Python - Modelagem

Python oferece diferentes recursos de formatação de texto. Inclui operadores de formatação, a função format() do Python e a string f. Além disso, a biblioteca padrão do Python inclui um módulo string que vem com mais opções de formatação.

A classe Template no módulo string é útil para formar um objeto string dinamicamente pela técnica de substituição descrita em PEP 292. Sua sintaxe e funcionalidade mais simples facilitam a tradução em caso de internalização do que outros recursos de formatação de string integrados em Python.

As strings de modelo usam o símbolo $ para substituição. O símbolo é imediatamente seguido por um identificador que segue as regras de formação de um identificador Python válido.

Sintaxe

```
from string import Template

tempStr = Template('Hello $name')
```

A classe Template define os seguintes métodos -

substituto()

Este método realiza a substituição de valor dos identificadores no objeto Template. O uso de argumentos de palavra-chave ou um objeto de dicionário pode ser usado para mapear os identificadores no modelo. O método retorna uma nova string.

Exemplo 1

O código a seguir usa argumentos de palavras-chave para o método replace().

```
from string import Template

tempStr = Template('Hello. My name is $name and my age is $age')
newStr = tempStr.substitute(name = 'Pushpa', age = 26)
print (newStr)
```

Ele produzirá a seguinte **saída** -

```
Hello. My name is Pushpa and my age is 26
```

Exemplo 2

No exemplo a seguir, usamos um objeto de dicionário para mapear os identificadores de substituição na string do modelo.

```
from string import Template

tempStr = Template('Hello. My name is $name and my age is $age')
dct = {'name' : 'Pushpalata', 'age' : 25}
newStr = tempStr.substitute(dct)
print (newStr)
```

Ele produzirá a seguinte **saída** -

```
Hello. My name is Pushpalata and my age is 25
```

Exemplo 3

Se o método replace() não for fornecido com parâmetros suficientes para corresponder aos identificadores na string do modelo, o Python gerará KeyError.

```
from string import

tempStr = Template('Hello. My name is $name and my age is $age')
dct = {'name' : 'Pushpalata'}
newStr = tempStr.substitute(dct)
print (newStr)
```

Ele produzirá a seguinte **saída** -

```
Traceback (most recent call last):
File "C:\Users\user\example.py", line 5, in
newStr = tempStr.substitute(dct)
^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Python311\Lib\string.py", line 121, in substitute
return self.pattern.sub(convert, self.template)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Python311\Lib\string.py", line 114, in convert
return str(mapping[named])
~~~~~~~^^^^^^^^
KeyError: 'age'
```

substituto_seguro()

Este método se comporta de forma semelhante ao método replace(), exceto pelo fato de não gerar erro se as chaves não forem suficientes ou não corresponderem. Em vez disso, o espaço

reservado original aparecerá intacto na string resultante.

Exemplo 4

```
from string import Template
tempStr = Template('Hello. My name is $name and my age is $age')
dct = {'name' : 'Pushpalata'}
newStr = tempStr.safe_substitute(dct)
print (newStr)
```

Ele produzirá a seguinte **saída** -

```
Hello. My name is Pushpalata and my age is $age
```

é válido()

Retornará falso se o modelo tiver espaços reservados inválidos que farão com que replace() gere ValueError.

get_identifiers()

Retorna uma lista dos identificadores válidos no modelo, na ordem em que aparecem pela primeira vez, ignorando quaisquer identificadores inválidos.

Exemplo 5

```
from string import Template

tempStr = Template('Hello. My name is $name and my age is $23')
print (tempStr.is_valid())
tempStr = Template('Hello. My name is $name and my age is $age')
print (tempStr.get_identifiers())
```

Ele produzirá a seguinte **saída** -

```
False

['name', 'age']
```

Exemplo 6

O "" $simm\ b\ o\ l\ foi\ o\ fime\ feu\ n\ e\ dcomo\ o\ personagem\ substituto$ para a string, ele deve ser escapado. Em outras palavras, use $$ para usá-lo na string. $EU\ fsinecoc\hat{a}precis$

```
from string import Template

tempStr = Template('The symbol for Dollar is $$')
print (tempStr.substitute())
```

Ele produzirá a seguinte **saída** -

```
The symbol for Dollar is $
```

Exemplo 7

Se você deseja usar qualquer outro caractere em vez de "$" como símbolo de substituição, declare uma subclasse da classe Template e atribua -

```
from string import Template

class myTemplate(Template):
   delimiter = '#'

tempStr = myTemplate('Hello. My name is #name and my age is #age')
print (tempStr.substitute(name='Harsha', age=30))
```