

Informatique 1
Introduction à la programmation

Mission 4 : restructuration

Kim Mens Charles Pecheur Cristel Pelsser

Séquences

Chaînes de caractères (Strings)

```
"Louvain-la-Neuve"
'Louvain-la-Neuve'
'"Louvain-la-Neuve"'
"'Louvain-la-Neuve'"
"louvain-la-neuve"
"ACTGGT"
"actggt"
"a"
```

Listes

```
[0, 1, 2, 3, 4, 5]
[0, 2, 4, 10, 12]
["Louvain", "Ans" ]
["Louvain"]
["Louvain", 1, 2.0]
[[1,2],[3,4]]
```

Séquences : éléments

Chaînes de caractères (Strings)

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

```
s = "LOUVAIN"
for x in s: print (x)
```

```
L
O
U
V
A
I
N
```

Listes

1	9	3	5	9
0	1	2	3	4

```
L = [1,9,3,5,9]
for x in l: print (x)
```

```
1
9
3
5
9
```

Séquences : Indices

Chaînes de caractères (Strings)

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

```
s = "LOUVAIN"
for x in range(len(s)):
    print (x)

?
```

Listes

1	9	3	5	9
0	1	2	3	4

```
l = [1,9,3,5,9]
for x in range(len(l)):
    print (x)

?
```

Séquences

Chaînes de caractères (Strings)

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

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

[0,1,2,3,4,5,6]

```
for x in range(len(s)):
    print (  s[x] )
```

Index dans la liste

Listes

1	9	3	5	9
0	1	2	3	4

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

```
for x in range(len(l)):
    print ( l[x] )
```

Séquences

Chaînes de caractères (Strings)

Listes

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

1	9	3	5	9
0	1	2	3	4

```
for x in len(s):
    print (x)
```

```
for x in len(l):
    print (x)
```



```
for x in len(s):
TypeError: 'int' object is not
iterable
```

Noms de Variables

Chaînes de caractères (Strings)

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

```
for element in s:
    print (element)
```

```
for index in range(len(s)):
    print (  s[index] )
```

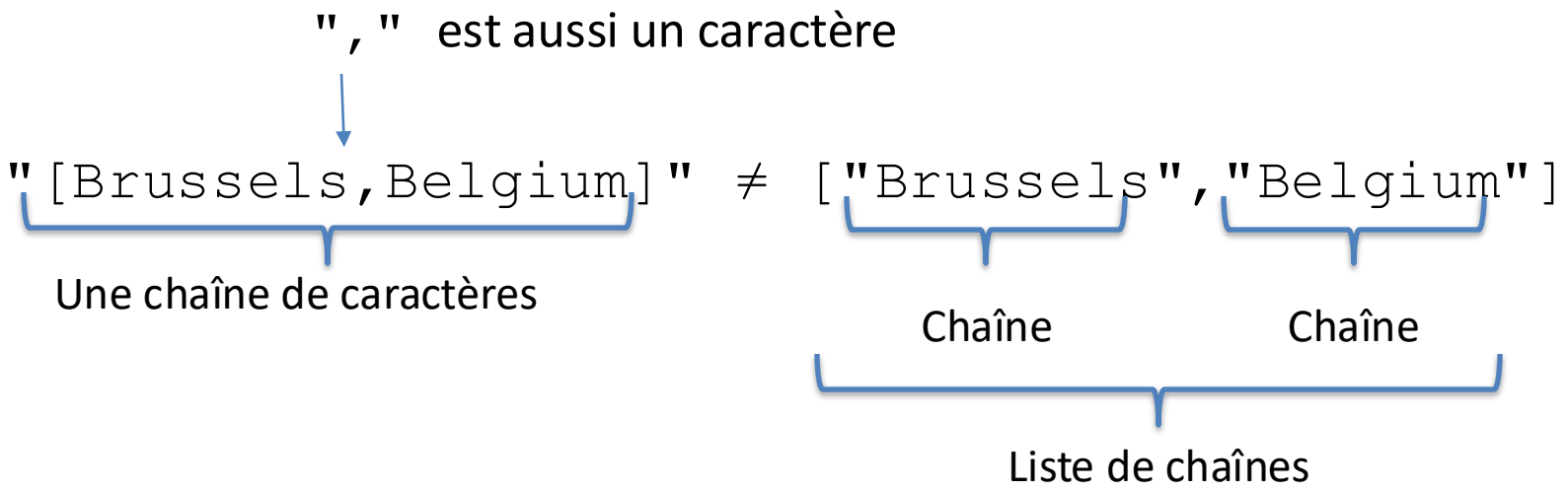
Listes

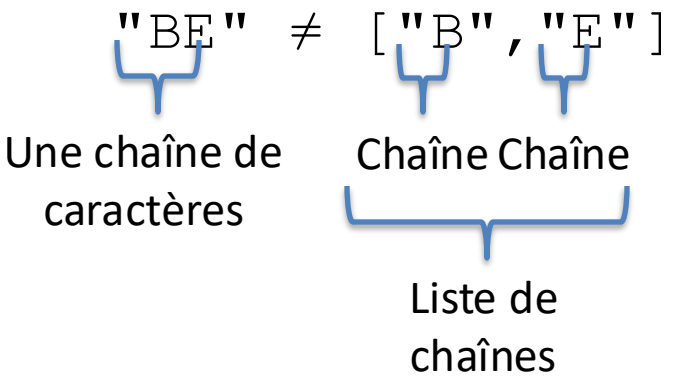
1	9	3	5	9
0	1	2	3	4

```
for element in l:
    print (element)
```

```
for index in range(len(l)):
    print ( l[index] )
```

Séquences


"," est aussi un caractère
"[Brussels,Belgium]" ≠ ["Brussels","Belgium"]
Une chaîne de caractères
Chaîne
Chaîne
Liste de chaînes


"BE" ≠ ["B","E"]
Une chaîne de caractères
Chaîne Chaîne
Liste de chaînes

Séquences

Chaînes de caractères (Strings)

Chaînes ne sont **pas** muables

```
s = ""
s.append ( "a" )
s.append ( "b" )
print ( s )
```



```
'str' object has no
  attribute 'append'
```

Listes

Listes sont muables

```
l = []
l.append ( 1 )
l.append ( 2 )
print ( l )
```

```
[ 1, 2 ]
```

Séquences

Chaînes de caractères (Strings)

Chaînes ne sont **pas** muables

```
s = "Tune"
s[0] = "L"
print ( s )
```



```
'str' object does not
  support item
  assignment
```

Listes

Listes sont muables

```
l = [ 1, 2 ]
l[0] = 3
print ( l )
```

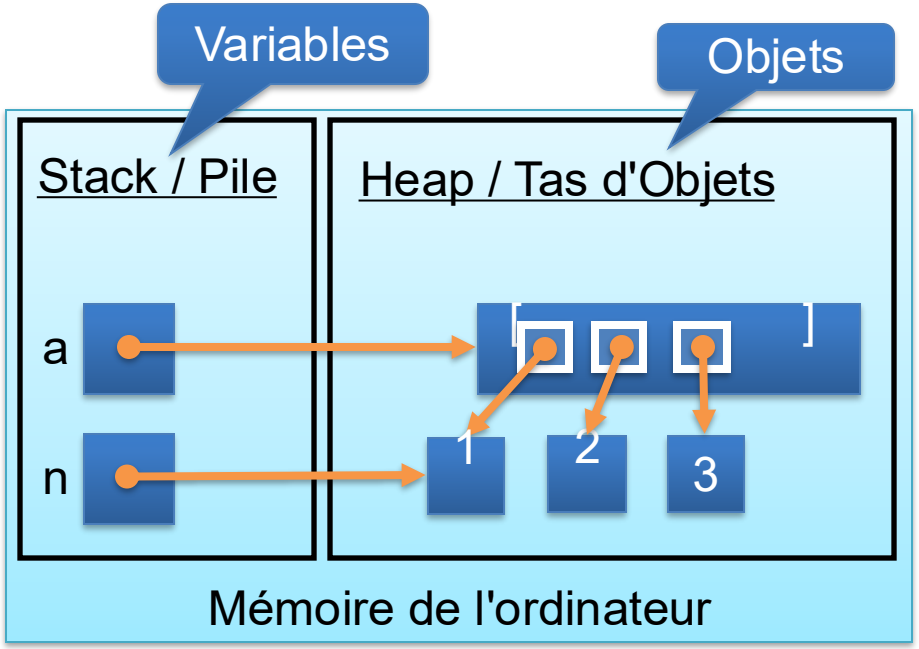
```
[ 3, 2 ]
```

Références



Que se passe-t-il vraiment quand on crée une liste ?

```
a = [1,2,3]
n = 1
```



La variable `a` contient une **référence** (adresse,pointeur) vers la liste (objet)

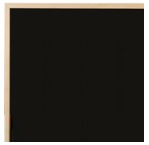
- C'est la même chose pour toutes les valeurs (toutes des objets) !

Modification de Listes: Complications

Qu'est-ce qui est imprimé ici?

```
a = [1,2,3]
b = a
a.append ( 4 )
print(a)
print(b)
```

```
[1,2,3,4]
[1,2,3,4]
```



Modification de Listes: Complications

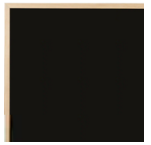
Qu'est-ce qui est imprimé ici?

```
a = [1,2,3]
b = a
a.append ( 4 )
print(a)
print(b)
```

```
[1,2,3,4]
[1,2,3,4]
```

```
a = [1,2,3]
b = a
a = a + [4]
print(a)
print(b)
```

```
[1,2,3,4]
[1,2,3]
```

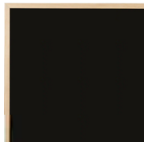


Modification de Listes

Quelle est la différence entre:

```
def add(l):
  for i in range(1000):
    l = l + [i]
l  = []
add ( l )
print ( l )
```

```
def add(l):
  for i in range(1000):
    l.append ( i )
l  = []
add ( l )
print ( l )
```



Opérateurs sur Chaînes de Caractères

Similaire pour les chaînes de caractères

```
a = ""
b = a
for i in range(1000):
  a = a + "*"
```

```
>>>> print(b)

>>>>
```



Chaîne de caractères vide

Enlever des Éléments

- Basé sur index:

```
a = ["Bruxelles","Louvain","Mons"]
del a[1]
print(a)
```

```
["Bruxelles","Mons"]
```

- Basé sur valeur:

```
a = ["Bruxelles","Louvain","Mons"]
a.remove("Mons")
print(a)
```

```
["Bruxelles","Louvain"]
```

Insérer des Éléments*

- Affectation avec des crochets [:] :

```
l = [1, 2, 3]
l[1:1] = [5,6]
print(l)
```

```
[1,5,6,2,3]
```

```
l = [1, 2, 3]
l[3:] = [5,6]
print(l)
```

```
[1,2,3,5,6]
```

Création de Listes Imbriquées

Approche le plus simple

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

Création de Listes Imbriquées

Approche incorrecte

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

  for i in range(2):
    matrix.append ( row )

  return matrix
```

