Les chaînes de caractères

S. Vialette

IGM-LabInfo, Université Paris-Est

21 janvier 2008

Introduction

```
>>> seq1 = "aagagacgtatttagagagatttaaaaagtgaaagtgtg"
>>> seq1
'aagagacgtatttagagagatttaaaaagtgaaagtgt'
>>> seq2 = "aggagag"
>>> seq2
'aggagag'
>>> seq1 + seq2
'aagagacgtatttagagagatttaaaaagtgaaagtgtgaggagag'
>>> seqFinal = seq1 + seq2
>>> seqFinal
'aagagacgtatttagagagatttaaaaagtgaaagtgtgaggagag'
>>> len(seqFinal)
46
>>>
```

Introduction

```
>>> seq = ""
>>> len(seq)
0
>>> seq = "aagagacgtatttagagg"
>>> 'a' in seq
True
>>> 'b' in seq
False
>>> '' in seq
True
>>> seq == "aagagacgtatttagagg"
True
```

Introduction

```
>>> seq = "aagagacgtatttagagg"
>>> import string
>>> string.count(seq, 'a')
7
>>> count(seq, 'a')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'count' is not defined
>>> from string import count
>>> count(seq, 'a')
7
>>> replace(seq , 'a', 'A')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'replace' is not defined
>>> from string import *
>>> replace(seq , 'a', 'A')
'AAgAgAcgtAtttAgAgg'
>>> seq
'aagagacgtatttagagg'
```

Accès aux éléments

```
>>> seq = "abcdefgh"
>>> seq[0]
'a'
>>> seq[len(seq)-2]
'g'
>>> seq[len(seq)-1]
'h'
>>> seq[-1]
'h'
>>> seq[-2]
'g'
>>> seq[-len(seq)]
'a'
>>> seq[len(seq)]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>>
```

Les tranches

```
>>> seq = "abcdefgh"
>>> seq[1:4]
'bcd'
>>> seq[1:]
'bcdefgh'
>>> seq[:4]
'abcd'
>>> seq[:]
'abcdefgh'
>>> seq[:-1]
'abcdefg'
>>> seq[1:len(seq)+1]
'bcdefgh'
>>> seq[1:len(seq)]
'bcdefgh'
>>> seq[5:4]
''
>>> seq[len(seq):len(seq)+1]
''
>>>
```

Les chaînes de caractères sont immutables

```
>>> seq = "abcdefgh"
>>> seq[0] = 'A'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> seq[1:3] = 'x'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> newSeq = "ABC" + seq[:3]
>>> newSeq
'ABCabc'
>>> newSeq = seq[:3] + "XXX" + seq[3:]
>>> newSeq
'abcXXXdefgh'
>>>
```

Quelques fonctions sur les chaînes de caractères

- *myString*`.capitalize()` : Retourne une copie de *myString* dans laquelle la première lettre est en majuscule et les autres en minuscules.
- *myString*`.lower()` : Retourne une copie de *myString* dans laquelle toutes les lettres sont en minuscules. Il existe également la fonction inverse *myString*`.upper()`.
- *myString*`.strip(`*chars*`)` : Retourne une copie de *myString* dans laquelle les premières et dernières occurrences des caractères spécifiés ,*i.e.*, *chars* , ont été supprimés. Si *chars* n'est pas spécifié, la fonction retourne une copie de *myString* dans laquelle les premier et derniers blancs ont été supprimés.

Quelques fonctions sur les chaînes de caractères

```
>>> myString = "abCd"
>>> myString.capitalize()
'Abcd'
>>> myString
'abCd'
>>> myString.lower()
'abcd'
>>> myString.upper()
'ABCD'
>>> myString.isupper()
False
>>> myString.upper().isupper()
True
>>> myString = myString.upper()
>>> myString.isupper()
True
```

Quelques fonctions sur les chaînes de caractères

```
>>> myString = "␣␣␣abc␣␣␣␣␣"
>>> myString.strip()
'abc'
>>> myString.strip().upper()
'ABC'
>>> myString
'␣␣␣abc␣␣␣␣␣'
>>> myString = "www.univ-mlv.fr"
>>> myString.strip("w")
'.univ-mlv.fr'
>>> myString.strip("w.fr")
'univ-mlv'
>>> myString = "aabbbaaaa"
>>> myString.strip("a")
'bbb'
>>> myString[2:].strip("a")
'bbb'
```

Rechercher dans des chaînes de caractères

- *myString*.`find(`*sub*`[,`*start*`[,`*end*`]])` : Retourne la plus petite position à laquelle *sub* apparaît dans la tranche *[start,end]* de *myString*. Retroune -1 si *sub* n'apparaît pas dans la tranche *[start,end]* de *myString*.
- *myString*.`find(`*sub*`[,`*start*`[,`*end*`]])` : Équivalent à *myString*.`find(`*sub*`[,`*start*`[,`*end*`]])` mais lève l'exception `ValueError` si *sub* n'apparaît pas dans la tranche *[start,end]* de *myString*.
- *myString*.`replace(`*old*`,`*new*`, [`*count*`])` : Retourne une copie de *myString* dans laquelle toutes les occurrences de *old* sont remplacées par *new*. Si l'argument optionel *count* est spécifié, seules les *count* premières occurrences sont remplacées.

Rechercher dans des chaînes de caractères

```
>>> myString = "abcabcabc"
>>> myString.find("ab")
0
>>> myString.find("ab", 1)
3
>>> myString.find("ab", 5)
6
>>> myString.find("ab", 8)
-1
>>> myString.find("ab", 1, 3)
-1
>>> mySubstring = "ab"
>>> myString.find(mySubstring, 1)
3
>>>
```

Rechercher dans des chaînes de caractères

```
>>> myString = "abcabcabc"
>>> mySubstring = "abc"
>>> myString.find("ab")
0
>>> myString.find(mySubstring, 2)
3
>>> myString.index(mySubstring, 2)
3
>>> myString.find(mySubstring, 1, 4)
-1
>>> myString.index(mySubstring, 1, 4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
>>>
```

Rechercher dans des chaînes de caractères

```
>>> myString = "abacabac"
>>> myString.replace("aba", "AABBAA")
'AABBAAcAABBAAc'
>>> myString.replace("aba", "AABBAA", 1)
'AABBAAcabac'
>>> myString
'abacabac'
>>> myString = "abababababa"
>>> myString.replace("aba", "AABBAA")
'AABBAAbAABBAAbAABBAA'
>>>
```

En vrac ...

```
>>> "abcde" * 2
'abcdeabcde'
>>> "abcde" * 3
'abcdeabcdeabcde'
>>> "abcde".index('c')
2
>>> "abcde".index("c")
2
>>> "abcde".startswith("ab")
True
>>> "abcde".startswith("ac")
False
>>> "abcde".endswith("cde")
True
>>> "abcde".endswith("e")
True
>>> "abcde".endswith("ae")
False
>>>
```

En vrac ...

```
>>> myString = "␣␣␣"
>>> myString.isspace()
True
>>> "abcde".isspace()
False
>>> "aBcDeF".swapcase()
'AbCdEf'
>>> "voici␣un␣titre".title()
'Voici␣Un␣Titre'
>>> "voici␣un␣titre".istitle()
False
>>> "voici␣un␣titre".title().istitle()
True
>>>
```

Classe

- Python est un langage objet . . .

```
>>> type("abc")
<type 'str'>
>>> type("abc" + "def")
<type 'str'>
>>> "abc".__class__
<type 'str'>
>>>
```

Tout connaître . . .

- `dir(obj)` : retourne tous les noms définis dans l'espace de nommage local de l'objet obj.

```
>>> dir("abc")
['__add__', '__class__', '__contains__', '__delattr__', '__doc__',
 '__eq__', '__ge__', '__getattribute__', '__getitem__',
 '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__',
 '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__',
 '__rmul__', '__setattr__', '__str__', 'capitalize', 'center',
 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find',
 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace',
 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition',
 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip',
 'swapcase', 'title', 'translate', 'upper', 'zfill']
```