1
    return cnt
```

(1) Déterminer la **séquence** : **éléments**
(2) Écrire la **condition** de traitement
(3) Écrire le **corps** de la boucle

```
>>> count_A ( "BANANA" )
3
>>> count_A ( "BaNANA" )
2
```

# Parcourir une séquence

Exemple: compter le nombre de 'A' dans une chaîne

```
def count_A ( s ):
    """ Pre:  Une chaîne de caractères s
    Post: Retourne le nombre de caractères 'A' dans s """
    cnt = 0
    for c in s:
        if c == 'A':
            cnt += 1
    return cnt
```

(1) Ecrire l'**entête** de la fonction

(2) Rédiger les **spécifications** de la fonction

(3) Écrire le **corps** de la fonction

(4) **Return** un résultat, si nécessaire

```
>>> count_A ( "BANANA" )
3
>>> count_A ( "BaNANA" )
2
```

# Parcourir une séquence

Exemple: compter le nombre de 'A' dans une chaîne

```
def count_A ( s ):
    """ Pre:  Une chaîne de caractères s
    Post: Retourne le nombre de caractères 'A' dans s """
    cnt = 0
    for c in s:
        if c == 'A':
            cnt += 1
    return cnt
```

la chaîne `s` est une séquence de caractères

`c` est une chaîne avec un seul caractère

```
>>> count_A ( "BANANA" )
3
>>> count_A ( "BaNANA" )
2
```

# La notation […]

```
s = "BANANA"
```

- `s[0] == "B"`
- `s[3] == "A"`

```
s = ["LOUVAIN", "BRUXELLES", "MONS", "TOURNAI"]
```

- `s[1] == "BRUXELLES"`
- `s[3] == "TOURNAI"`

# La notation […]

```
s = "LOUVAIN"
```

| L | O | U | V | A | I | N |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| -7 | -6 | -5 | -4 | -3 | -2 | -1 |

| | |
|---|---|
| `s[2]` | `"U"` |
| `s[-1]` | `"N"` |
| `s[1:3]` | `"OU"` |
| `s[-4:-1]` | `"VAI"` |
| `s[2:]` | `"UVAIN"` |
| `s[:2]` | `"LO"` |
| `s[:]` | `"LOUVAIN"` |

# La fonction len (…)

Retourne la longueur (= nombre d'éléments) d'une séquence:

```
s = ["LOUVAIN", "BRUXELLES", "MONS", "TOURNAI"]
len ( s ) == 4
len ( s[0] ) == 7
len ( s[1] ) == 9
len ( s[2] ) == 4
```

# Boucles

Est-ce qu'il y a un caractère 'a' dans une chaîne de caractères?

**Sans range / len**

```
def a_a ( s ):
  for c in s:
    if c == 'a':
      return True
  return False
```

Correct,

préféré

**Avec range / len**

```
def a_a ( s ):
  for i in range ( len ( s ) ):
    if s[i] == 'a':
      return True
  return False
```

Correct,

mais pas préféré

# Concaténation

"+" calcule une nouvelle séquence qui est la concaténation des deux séquences d'origine

```
>>> s = "ab" + "bc"
>>> s
"abbc"
>>> l =  [ 1, 2 ] + [ 3, 4 ]
>>> l
[ 1, 2, 3, 4 ]
```

# Méthodes sur les chaînes de caractères

Beaucoup de méthodes existent pour créer des chaînes de caractères basés sur une autre chaîne de caractères:

```
>>> print ( "hello".upper () )
HELLO
>>> print ( "HELLO".lower () )
hello
>>> print ( "          text".strip () )
Text
```

Sans espaces

https://docs.python.org/3/library/stdtypes.html#string-methods

# Modification de listes

On peut **modifier** une liste

```
>>>> s = ["LOUVAIN", "BRUXELLES", "MONS", "TOURNAI"]
>>>> s.append ( "LIEGE" )
>>>> s
["LOUVAIN", "BRUXELLES", "MONS", "TOURNAI", "LIEGE"]
>>>> s[3]
"TOURNAI"
>>>> s[3] = "ARLON"
>>>> s
["LOUVAIN", "BRUXELLES", "MONS", "ARLON", "LIEGE"]
```

**Mais**…

```
>>>> s + ["GAND"]
["LOUVAIN", "BRUXELLES", "MONS", "ARLON", "LIEGE", "GAND"]
>>>> s
["LOUVAIN", "BRUXELLES", "MONS", "TOURNAI", "LIEGE"]
```

# Exemple: Créer une liste

- Créer la liste
  `l = [2,4,8,16,…,2**n]`

```
l = []
for k in range(1,n+1):
    l.append ( 2**k )
```

# Exemple: Modifier une liste

- Incrementer chaque élément dans une liste `l`

  `l = [4, 5, 7, 1] -> [ 5, 6, 8, 2]`

```
for i in range(len(l)):
    l[i] = l[i] + 1
```

# Exemple: Modifier une liste

Incrementer la valeur de chaque élément dans une liste `l`

`l = [4, 5, 7, 1] -> [ 5, 6, 8, 2]`

```
for i in range(len(l)):
    l[i] = l[i] + 1
```

```
for e in l:
    e = e + 1
```

**INCORRECT!**

# Listes comme paramètres

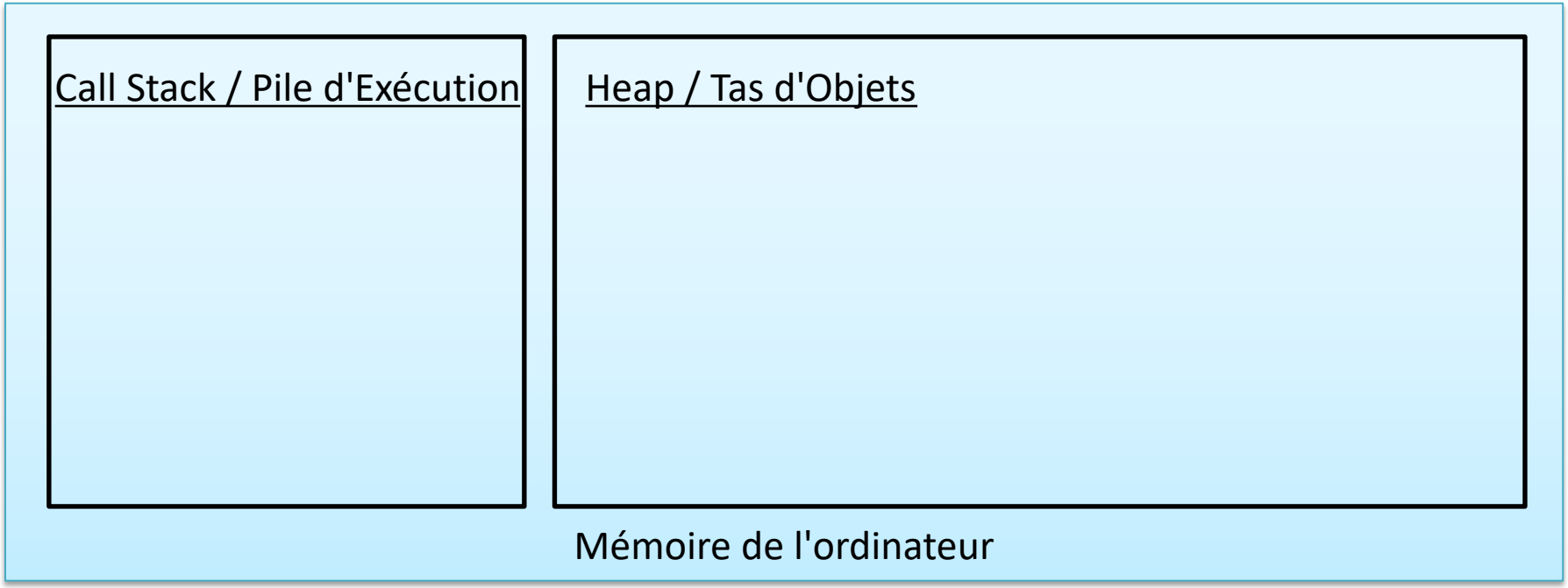
Une liste peut être l'argument d'une fonction

```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

```
[ 2, 3, 4 ]
```

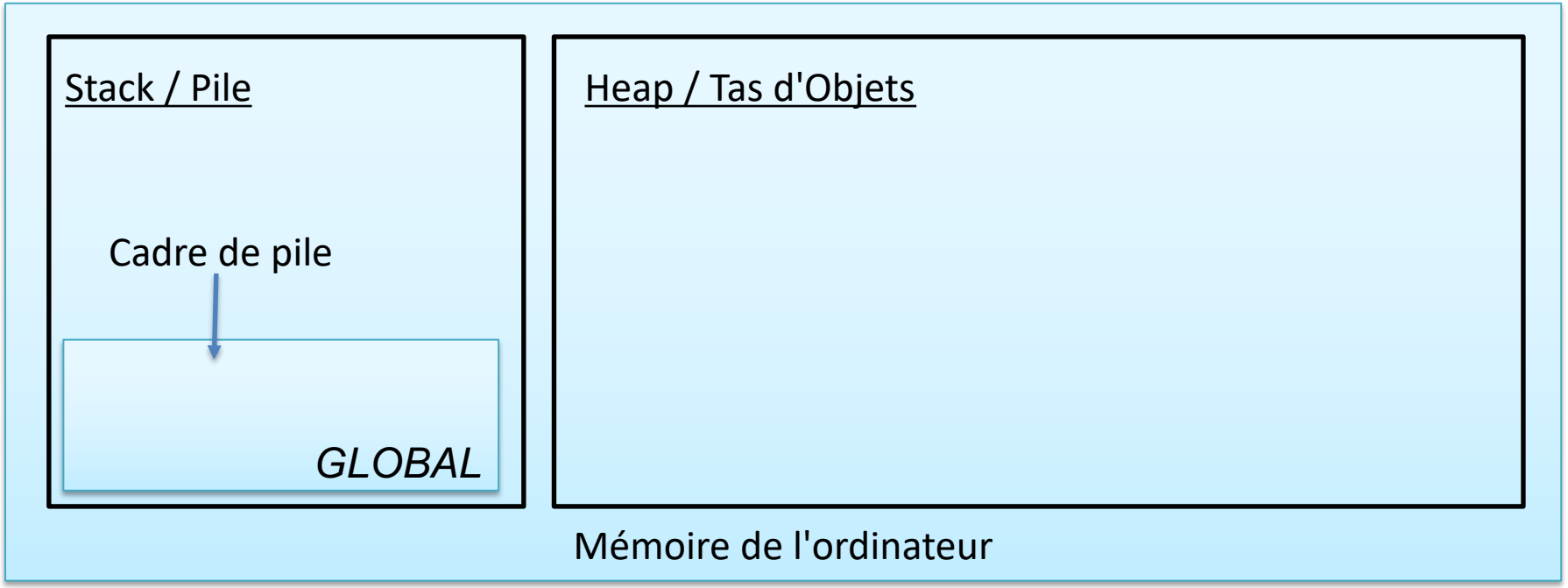
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

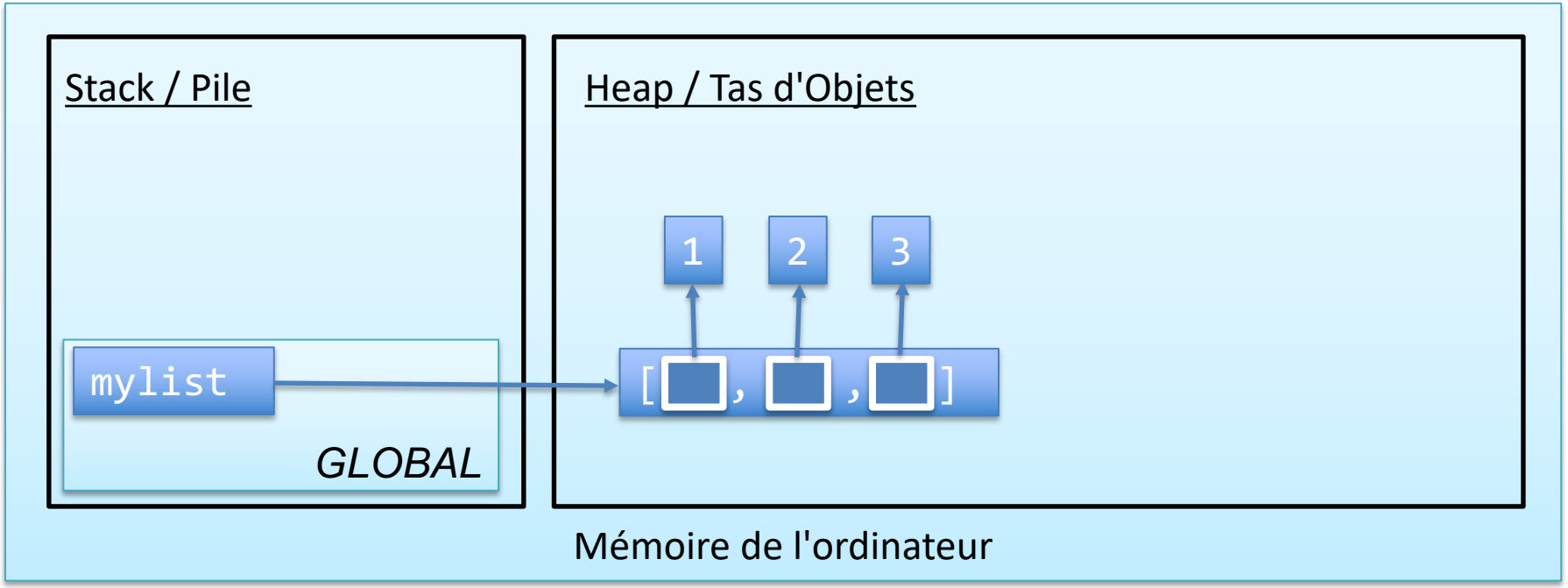
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

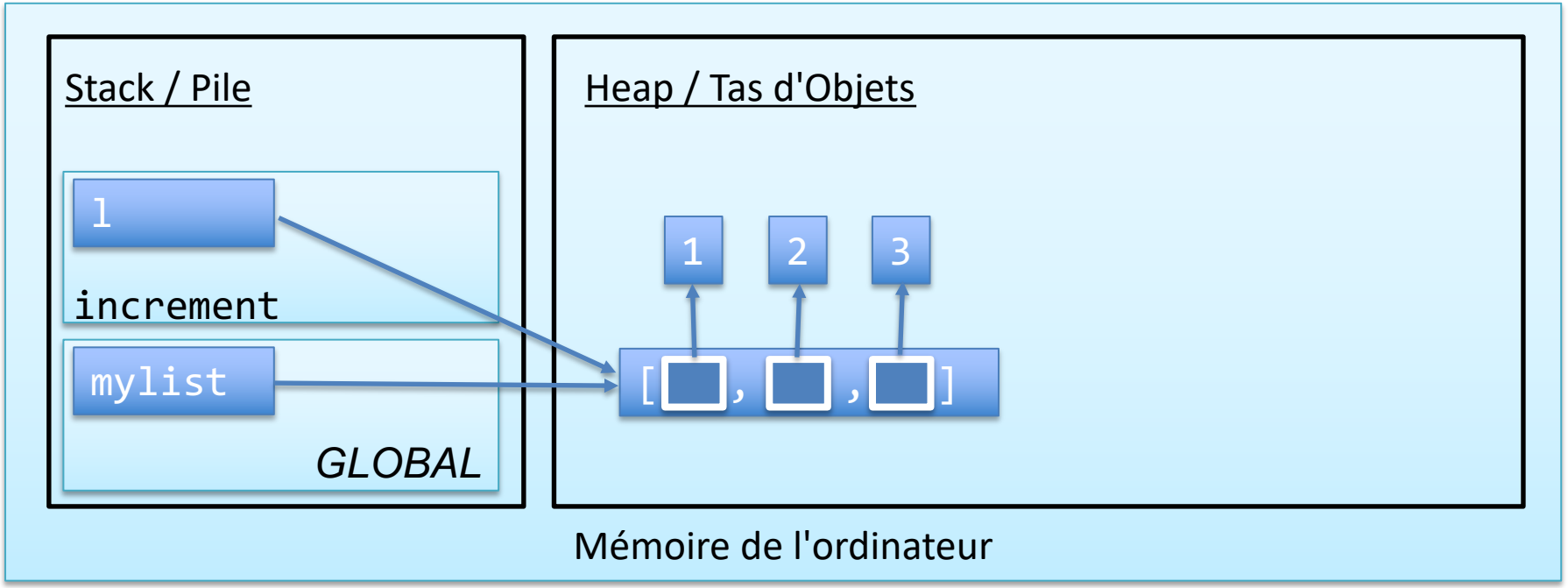
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

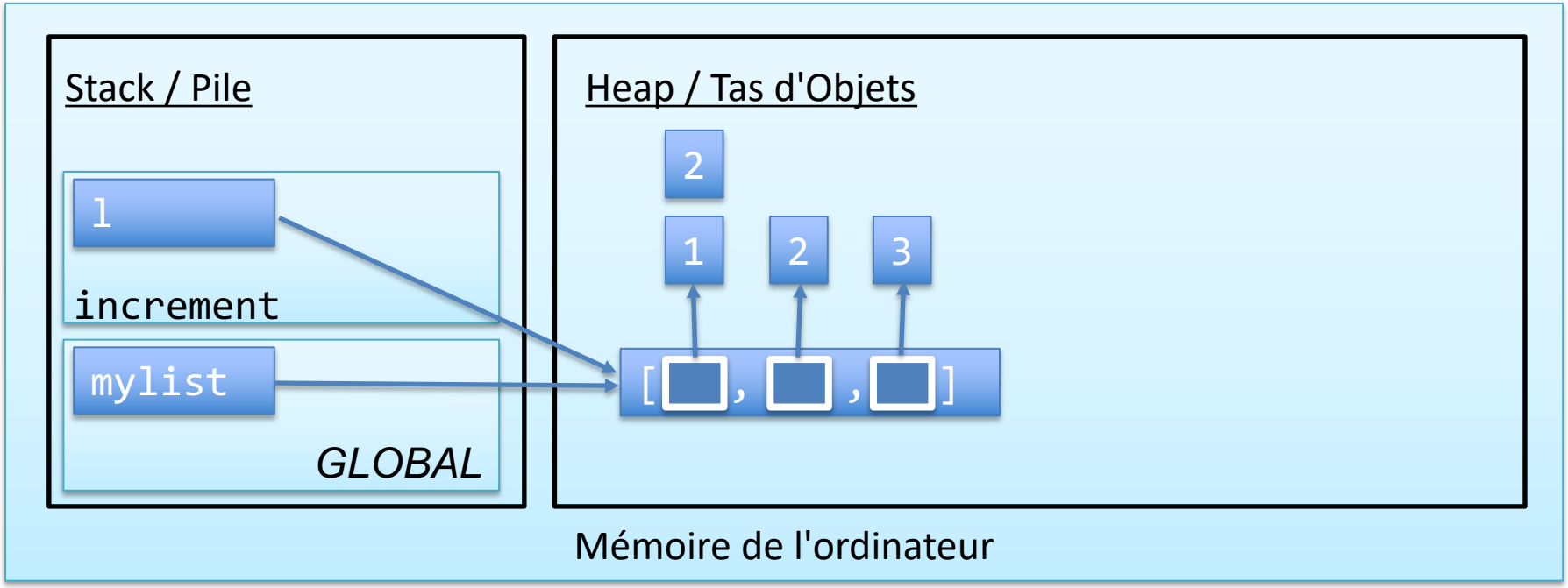
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

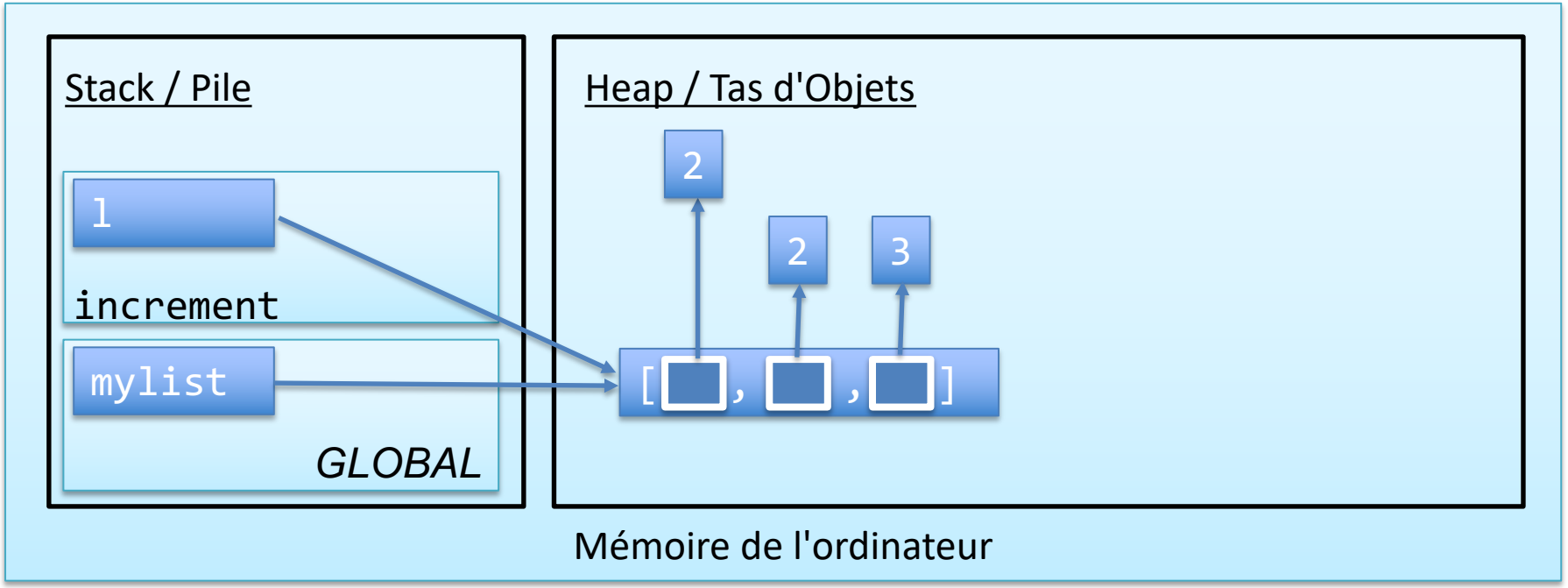
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

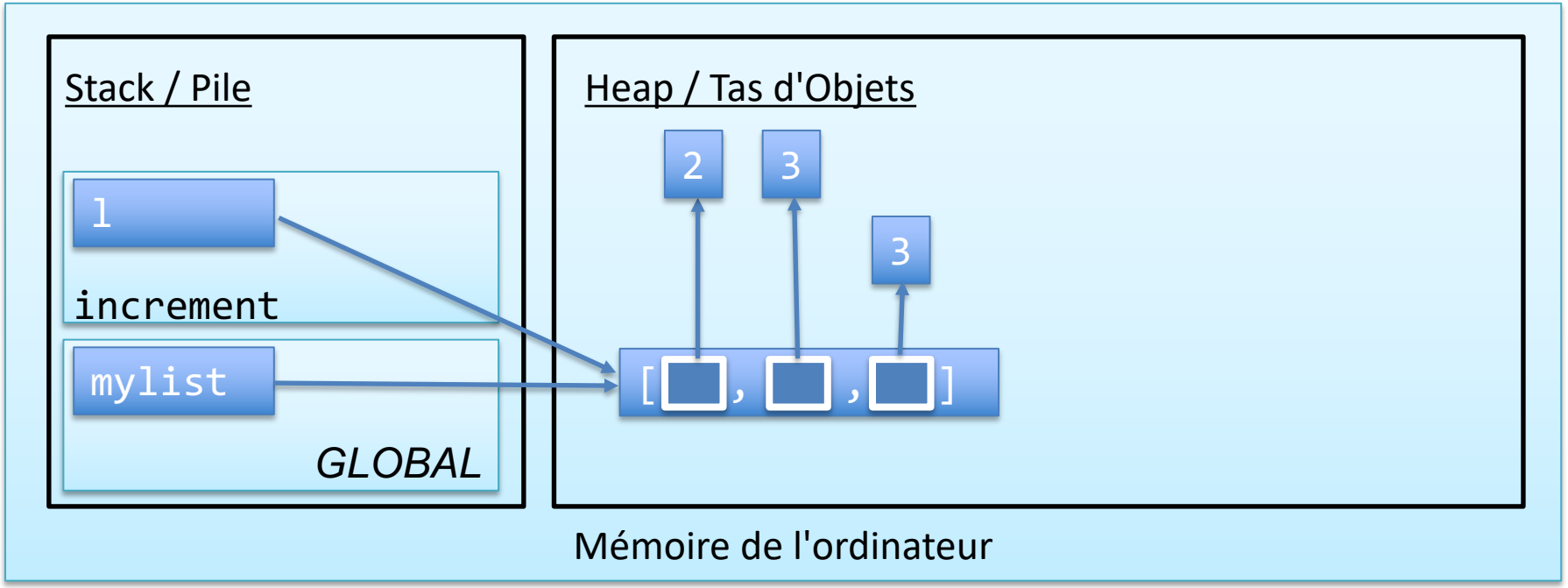
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

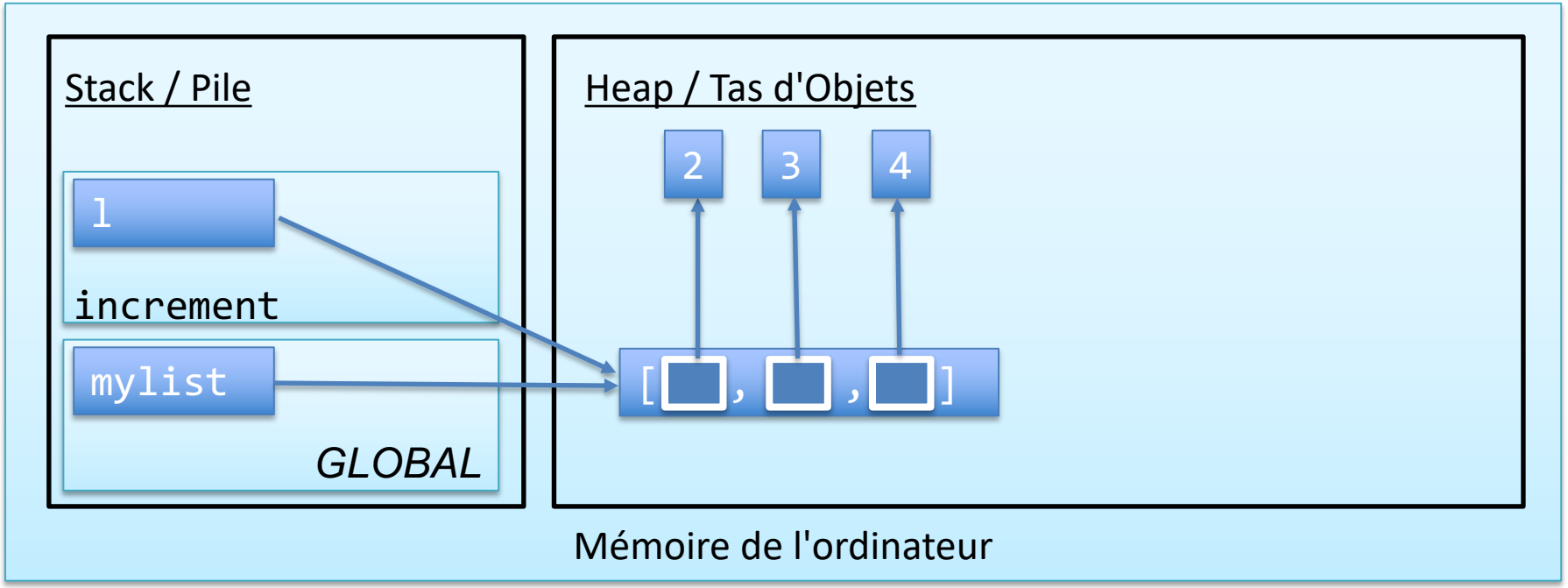
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

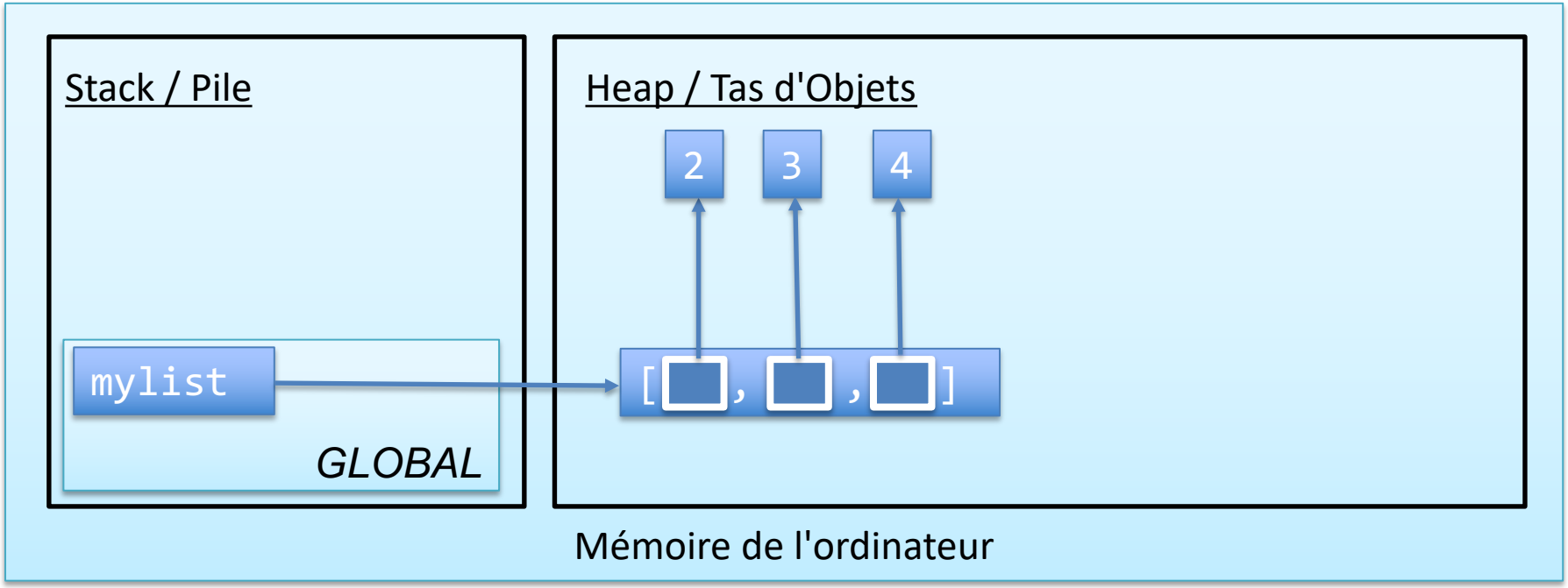
# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

# Exécution: Pile + Tas d'Objets




```
def increment ( l ):
  for i in range ( len ( l ) ):
    l[i] = l[i] + 1

mylist = [ 1, 2, 3 ]
increment ( mylist )
print ( mylist )
```

# Listes imbriquées

- Nous pouvons stocker n'importe quel type d'objet dans une liste
- On peut aussi stocker des listes dans une liste!
- Exemple: stocker une matrice / tableau

```
matrix = [ [ 1.0, 2.0 ,3.0 ], [ 3.0, 4.0, 5.0] ]
matrix[0][1] == 2.0, matrix[1][0] == 3.0
```

| | | |
|---|---|---|
| 1.0 | 2.0 | 3.0 |
| 3.0 | 4.0 | 5.0 |

# Création de Matrices

| | | |
|---|---|---|
| 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 |

```
matrix = [ [ 0.0, 0.0 , 0.0 ], [ 0.0, 0.0, 0.0] ]
```

```
matrix = []
for i in range(2):
  row = []
  for j in range(3):
    row.append(0.0)
  matrix.append ( row )
```

matrix=[] | i=0 | row = [] | j=0

row = [0.0] | j=1

row = [0.0,0.0] | j=2

row = [0.0,0.0,0.0]

# Mission 4

- Objectifs
  - chaînes
  - listes
  - matrices
- Problème
  - Manipuler des chaînes d'ADN