On découvre

- comment accéder à une valeur donnée d'une séquence (par exemple, un caractère d'une chaîne ou un élément d'une liste)
- comment accéder à une sous-partie d'une séquence

Une chaîne de caractères est une **séquence ordonnée** de caractères, tout comme une liste est une séquence ordonnée d'éléments. Un intervalle est une séquence ordonnée d'entiers.

Les principes présentés ci-dessous s'appliquent aux types `str`, `list` et `range`, qui sont des séquences ordonnées en plus d'être des itérables.

1 Accès à une valeur donnée d'une séquence

Chacun des éléments d'une séquence peut être désigné par sa place dans la séquence à l'aide d'un **indice** (on parle aussi d'index).

L'**indiçage** ou **indexation** revient à numéroter les éléments de la séquence **à partir de 0** (et non à partir de 1).

Par exemple pour la chaîne `"Informatique"`, de longueur `12` :

I	n	f	o	r	m	a	t	i	q	u	e
0	1	2	3	4	5	6	7	8	9	10	11

- le caractère `'r'` est à l'indice `4`,
- le premier caractère `'I'` est à l'indice `0`,
- le caractère `'i'` est à l'indice `8`,
- le dernier caractère de la chaîne, `'e'`, est à l'indice `11`, donc `len("informatique")-1`.

Par exemple pour la liste `[12.0, 10.5, 9.5, 19., 14.5]` de longueur `5` :

12.0	10.5	9.5	19.	14.5
0	1	2	3	4

- le premier élément est à l'indice `0`,
- le dernier élément est à l'indice `4`, soit `len([12.0, 10.5, 9.5, 19., 14.5])-1`.

On accède à l'élément d'une séquence via son indice avec la syntaxe suivante :

`<sequence>[<indice>]`

Par exemple :

```
>>> mot = "Informatique"
>>> mot[8]
'i'
>>> mot[11]
'e'
>>> mot[len(mot) - 1]
'e'
>>> notes = [12.0, 10.5, 9.5, 19., 14.5]
>>> notes[0]
12.0
>>> intervalle = range(2, 10) # 2, 3, 4, 5, 6, 7, 8, 9, longueur 8
>>> intervalle[0]
2
>>> intervalle[7]
9
```

Toute tentative d'accès à une valeur par un indice qui est en dehors des indices de la séquence se traduit par une `IndexError` (erreur d'index) :


```
>>> mot[len(mot)]
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
IndexError: string index out of range
```

2 Accès avec un indice négatif

Python autorise l'indiçage des valeurs d'une séquence à partir de la fin de la séquence.

Ces indiçages se font avec des valeurs d'indice négatives.

L'indice `-1` est utilisé pour désigner la dernière valeur, l'indice `-2` pour désigner l'avant dernière, ainsi de suite jusqu'à la valeur de `-len(sequence)` pour désigner la première valeur.

Par exemple :

I	n	f	o	r	m	a	t	i	q	u	e
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> mot = "Informatique"
>>> mot[-1]
'e'
>>> mot[-2]
'u'
>>> mot[-len(mot)]
'I'
```

De même que pour les indexations avec une valeur d'index positive, une erreur se produit si l'index ne désigne pas un élément de la séquence.

```
>>> mot[-len(mot)-1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

3 Autres indiçages

L'accès à une valeur d'une séquence par indiçage est une opération courante dans les langages de programmation. Python propose d'autres indiçages.

3.1 Accès à une sous-partie d'une séquence

Une **tranche**, ou sous-partie, d'une séquence est une **nouvelle séquence** qui contient une séquence d'éléments de la séquence initiale.

La syntaxe est la suivante :

`<sequence>[<indice_debut_inclus>: <indice_fin_exclu>]`

Les éléments de cette nouvelle séquence, tranche de `<sequence>`, sont les éléments consécutifs de `<sequence>` :

- dont le premier élément est à l'indice `<indice_debut_inclus>`
- mais dont le dernier élément est celui qui, dans `<sequence>`, précède celui d'indice `<indice_fin_exclu>` (s'il existe).

L'indice `<indice_fin_exclu>` est **exclu** de la tranche.

Par exemple : `"Informatique"[0:4]` est une séquence de type `str` qui contient les caractères présents dans `"Informatique"` aux indices de `0` à `3`, c'est à dire `"Info"`.

Par exemple :

```
>>> mot = "Informatique" # séquence de type str
>>> mot[0:4] # tranche de type str
'Info'
>>> des_mots = ["une", "liste", "Python"] # séquence de type list
>>> des_mots[0:2] # séquence de type list
['une', 'liste']
```

On retiendra : `<sequence>[<debut_inclus>: <fin_exclu>]`.

Valeurs par défaut

Les indices de la tranche peuvent être omis.

`<sequence>[:<fin_exclu>]` est équivalent à `<sequence>[0:<fin_exclu>]` : la tranche commence au début de `<sequence>`.

```
>>> mot[:4]
'Info'
```

`<sequence>[<debut_inclus>:]` est équivalent à `<sequence>[<debut_inclus>:len(<sequence>)]` : la tranche termine en fin de `<sequence>`.

```
>>> mot[4:]
'rmatique'
```

Donc `<sequence>[:]` contient tout `<sequence>`.

Si l'indice de fin indiqué pour une sous-partie est supérieur au plus grand indice de la séquence, cela ne produit pas d'erreur. La sous-partie s'arrête à la dernière valeur de la séquence.

```
>>> mot[4:len(mot)]
'rmatique'
>>> mot[len(mot)]
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
IndexError: string index out of range
>>> mot[4:40]
'rmatique'
```

3.2 Accès à une tranche / sous-partie d'une séquence avec un pas

Le **pas** est la différence entre deux indices consécutifs de la tranche / sous-partie.

Un pas qui vaut `n` extrait une valeur sur `n` de la séquence initiale.

Par exemple un pas de 2 permet d'extraire de la chaîne `"Informatique"` les caractères présents à des indices pairs. On obtiendra la nouvelle séquence `'Ifraiu'` :

I	n	f	o	r	m	a	t	i	q	u	e
0		2		4		6		8		10	

La syntaxe est la suivante :

`<sequence>[<indice_debut_inclus>: <indice_fin_exclu>: <pas>]`

Par exemple, la tranche `<sequence>[::2]` (équivalent à `<sequence>[0:len(<sequence>):2]`) contient les éléments d'indice 0, 2, 4 etc de `<sequence>`.

```
>>> "Informatique"[::2]
'Ifraiu'
>>> "J'aime programmer"[::2]
'Jam rgamr'
```

On peut préciser les indices de début (inclus) et fin (**exclu**) :


```
>>> "Informatique"[0:4:2]
'If'
```

Le pas peut-être omis, il vaut alors 1. La tranche contient des valeurs qui sont consécutives dans la séquence.

`<sequence>[<debut_inclus>:<fin_exclu>]` est équivalent à `<sequence>[<debut_inclus>:<fin_exclu>:1]`

Un pas de -1 permet de parcourir la séquence de la fin vers le début.

```
>>> 'Informatique'[::-1]
'euqitamrofnI'
```

Dans le cas d'un pas négatif, la valeur de début est supérieure à celle de fin. Par exemple pour `'Informatique'[7:4:-1]` :

I	n	f	o	r	m	a	t	i	q	u	e
					5	6	7				

```
>>> 'Informatique'[7:4:-1]
'tam'
```

Si l'indice de début de tranche n'est pas supérieur à l'indice de fin de tranche, on obtient une séquence vide.

```
>>> 'Informatique'[2:6:-1]
''
```

3.3 Le cas du type `range`

Comme on l'a déjà vu l'itérable `range(fin)` contient une **séquence d'entiers** qui commence à 0 inclus et termine à `fin` exclu. Par exemple :

```
>>> list(range(5))
[0, 1, 2, 3, 4]
```

La séquence `range(debut, fin)` contient une séquence d'entiers qui commence à `debut` inclus et termine à `fin` exclu. Par exemple :

```
>>> list(range(2, 5))
[2, 3, 4]
```

Enfin la syntaxe du constructeur du type `range` inclus le pas : `range(debut, fin, pas)` indique le pas entre 2 valeurs dans l'intervalle. Par exemple :

```
>>> list(range(5, 2, -1))
[5, 4, 3]
```

ou

```
>>> list(range(0, 10, 2))
[0, 2, 4, 6, 8]
```

On retrouve les éléments de début, fin et le pas déjà vus pour les tranches de séquences. On notera la différence de syntaxe avec les tranches (`:` pour les tranches vs `,` pour le range).

Le `pas` est facultatif, la valeur par défaut est 1.

Le `debut` aussi est facultatif, la valeur par défaut est 0.

Si deux paramètres sont indiqués, il s'agit toujours du `debut` et de la `fin` de la séquence. Si un paramètre est indiqué, il s'agit toujours de la `fin`.

Ainsi, `range(3, 5)` désignera la séquence d'entiers 3, 4 et `range(5)` désignera la séquence d'entiers 0, 1, 2, 3, 4.