Символни низове

д-р Филип Андонов

10 юни 2022 г.

Съдържание

Низове

Едноредовите символни низове могат да бъдат оградени в единични или двойни кавички.

```
single='test'
single2="test"
```

Многоредовите символни низове са оградени с три двойни кавички

```
multiline="""
line 1
line 2
line 3
"""
```

Низове

Всички стандартни оператори за последователности (индексиране, срязване, мултиплициране, принадлежност, дължина, минимум и максимум) работят със символни низове. Но низовете се неизменими, така че всякакви присвоявания на отрязъци не са допистими:

```
>>> website = 'http://www.python.org'
>>> website[-3:] = 'com'
Traceback (most recent call last):
File "<pyshell#19>", line 1, in ?
website[-3:] = 'com'
TypeError: object doesn't support slice
    assignment
```

Форматиране

Само кортежи и речници ви позволяват да форматирате повече от една стойност.

```
>>> formattedString="Hello! I study %s in %s"
>>> values = ("computer science", "NBU")
>>> print(formattedString % values)
Hello! I study computer science in NBU
```

str vs bytes

Python 3 има два типа за символни низове - един за Unicode и един за байтове. Типът str съхранява Unicode низове, а типът bytes - байтове. Като цяло ние искаме да съхраняваме низовете като Unicode, а да работим с байтове само при взаимодействие на програмата с външния свят. Препоръчителната стратегия е да декодираме байтовете възможно най-рано, превръщайки ги в Unicode с метода decode и ако се налага да ги изведем от програмата като байтове, да го направим възможно най-късно, като подадем низа като аргумент на функцията bytes.

str vs bytes

Важно е да се запомни, че Python 3 не преобразува автоматично байтовите низове. Ако се опитаме да комбинираме байтов низ с Unicode низ, ще получим грешка. Python 3 не иска да налучква типа на кодирането, а изисква от нас да го укажем явно. За целта е необходимо да знаем дадена редица от битове по какъв начин е била кодирана (типът bytes не съхранява никаква мета-информация за използването кодиране).

Форматиране

Спецификатори на конвертирането:

- Символът % . Обозначава началото на спецификаторите на конвертирането

Флагове (незадължителни):

- - ляво подравняване;
- +, стойността трябва да е предшествана от знак
- “ ” (интервал), положителните числа трябва да са предшествани от интервал

Форматиране

Спецификатори на конвертирането:

- Минимална дължина (незадължителна). Конвертираният низ ще е поне толкова дълъг. Ако е символът за звезда *, дължината ще е прочетена от кортежа
- Точка . Последвана от точност (незадължителен). Ако се конвертира реално число, толкова знака след десетичната точка ще бъдат показани. Ако това е низ, това число ще бъде максималната дължина на полето. Ако точността е означена със звезда, то точността ще бъде прочетена от кортежа със стойности
- Тип

Просто конвертиране

```
>>> "The answer is %d" % 42
'The answer is 42'
>>> "65536 in HEX is %x" % 65536
'65536 in HEX is 10000'
```

Форматиране

Тип	Смисъл
d,i	Цяло число със знак
o	Осмична стойност без знак
u	Десетична стойност без знак
x	Шестнайсетична стойност без знак в долен регистър
X	Шестнайсетична стойност без знак в горен регистър
e	Число с плаваща запетая в експоненциален формат, долен регистър
E	Число с плаваща запетая в експоненциален формат, горен регистър
f, F	Число с плаваща запетая в десетичен формат
c	Символ
r	Символен низ (използва repr)

Таблица: Спецификатори на форматирането

Просто конвертиране

```
1 >>> "%10f" % pi
2 '  3.141593'
3 >>> "%10.2f" % pi
4 '      3.14'
5 >>> "%.2f" % pi
6 '3.14'
7 >>> '%.5s'%'Albus Percival Wulfric Brian
      Dumbledore'
8 'Albus'
```

Просто конвертиране

```
1 >>> "%05.2f" % pi
2 '03.14'
3 >>> "%-10.2f" % pi
4 '3.14      '
5 >>> "% 5d" % 10
6 '   10'
7 >>> "% 5d" % -10
8 '  -10'
```

Просто конвертиране

```
1 >>> "%+5d" % 10
2 '  +10'
3 >>> "%+5d" % -10
4 '  -10'
```

Пример

```
1  pwidth = 10
2  iwidth = 30
3  header = "%-*s%*s"
4  item = "%-*s %*.2f"
5  print("=" * 40)
6  print(header % (iwidth,"item",pwidth,"price"))
7  print("-" * 40)
8  print(item % (iwidth,'coffee',pwidth,1.5))
9  print(item % (iwidth,'Muffin',pwidth,3.5))
10 print(item % (iwidth,'Ice Latte',pwidth,6.8))
11 print("="* 40)
```

Нов стил форматиране

В Python 3 бе въведено нов вид форматиране на низове. То не използва оператора %. Вместо това се използва метод .format() върху обект от тип символен низ.

Нов стил форматиране

Позиционно форматиране, подобно на стария стил:

```
'Hello, {}'.format(name)
'Hello, Bob'
```

Можем обаче да използваме и имена, за да ги използваме в какъвто искаме ред. Това позволява да се променя подредбата, която се извежда, без да се променят аргументите, подадени на format().

```
>>> 'Hey {name}, there is a 0x{errno:x} error!'.
    format(
...         name=name, errno=errno)
'Hey Bob, there is a 0xbadc0ffee error!'
```

Модула string

име	смисъл
string.digits	Всички цифри от 0 до 9
string.ascii_letters	Всички букви – главни и малки
string.ascii_lowercase	Всички малки букви
string.ascii_uppercase	Всички главни букви
string.printable	Всички печатими символи
string.punctuation	Всички пунктуационни знаци

Таблица: Стойности от модула string

count(sub[, start[, end]])

Връща броя на срещанията на подниза sub в низа S[start:end]. Незадължителните параметри start и end се интерпретират като срез.

find(s[, start[, end]])

Методът find търси подниз в по-голям низ. Връща индексът на най-лявото появяване на подниза. Ако не е намерен, връща -1

```
>>> title="Buy cheap $$VIAGRA$$ NOW!!!"
>>> title.find("viagra")
-1
>>> title.find("VIAGRA",2,26)
12
```

isalnum()

Връща True ако всички символи в низа са цифри или букви и има поне един символ, в противен случай връща False

```
1 a="ab2cd57ef"
2 print(a.isalnum())
3 True
```

isalpha()

Връща True ако всички символи в низа са букви и той съдържа поне един символ, в противен случай връща False

isdigit())

Връща True ако всички символи в низа са цифри и той съдържа поне един символ, в противен случай връща False

islower()

Връща True ако всички дву-регистърни символи в низа са в долен регистър и той съдържа поне един символ, в противен случай връща False

isupper()

Връща True ако всички дву-регистърни символи в низа са в горен регистър и той съдържа поне един символ, в противен случай връща False

join()

Методът join е обратният на split – той обединява няколко символни низа от последователност в един низ

```
>>> dirs = "usr", "share", "lib" , "gnome"
>>> print('\\'.join(dirs))
usr\share\lib\gnome
>>> print('/'.join(dirs))
usr/share/lib/gnome
```

lower()

Методът lower връща версия на символния низ, съставена само от малки букви

```
1 >>> title="Buy cheap ViAgRa here! BEST prices!!!"
2 >>> title.lower().find("viagra")
```

replace()

Методът replace връща символен низ, в който всичките срещания на даден низ са подменени с друг.

```
1 >>> title ="make␣BIG␣$$$␣MONEY␣NOW!!!␣GET␣STNKING
      ␣RICH␣in␣24␣hours␣and␣have␣millions␣of␣$$$!!!"
2 >>> print(title.replace("$","GBP"))
3 make BIG GBPGBPGBP MONEY NOW!!! GET STNKING RICH
      in 24 hours and have millions of GBPGBPGBP!!!
```

split()

Методът split, обратно на join, разделя символния низ на последователност:

```
1 url = "www.kino.dir.bg"
2 if url.split(".")[-1] not in ['com','net','org','
    edu','mil']:
3  print("not a valid URL!!!")
```

strip()

Методът strip връща низ, в който празните интервали отляво и отдясно (но не и вътре в низа) са премахнати:

```
1 >>> name="   Arnold Schwarzenegger  "
2 >>> name.strip()
3 'Arnold Schwarzenegger'
```

translate()

Подобно на replace, translate подменя части на низа, но той работи само с отделни символи. Силата му е в това, че може да извършва няколко заместявания едновременно, при това много ефективно. Но за да работи, ни трябва таблица на заместванията. Тя съдържа 256 символа, затова няма да я пишем, а ще използваме статичния метод maketrans на някой от класовете str, bytes или bytearray. Той приема като аргументи низ със символи които да бъдат заменени, низ със символите, които ги заменят, както и незадължителен параметър за изтриване на символи.

Методи

```
1 import string
2 s = "The␣leet␣hackers!"
3 s=s.translate(str.maketrans("elta","3174"))
4 print(s)
5 s = "string.␣With.␣Punctuation?" # Sample string
6 out=s.translate(str.maketrans("","",string.
      punctuation))
7 print(out)
```

Сравняване

Използва се познатия оператор ==

```
1 if word == 'banana':
2     print('Oo, bananas are great!')
```

Операторът in

Операторът in е булев, като приема два аргумента и връща True ако първият се намира във втория

```
>>> "a" in "banana"
True
>>> "qw" in "banana"
False
```

ASCII

В Python превръщането от ASCII код в символ и обратно се извършва с функции, чийто имена са взаимствани от Pascal.

- ord('A') връща 65, което е ASCII кода на A
- chr(65) връща 'A'

Параметри от командния ред

Параметрите от командния ред се достъпват чрез функционалност, наличност в модула sys.
Те могат да бъдат прочетени в списъка sys.argv
Първият елемент на списъка е името на файла
Първият подаден параметър е на второ място (индекс 1)

Параметри от командния ред

```
1  import sys
2  for arg in sys.argv:
3      print(arg)
4  ~/ python cmdparam.py ala bala nica turska panica
5  cmdparam.py
6  ala
7  bala
8  nica
9  turska
10 panica
```

Шифри със субституция и транспозиция

Преди компютрите, криптографията се е състояла от алгоритми, базирани на символи. Различни криптографски алгоритми или са заменяли един символ с друг, или са размествали символите един с друг. Добрите алгоритми са правели и двете, многократно.

Субституционни шифри

Всеки символ от открития текст се заменя с друг в шифрирания текст. Получателят прилага обратната замяна, за да възстанови нешифрирания текст. В класическата криптография съществуват четири вида субституционни шифри:

Субституционни шифри

- Пряк субституционен шифър – при него всеки символ от открития текст се заменя само с един символ в шифрирания текст.
- Хомофонен субституционен шифър – подобен на прекия, но един символ в открития текст може да съответства на няколко символа в шифрирания текст
- Полиграмен -група символи се шифрират по групи. Например “ABA” може да съответства на “RTQ”, а “ABB” да съответства на “SLL”.
- Полиазбучен - състои се от множество пряки субституционни шифри. Например може да се използват пет пряки шифъра, в зависимост от позицията на символа в открития текст.

Шифър на Цезар

- Използван от Юлий Цезар
- Е пряк субституционен шифър
- Всеки символ от открития текст се заменя със символа на позиция три на дясно по модул 26 (“A” се заменя с “D,” “B” се заменя с “E,”..., “W” се заменя с “Z,” “X” се заменя с “A,” “Y” се заменя с “B,” и “Z” се заменя с “C”)

Лицето, извършващо шифриране, избира ключ – ключ между 1 и 25, определящо отместването, което трябва да бъде използвано при шифрирането на всяка буква.

M	E	E	T	M	E	A	T
P	H	H	W	P	H	D	W

Таблица: Примерно шифриране

Шифърът на Цезар с фиксирано отместване на буквите е съвсем елементарен – трябва просто да изпробваме 25-те възможни ключа.

Шифър на Виженер

- Кръстен на Blaise de Vigenère.
- Изобретен по-рано от Giovan Battista Bellaso.
- Виженер е изобретил по-силния шифър с автоключ.

Криптоанализ

- Публикуван за пръв път през 1586
- Преодолява проблема с честотния анализ, като кодира една и съща буква по различен начин в зависимост от нейната позиция в документа.
- Състои се от няколко шифъра на Цезар в последователна смяна с по един символ.

Ако искаме да шифрираме текста ATTACKATDAWN с ключ LEMON:

- Кодираната азбука получаваме, като изместим обикновената азбука така, че да започва с първата буква от ключа LEMON. Първата буква от открития текст шифрираме с тази азбука.
- Вторият символ от открития текст шифрираме, като отместим обикновената азбука така, че да започва с втория символ в ключа.
- Продължаваме така с всички символи от открития текст, като след като надминем дължината на ключа, започваме отначало (шестата буква ще шифрираме със същата азбука, с която и първата)

Криптоанализ

PLAINTEXT=ATTACKATDAWN
NORMAL=ABCDEFGHIJKLMNOPQRSTUVWXYZ
CAESAR= LMNOPQRSTUVWXYZABCDEFGHIJK
CAESAR= EFGHIJKLMNOPQRSTUVWXYZABCD
CAESAR= MNOPQRSTUVWXYZABCDEFGHIJKL
CAESAR= OPQRSTUVWXYZABCDEFGHIJKLMN
CAESAR= NOPQRSTUVWXYZABCDEFGHIJKLM
CAESAR= LMNOPQRSTUVWXYZABCDEFGHIJK
CYPHERTEXT=LXFOPVEFRNHR

Задачи

- Реализирайте шифъра на Цезар на python
- Реализирайте шифъра на Виженер на python

Благодаря за вниманието!

Въпроси?

References