Python eval()

The eval() method parses the expression passed to this method and runs python expression (code) within the program.

In simple terms, the eval() method runs the python code (which is passed as an argument) within the program.

The syntax of eval() is:

```
eval(expression, globals=None, locals=None)
```

eval() Parameters

The eval() takes three parameters:

- **expression** - this string as parsed and evaluated as a Python expression
- **globals** (optional) - a dictionary
- **locals** (optional)- a mapping object. Dictionary is the standard and commonly used mapping type in Python.

The use of `globals` and `locals` will be discussed later in this article.

Return Value from eval()

The eval() method returns the result evaluated from the `expression`.

Example 1: How eval() works in Python?

script.py IPython Shell

```
x = 1
print(eval('x + 1'))
```

Run

Powered by DataCamp

When you run the program, the output will be:

```
2
```

Here, the eval() evaluates the expression `x + 1` and print it.

Example 2: Practical Example to Demonstrate Use of eval()

```
# Perimeter of Square
def calculatePerimeter(l):
   return 4*l

# Area of Square
def calculateArea(l):
   return l*1
```

```
property = input("Type a function: ")

for l in range(1, 5):
     if (property == 'calculatePerimeter(l)'):
          print("If length is ", l , ", Perimeter = ", eval(property))
     elif (property == 'calculateArea(l)'):
       print("If length is ", l , ", Area = ", eval(property))
     else:
        print('Wrong Function')
        break
```

The output of the above program will be:

```
Type a function:  calculateArea(l)
If length is  1 , Area =  1
If length is  2 , Area =  2
If length is  3 , Area =  3
If length is  4 , Area =  4
```

Why you should be careful while using eval()?

Consider a situation, you are using a Unix system (macOS, Linux etc) and you have imported `os` module. The os module provides portable way to use operating system functionalities like: read or write a file.

If you allow users to input a value using `eval(input())`, the user may issue commands to change file or even delete all the files using command `os.system('rm -rf *')`.

If you are using `eval(input())` in your code, it's a good idea to check which variables and methods the user can use. You can see which variables and methods are available using dir() method.

script.py IPython Shell

Run

Powered by DataCamp

When you run the program, the output would be something like:

```
['__annotations__', '__builtins__', '__doc__', '__file__', '__loader__',
```

Restricting the Use of Available Methods and Variables in eval()

More often than not, all the available methods and variables used in the `expression` (first parameter to the eval()) may not be needed, or even may have a security hole. You may need to restrict the use of these methods and variables for eval(). You can do so by passing optional `globals` and `locals` parameters (dictionaries) to the eval() method.

1. When both globals and locals parameters omitted

If both parameters are omitted (as in our earlier examples), the `expression` is executed in the current scope. You can check the available variables and methods using following code:

```
print(eval('dir()'))
```

2. Passing globals parameter; locals parameter is omitted

The globals and locals parameters (dictionaries) are used for global and local variables respectively. If the locals dictionary is omitted, it defaults to globals dictionary. Meaning, globals will be used for both global and local variables.

Note: You can check the current global and local dictionary in Python using globals() and locals() built-in methods respectively.

Passing empty dictionary as globals parameter

script.py IPython Shell

```
1  from math import *
2  print(eval('dir()', {}))
3
4  # The code below will raise an exception
5  # print(eval('sqrt(25)', {}))
```

Run

Powered by DataCamp

If you pass an empty dictionary as globals , only the __builtins__ are available to expression (first parameter to the eval()). Even though we have imported math module in the above program, expression can't access none of the functions provided by the math module.

When you run the program, the output will be:

```
['__builtins__']
```

Making Certain Methods available

script.py IPython Shell

```
1  from math import *
2  print(eval('dir()', {'sqrt': sqrt, 'pow': pow}))
```

Run

Powered by DataCamp

Here, the expression can also use sqrt() and pow() methods along with __builtins__ .

Also, it's possible to change the name of the method available for the expression according to your wish.

script.py IPython Shell

```
1  from math import *
2  print(eval('dir()', {'squareRoot': sqrt, 'pow': pow}))
3
4  # Using squareRoot in Expression
5  print(eval('squareRoot(9)', {'squareRoot': sqrt, 'pow': pow}))
```

Run

Powered by DataCamp

In the above program, squareRoot() calculates the square root (similar functionality like sqrt()). However, trying to use sqrt() will raise an error.

Restricting the Use of built-ins

You can restrict the use of __builtins__ in the expression as follows:

```
eval(expression, {'__builtins__': None})
```

3. Passing both globals and locals dictionary

You can make needed functions and variables available for use by passing locals dictionary. For example:

script.py IPython Shell

```
1  from math import *
2
3  a = 5
4  print(eval('sqrt(a)', {'__builtins__': None}, {'a': a, 'sqrt': sqrt}))
```

Run

Powered by DataCamp

When you run the program, the output will be:

```
2.23606797749979
```

In this program, the `expression` (first parameter to the eval) can have `sqrt()` method and variable `a` only. All other methods and variables are unavailable.

Restricting the use of eval() by passing `globals` and `locals` dictionary will make your code secure particularly when you are using input provided by the user to the eval() method.

Built-in Methods

Python abs()

Python any()

Python all()

Python ascii()

Python bin()

Python bool()

Python bytearray()

Python callable()

Python bytes()

Python chr()

Python compile()

Receive the latest tutorial to improve your programming skills.

Enter Email Address*

Join

TUTORIALS

Python Tutorials

C Tutorials

Java Tutorials

Kotlin Tutorials

C++ Tutorials

Swift Tutorials

R Tutorials

DSA

EXAMPLES

Python Examples

C Examples

Java Examples

Kotlin Examples

C++ Examples

R Examples

COMPANY

About

Advertising

Contact

LEGAL

Privacy Policy

Terms And Conditions

App's Privacy Policy

App's Terms And Conditions