INFO-H-100 - Informatique

Séance d'exercices 4
Introduction à Python
Slicing et listes

Université Libre de Bruxelles
École polytechnique de Bruxelles

2018-2019

Listes

En Python, une liste est, tout comme un tuple, une séquence d'éléments qui peuvent être de types différents.

Une liste de taille n est indicée de 0 à $n-1$ et de -1 à $-n$ et on peut accéder à chaque élément à l'aide des crochets.

On utilise également les crochets pour construire une liste.

```
>>> li1 = []        #empty list []
>>> type(li1)
<type 'list'>
>>> li1
[]
>>> li2 = [1,2,3,4]
>>> li2
[1, 2, 3, 4]
>>> li2[2]
3
>>> li2[-1]
4
>>> li3 = ["SPAM", True, ('eggs', 42)] #different types of elements
>>> print(li3[2][0][3])
s
```

Indices et slicing

On peut accéder aux éléments d'une séquence (tuples, strings, listes) via l'usage de crochets :

```
>>> mot = "HelloWorld"
>>> mot[0] + ' ' + mot[5]
'H W'
```

Un indice négatif peut être utilisé pour accèder à un élément par rapport à la fin de la séquence :

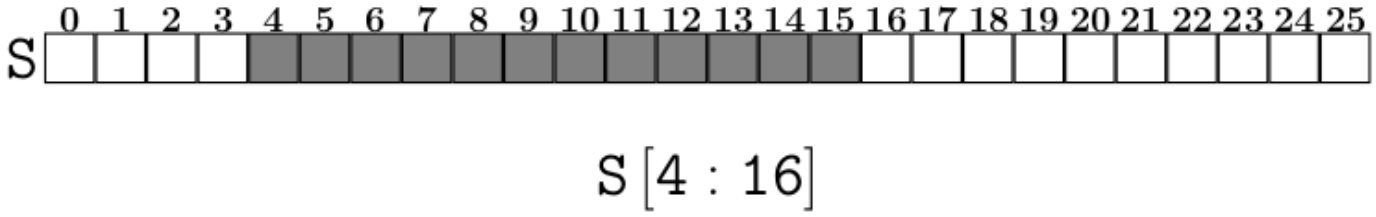
```
>>> mot = "HelloWorld"
>>> mot[-1] + ' ' + mot[-3]
'd r'
```

Les indices positifs (négatifs) vont de 0 à n-1 (-1 à -n)

```
              0       1     2       3
>>> s = ('Hello', ' ', 'World', '!')
             -4      -3    -2      -1
```

Indices et slicing

Le slicing (découpage) $s[a, b]$ permet d'accéder aux éléments d'un séquence s dont les indices vont de a compris jusqu'à b **non compris**



Exemples :

```
>>> date = "18/06/2017"
>>> mois = date[3:5]
>>> print(mois)
06
>>> date[-5:-1]
'/201'
```

Indices et slicing

Si le début ou la fin de la découpe n'est pas précisé, l'extrémité de la séquence est utilisée

```
>>> date = "18/06/2017"
>>> date[:5]
'18/06'
```

N'hésitez pas à expérimenter par vous-même :

```
>>> date[6:2]
???
>>> date[-1: -5]
???
>>> date[3:50]
???
```

Opérations sur les listes

Les listes et les chaînes étant des séquences, certaines opérations sont similaires.

```
>>> li1 = [1,2,3,4]
>>> li2 = [5,6,7,8]
>>> li1 + li2
[1, 2, 3, 4, 5, 6, 7, 8]
>>> li1[1:] + li2[:-1]
[2, 3, 4, 5, 6, 7]
>>> len(li1)
4
>>> 5 in li1
False
>>> li2 = li2 * 3
>>> li2
[5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8]
>>> li2.index(7)
2
>>> li2.count(8)
3
```

Listes et `for`

```
>>> ls = [1, 2, 3]
>>> for x in ls:
        print(x)


1
2
3
```

```
def sum_list(li):
    total = 0
    for item in li:
        total += item
    return total

ls =  [ 1, 2, 3 ]

print(sum_list(ls)) # -> 6
```

Range et for

```
>>> for i in range(3):
        print(i)


0
1
2
```

```
>>> ls = [1, 2, 3]
>>> for i in range(len(ls)):
        print("pos:", i, ", val:", ls[i])

pos: 0 -> val: 1
pos: 1 -> val: 2
pos: 2 -> val: 3
```

Consultez la documentation pour trouver d'autres opérations.

Une liste est une séquence mutable.

Contrairement aux chaînes et aux tuples, une liste est mutable, c'est-à-dire que l'on peut la changer.

```
>>> my_list = [1,7]
>>> my_list[1] = 2
>>> my_list
[1, 2]
```

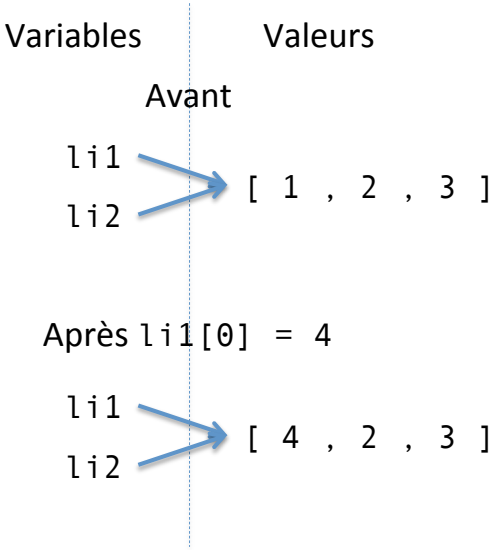
```
>>> my_tuple = (1, 7)
>>> my_tuple[1] = 2
TypeError: 'tuple' object does not support item assignment
```

```
>>> message = "bienvenue"
>>> message[0] = 'B'
TypeError: 'str' object does not support item assignment
```

Une liste est une séquence mutable.

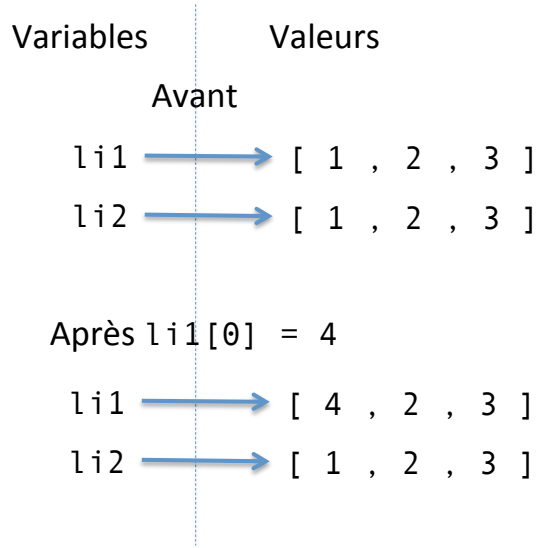
```
>>> li1 = [1,2,3]
>>> li2 = li1
>>> li1[0] = 4
>>> print(li1)
[4, 2, 3]
>>> print(li2)
[4, 2, 3]
>>> li1 == li2
True
```

li2 = li1




```
>>> li1 = [1,2,3]
>>> li2 = li1[:]
>>> li1[0] = 4
>>> print(li1)
[4, 2, 3]
>>> print(li2)
[1, 2, 3]
>>> li1 == li2
False
```

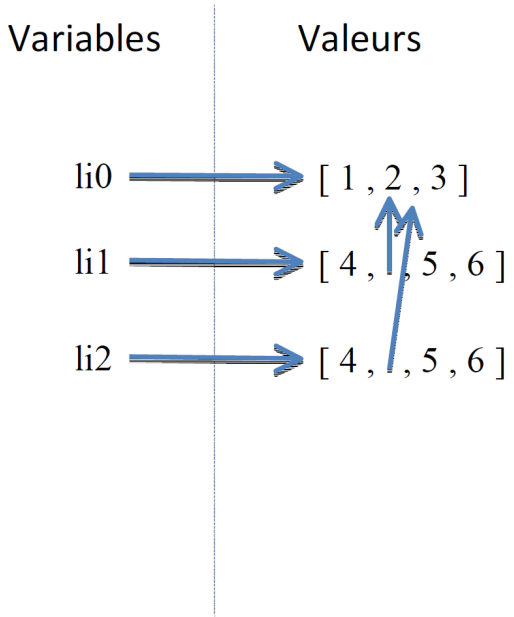
li2 = li1[:]



Une liste est une séquence mutable.

Les copies sont superficielles

```
>>> li0 = [1, 2, 3]
>>> li1 = [4, li0, 5, 6]
>>> li2 = li1[:]          # make a copy
>>> print(li2)
[4, [1, 2, 3], 5, 6]
>>> li1[1][0] = 8         # change li1
>>> print(li2)
[4, [8, 2, 3], 5, 6]      # li2 changed too
```



Nouvelles opérations sur les listes

De par leur mutabilité, les listes possèdent d'autres opérations :

```
>>> li1 = [5,2,6,7,1]
>>> li1.append(9)                    #add an item to the end
>>> li1
[5, 2, 6, 7, 1, 9]
>>> li1.sort()                       #sort the items by ascending order
>>> li1
[1, 2, 5, 6, 7, 9]
>>> li1.insert(2,'eggs')             #insert an item at a given position
>>> print(li1)
[1, 2, 'eggs', 5, 6, 7, 9]
>>> del li1[4]                       #remove an item
>>> li1
[1, 2, 'eggs', 5, 7, 9]
>>> list("SPAM")                     #convert to list
['S', 'P', 'A', 'M']
>>> li1.extend([3, 4, 5])            #extend with another list
>>> li1
[1, 2, 'eggs', 5, 7, 9, 3, 4, 5]
>>> li1.append([6, 7, 8])            # append another list
>>> li1
[1, 2, 'eggs', 5, 7, 9, 3, 4, 5, [6, 7, 8]]
```

Consultez la documentation pour trouver d'autres opérations.

Prochain TP

- Listes et séquences (suite)

Exercices

- 5.b.01, 5.b.02, 5.b.06,5.b.08, 5.8
- 5.12, 5.14, 5.B.2