Python - Sistema Unicode

O que é o sistema Unicode?

Os aplicativos de software geralmente exigem a exibição de mensagens emitidas em vários idiomas, como inglês, francês, japonês, hebraico ou hindi. O tipo string do Python usa o padrão Unicode para representar caracteres. Isso torna possível ao programa trabalhar com todos esses diferentes caracteres possíveis.

Um caractere é o menor componente possível de um texto. 'A', 'B', 'C', etc., são todos caracteres diferentes. O mesmo acontece com 'È' e 'Í'. Uma string Unicode é uma sequência de pontos de código, que são números de 0 a 0x10FFFF (1.114.111 decimal). Esta sequência de pontos de código precisa ser representada na memória como um conjunto de unidades de código, e as unidades de código são então mapeadas para bytes de 8 bits.

Codificação de caracteres

Uma sequência de pontos de código é representada na memória como um conjunto de unidades de código, mapeadas em bytes de 8 bits. As regras para traduzir uma string Unicode em uma sequência de bytes são chamadas de codificação de caracteres.

Estão presentes três tipos de codificações, UTF-8, UTF-16 e UTF-32. UTF significa **Formato de Transformação Unicode** .

Suporte Unicode do Python

Python 3.0 em diante tem suporte integrado para Unicode. O tipo **str** contém caracteres Unicode, portanto, qualquer string criada usando a sintaxe de string entre aspas simples, duplas ou triplas é armazenada como Unicode. A codificação padrão para o código-fonte Python é UTF-8.

Portanto, string pode conter representação literal de um caractere Unicode (3/4) ou seu valor Unicode (\u00BE).

Exemplo

```
var = "3/4"
print (var)
var = "\u00BE"
print (var)
```

Este código acima produzirá a seguinte **saída** -

```
3/4
¾
```

Exemplo

No exemplo a seguir, uma string '10' é armazenada usando os valores Unicode 1 e 0, que são \u0031 e u0030, respectivamente.

```
var = "\u0031\u0030"
print (var)
```

Ele produzirá a seguinte **saída** -

```
10
```

As strings exibem o texto em um formato legível e os bytes armazenam os caracteres como dados binários. A codificação converte dados de uma sequência de caracteres em uma série de bytes. A decodificação traduz os bytes de volta em caracteres e símbolos legíveis por humanos. É importante não

confundir esses dois métodos. encode é um método de string, enquanto decode é um método do objeto de byte Python.

Exemplo

No exemplo a seguir, temos uma variável string que consiste em caracteres ASCII. ASCII é um subconjunto do conjunto de caracteres Unicode. O método encode() é usado para convertê-lo em um objeto bytes.

```
string = "Hello"
tobytes = string.encode('utf-8')
print (tobytes)
string = tobytes.decode('utf-8')
print (string)
```

O método decode() converte o objeto byte de volta no objeto str. O método de codificação usado é utf-8.

```
b'Hello'
Hello
```

Exemplo

No exemplo a seguir, o símbolo da rupia ($) é armazenado na variável usando seu valor Unicode. Convertemos a string em bytes e de volta em str.

```
string = "\u20B9"
print (string)
tobytes = string.encode('utf-8')
print (tobytes)
string = tobytes.decode('utf-8')
print (string)
```

Quando você executa o código acima, a seguinte saída será produzida -

```
₹
b'\xe2\x82\xb9'
₹
```