Python - Literais

O que são literais Python?

Literais ou constantes Python são a notação para representar um valor fixo no código-fonte. Em contraste com variáveis , literais (123, 4.3, "Hello") são valores estáticos ou você pode dizer constantes que não mudam durante a operação do programa ou aplicativo. Por exemplo, na seguinte instrução de atribuição.

```
x = 10
```

Aqui, 10 é um valor literal que representa 10, que é armazenado diretamente na memória. No entanto,

```
y = x*2
```

Aqui, mesmo que a expressão seja avaliada como 20, ela não está literalmente incluída no código-fonte. Você também pode declarar um objeto int com a função int() integrada. Porém, esta também é uma forma indireta de instanciação e não literal.

```
x = int(10)
```

Tipos de literais Python

Python fornece os seguintes literais que serão explicados neste tutorial:

- Python - Literal Inteiro
- Python - literal flutuante
- Python - Literal Complexo
- Python - String Literal
- Python - Lista Literal
- Python - Tupla Literal
- Python - Dicionário Literal

Python - Literal Inteiro

Qualquer representação envolvendo apenas os símbolos de dígitos (0 a 9) cria um objeto do tipo **int** . O objeto assim declarado pode ser referido por uma variável usando um operador de atribuição.

Exemplo: Literal Decimal

Dê uma olhada no **exemplo** a seguir -

```
x = 10
y = -25
z = 0
```

Exemplo: Literal Octal

Python permite que um número inteiro seja representado como um número octal ou hexadecimal. Uma representação numérica com apenas símbolos de oito dígitos (0 a 7), mas prefixados por 0o ou 0O, é um número octal em Python.

```
x = 0O34
```

Exemplo: Literal Hexadecimal

Da mesma forma, uma série de símbolos hexadecimais (0 a 9 e a a f), prefixados por 0x ou 0X representa um número inteiro na forma hexedecimal em Python.

```
x = 0X1C
```

Exemplo: demonstração de notações octais e hexadecimais como números inteiros

No entanto, pode-se notar que, mesmo se você usar notação literal octal ou hexadecimal, o Python os trata internamente como do tipo **int** .

```
# Using Octal notation
x = 0O34
print ("0O34 in octal is", x, type(x))
# Using Hexadecimal notation
x = 0X1c
print ("0X1c in Hexadecimal is", x, type(x))
```

Quando você executa este código, ele produzirá a seguinte **saída** -

```
0O34 in octal is 28 <class 'int'>
0X1c in Hexadecimal is 28 <class 'int'>
```

Python - literal flutuante

Um número de ponto flutuante consiste em uma parte inteira e uma parte fracionária. Convencionalmente, um símbolo de ponto decimal (.) separa essas duas partes em uma representação literal de um ponto flutuante. Por exemplo,

Exemplo: literal flutuante

```
x = 25.55
y = 0.05
z = -12.2345
```

Para um número de ponto flutuante muito grande ou muito pequeno, onde o número de dígitos antes ou depois do ponto decimal é maior, uma notação científica é usada para uma representação literal compacta. O símbolo E ou e seguido de número inteiro positivo ou negativo segue após a parte inteira.

Exemplo: Literal de notação científica flutuante

Por exemplo, um número 1.23E05 é equivalente a 123.000,00. Da mesma forma, 1,23e-2 é equivalente a 0,0123

```
# Using normal floating point notation
x = 1.23
print ("1.23 in normal float literal is", x, type(x))
# Using Scientific notation
x = 1.23E5
print ("1.23E5 in scientific notation is", x, type(x))
x = 1.23E-2
print ("1.23E-2 in scientific notation is", x, type(x))
```

Aqui, você obterá a seguinte **saída** -

```
1.23 in normal float literal is 1.23 <class 'float'>
1.23E5 in scientific notation is 123000.0 <class 'float''>
```

```
1.23E-2 in scientific notation is 0.0123 <class 'float'>
```

Python - Literal Complexo

Um número complexo é composto por um componente real e imaginário. O componente imaginário é qualquer número (inteiro ou ponto flutuante) multiplicado pela raiz quadrada de "-1"

($\sqrt{-1}$). Em representação literal ($\sqrt{-1}$) é representação por "j" ou "J". Portanto, uma representação literal de um número complexo assume a forma x+yj.

Exemplo: Literal de tipo complexo

```
#Using literal notation of complex number
x = 2+3j
print ("2+3j complex literal is", x, type(x))
y = 2.5+4.6j
print ("2.5+4.6j complex literal is", x, type(x))
```

Este código produzirá a seguinte **saída** -

```
2+3j complex literal is (2+3j) <class 'complex'>
2.5+4.6j complex literal is (2+3j) <class 'complex'>
```

Python - String Literal

Um objeto string é um dos tipos de dados de sequência em Python . É uma sequência imutável de pontos de código Unicode . O ponto de código é um número correspondente a um caractere de acordo com o padrão Unicode. Strings são objetos da classe interna 'str' do Python.

Literais de string são escritos colocando uma sequência de caracteres entre aspas simples ('hello'), aspas duplas ("hello") ou aspas triplas ('''hello''' ou """hello""").

Exemplo: String Literal

```
var1='hello'
print ("'hello' in single quotes is:", var1, type(var1))
var2="hello"
print ('"hello" in double quotes is:', var1, type(var1))
var3='''hello'''
```

```
print ("''''hello'''' in triple quotes is:", var1, type(var1))
var4="""hello"""
print ('"""hello""" in triple quotes is:', var1, type(var1))
```

Aqui, você obterá a seguinte **saída** -

```
'hello' in single quotes is: hello <class 'str'>
"hello" in double quotes is: hello <class 'str'>
''''hello'''' in triple quotes is: hello <class 'str'>
"""hello""" in triple quotes is: hello <class 'str'>
```

Exemplo: String Literal com String Interna entre Aspas Duplas

Se for necessário incorporar aspas duplas como parte da string, a própria string deverá ser colocada entre aspas simples. Por outro lado, se o texto entre aspas simples for incorporado, a string deverá ser escrita entre aspas duplas.

```
var1='Welcome to "Python Tutorial" from TutorialsPoint'
print (var1)
var2="Welcome to 'Python Tutorial' from TutorialsPoint"
print (var2)
```

Ele produzirá a seguinte **saída** -

```
Welcome to "Python Tutorial" from TutorialsPoint
Welcome to 'Python Tutorial' from TutorialsPoint
```

Python - Lista Literal

O objeto de lista em Python é uma coleção de objetos de outro tipo de dados. Lista é uma coleção ordenada de itens não necessariamente do mesmo tipo. O objeto individual da coleção é acessado por índice começando com zero.

A representação literal de um objeto de lista é feita com um ou mais itens separados por vírgula e entre colchetes [].

Exemplo: Tipo de Lista Literal

```
L1=[1,"Ravi",75.50, True]
print (L1, type(L1))
```

Ele produzirá a seguinte **saída** -

```
[1, 'Ravi', 75.5, True] <class 'list'>
```

Python - Tupla Literal

Objeto tupla em Python é uma coleção de objetos de outro tipo de dados. Tupla é uma coleção ordenada de itens não necessariamente do mesmo tipo. O objeto individual da coleção é acessado por índice começando com zero.

A representação literal de um objeto tupla é feita com um ou mais itens separados por vírgula e colocados entre parênteses ().

Exemplo: Literal de tipo tupla

```
T1=(1,"Ravi",75.50, True)
print (T1, type(T1))
```

Ele produzirá a seguinte **saída** -

```
[1, 'Ravi', 75.5, True] <class tuple>
```

Exemplo: Literal tipo tupla sem parênteses

O delimitador padrão para sequência Python são parênteses, o que significa que uma sequência separada por vírgula sem parênteses também equivale à declaração de uma tupla.

```
T1=1,"Ravi",75.50, True
print (T1, type(T1))
```

Aqui também, você obterá o mesmo **resultado** -

```
[1, 'Ravi', 75.5, True] <class tuple>
```

Python - Dicionário Literal

Assim como a lista ou a tupla, o dicionário também é um tipo de dados de coleção. No entanto, não é uma sequência. É uma coleção não ordenada de itens, cada um dos quais é um par de valores-chave. O valor está vinculado à chave pelo símbolo

":". Um ou mais pares chave:valor separados por vírgula são colocados entre chaves para formar um objeto de dicionário.

Exemplo: Tipo de Dicionário Literal

```
capitals={"USA":"New York", "France":"Paris", "Japan":"Tokyo",
"India":"New Delhi"}
numbers={1:"one", 2:"Two", 3:"three",4:"four"}
points={"p1":(10,10), "p2":(20,20)}

print (capitals, type(capitals))
print (numbers, type(numbers))
print (points, type(points))
```

A chave deve ser um objeto imutável. Número, string ou tupla podem ser usados como chave. A chave não pode aparecer mais de uma vez em uma coleção. Se uma chave aparecer mais de uma vez, apenas a última será mantida. Os valores podem ser de qualquer tipo de dados. Um valor pode ser atribuído a mais de uma chave. Por exemplo,

```
staff={"Krishna":"Officer", "Rajesh":"Manager", "Ragini":"officer", "Anil":"C
```