スライス

インデックスの次に学習するのは、**スライス**（*slice*）です。文字列の部分を、連続あるいは一定周期で新しい文字列として取り出すものであり、2種類の形式があります。

スライス式

`s[i:j]` … `s[i]` から `s[j-1]` までの並び
`s[i:j:k]` … `s[i]` から `s[j-1]` までの `k` ごとの並び

スライス演算子を使った**スライス式**（*slicing*）は、`[ ]` の中に1個または2個のコロン `:` が入る形式です。指定するのは、以下の値です。

開始 `i` … 取り出す範囲の先頭要素のインデックスに相当
終了 `j` … 取り出す範囲の末尾要素の次の要素のインデックスに相当
ステップ `k` … 取り出す際の刻み幅

取出しは、"`s[j]` まで"ではなく、"`s[j]` の直前の要素まで"です（Column 6-1）。
また、`k` が負であれば、末尾から先頭側へと逆方向に取り出されます。

それでは、実際に確かめてみます。アルファベット大文字26個が並んだ文字列 `s` からの取出しを行います。右側の図と見比べながら、理解しましょう。

例 6-1　文字列とスライス式

```
>>> s = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> s[0:6]
'ABCDEF'
>>> s[0:10:2]
'ACEGI'
>>> s[5:20:3]
'FILOR'
>>> s[12:5:-1]
'MLKJIHG'
```

```
01234567890123456789012345
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

さて、`i`, `j`, `k` の指定には、次の規則が適用されます。

- `i` と `j` は、`len(s)` よりも大きければ、`len(s)` が指定されたものとみなされる。
  インデックスとは異なり、正当な範囲外の値を指定してもエラーとならない。
- `i` が省略されるか `None` であれば、`0` が指定されたものとみなされる。
- `j` が省略されるか `None` であれば、`len(s)` が指定されたものとみなされる。

複雑に感じられるでしょうが、実は、この規則のおかげで、極めて簡潔な指定が行えるようになっています。いくつかの例で見ていきます。

▪ 全要素

全要素を取り出すスライスは、`i` が `0` で、`j` が `len(s)` である `s[0:26]` です。ただし、`j` のみを省略した `s[0:]` でも表せますし、さらに `i` も省略した `s[:]` でも表せます。

▪ ある要素から末尾まで

インデックス *i* の要素から末尾までを取り出すには、*j* を省略して、*s*[*i*:] とします。

▪ 末尾の *n* 要素

末尾の *n* 要素を取り出すのは、*s*[-*n*:] となります。

＊

i, *j*, *k* の 1 個以上を省略するパターンのいくつかをまとめると、次のようになります。

s[:]	すべて	*s*[:]	'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
s[:*n*]	先頭の *n* 要素	*s*[:5]	'ABCDE'
s[*i*:]	*s*[*i*] から末尾まで	*s*[5:]	'FGHIJKLMNOPQRSTUVWXYZ'
s[-*n*:]	末尾の *n* 要素	*s*[-5:]	'VWXYZ'
s[::*k*]	*k* - 1 個おき	*s*[::3]	'ADGJMPSVY'
s[::-1]	すべてを逆向き	*s*[::-1]	'ZYXWVUTSRQPONMLKJIHGFEDCBA'

▶ *n* が要素数を超える場合は、全要素が取り出されます。

インデックス式とスライス式は、次章以降で学習する《リスト》や《タプル》などでも利用されます。

さて、文字列はイミュータブルであって、値は変更できません。そのため、代入の左辺にインデックス式やスライス式を置くと、エラーになります。

例 6-2　文字列 'ABCDEF…XYZ' の 'F' を 'X' に書きかえた文字列を生成

誤
```
>>> s = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'↵
>>> s[5] = 'X'↵                          ← 左辺はＮＧ
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

s[5] の文字のみを書きかえて生成
ABCDEXGHIJKLMNOPQRSTUVWXYZ

正
```
>>> s = s[:5] + 'X' + s[6:]↵    ← 右辺はＯＫ
>>> s↵                          ← sは新しく生成した文字列を参照する
'ABCDEXGHIJKLMNOPQRSTUVWXYZ'
```

'F' を 'X' に書きかえようとして、*s*[5] への代入を行いますが、エラーが発生します。

'E' までのスライスと、'X' と、'F' 以降のスライスを連結させることで、うまくいきます。

Column 6-1　range とスライス式における指定値について

第 4 章で学習した range(*n*) と range(*a*, *b*) は、生成する数列の最終値が *n* - 1 と *b* - 1 であり、実引数で指定した値である *n* あるいは *b* 自身は含まれません（一つ手前までとなります）。ここで学習しているスライス式 *s*[:*n*] と *s*[*a*:*b*] も同様です。

このような仕様となっているのは、以下のように便利だからです。

- *n* そのもの、あるいは *b* - *a* によって、長さ（要素数）が分かる。
- range(*a*, *b*) と range(*b*, *c*)、あるいは、*s*[:*n*] と *s*[*n*:] によって、途切れることなく、かつ、重複することなく要素を列挙できる。