Python

Topics: Expressions

Expressions

Tutorial

Python Expressions:

Expressions are representations of value. They are different from statement in the fact that statements do something while expressions are representation of value. For example any string is also an expressions since it represents the value of the string as well.

Python has some advanced constructs through which you can represent values and hence these constructs are also called expressions.

In this tutorial you will get to know about:

1. What are expressions in Python
2. How to construct expressions.

How to create an expressions

Python expressions only contain identifiers, literals, and operators. So, what are these?

Identifiers: Any name that is used to define a class, function, variable module, or object is an identifier. **Literals**: These are language-independent terms in Python and should exist independently in any programming language. In Python, there are the string literals, byte literals, integer literals, floating point literals, and imaginary literals. **Operators**: In Python you can implement the following operations using the corresponding tokens.

Operator	Token
add	+

Operator	Token
subtract	-
multiply	*
power	**
Integer Division	/
remainder	%
decorator	@
Binary left shift	<<
Binary right shift	>>
and	&
or	\
Binary Xor	^
Binary ones complement	~
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=
Check equality	==
Check not equal	!=

Following are a few types of python expressions:

List comprehension

The syntax for list comprehension is shown below:

```
[ compute(var) for var in iterable ]
```

For example, the following code will get all the number within 10 and put them in a list.

```
>>> [x for x in range(10)]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

?

Dictionary comprehension

This is the same as list comprehension but will use curly braces:

```
{ k, v for k in iterable }
```

For example, the following code will get all the numbers within 5 as the keys and will keep the corresponding squares of those numbers as the values.

```
>>> {x:x**2 for x in range(5)}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

Generator expression

The syntax for generator expression is shown below:

```
( compute(var) for var in iterable )
```

For example, the following code will initialize a generator object that returns the values within 10 when the object is called.

```
>>> (x for x in range(10))
<generator object <genexpr> at 0x7fec47aee870>
>>> list(x for x in range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Conditional Expressions

You can use the following construct for one-liner conditions:

```
true_value if Condition else false_value
```

Example:

```
>>> x = "1" if True else "2"
>>> x
'1'
```

Contributed by: Joydeep Bhattacharjee

?

Did you find this tutorial helpful? Yes No

View all comments

+1-650-461-4192

contact@hackerearth.com

?