Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

miércoles, 11 de marzo de 2015

Evaluar, ejecutar y compilar cadenas

Las funciones **eval()**, **exec()** y **compile()** son del grupo de funciones integradas o ***built-in functions*** de Python.

eval()

La función **eval()** se utiliza para evaluar cadenas de texto que pueden contener expresiones o distintos tipos de estructuras de datos que pueden utilizarse con Python, tales como listas, tuplas, diccionarios y otros objetos que admiten asignación.

Si la evaluación se realiza con éxito la función **eval()** retornará un objeto con el tipo que mejor represente al contenido analizado.

En el siguiente ejemplo se utiliza una lista que contiene varias cadenas que son evaluadas con la función **eval()**:

```
precio = 5
cadenas = ['(4+5)**2',
           '(1, 2, 3)',
           '["I", "II", "III"]',
           '{"a":1, "b":2, "c":3}',
           'len("Python")',
           '20 * precio',
           '__import__("platform").python_version()']

for cadena in cadenas:
    print(cadena, "=>", eval(cadena),
          "Tipo:",type(eval(cadena)))
```

Si ejecutamos el código anterior obtendríamos el siguiente resultado:

```
(4+5)**2 => 81 Tipo: <class 'int'>
(1, 2, 3) => (1, 2, 3) Tipo: <class 'tuple'>
["I", "II", "III"] => ['I', 'II', 'III'] Tipo: <class 'list'>
{"a":1, "b":2, "c":3} => {'c': 3, 'a': 1, 'b': 2} Tipo: <class 'dict'>
len("Python") => 6 Tipo: <class 'int'>
20 * precio => 100 Tipo: <class 'int'>
__import__("platform").python_version() => 3.4.0 Tipo: <class 'str'>
```

Si por el contrario la función no consigue evaluar la expresión se producirá una excepción que habrá que tratar convenientemente.

En el siguiente ejemplo la cadena de la variable ***entrada*** contiene una lista a la que le falta el corchete de cierre y este hecho provoca un error de sintaxis:

```
try:
    entrada = '["castaño","encina","roble"'
    print(entrada, "=>", eval(entrada))
```

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones que permiten obtener de distintos modos números a...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

```
except SyntaxError:
    print('Se ha producido un error de sintaxis')
```

exec()

La función **exec()** permite ejecutar código Python contenido en una cadena o en un archivo. Si el código no cumple con las reglas del lenguaje producirá una excepción.

En el siguiente ejemplo se ejecutan varias cadenas que contienen instrucciones de Python.

```
exec('secreto = input("Introducir clave secreta: ")')
exec('if secreto == "1234": print("¡Eureka!")')
exec('print("Clave secreta:", secreto)')
```

Por cierto, el término **exec** en Python 2.x forma parte del conjunto de palabras reservadas del lenguaje. Sin embargo, en Python 3.x **exec()** es una función y ya no forma parte de tan distinguido grupo.

```
>>> from keyword import iskeyword, kwlist
>>> kwlist   # List parabras reservadas
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
'return', 'try', 'while', 'with', 'yield']

>>> iskeyword("exec")   # Retorna True si el término es palabra reservada
False
```

compile()

La función **compile()**, que no se utiliza con frecuencia, se emplea para compilar el código fuente contenido en una cadena en código objeto. Después, según el caso, el código objeto puede evaluarse con la función **eval()** o ejecutarse con la función **exec()**.

Una de las aplicaciones posibles de **compile()** permite que los usuarios puedan introducir código en forma de cadenas en un programa Python que se está ejecutando. Después, esas cadenas pueden compilarla el mismo programa antes de ser ejecutadas o guardadas en un archivo.

Una de las razones por la que la función **compile()** se utiliza raramente es porque un uso indebido de esta función puede derivar en problemas de seguridad.

- compile() y eval():

```
>>> codobjeto = compile('[1, 2, 3, 4, 5]', 'lista', 'eval')
>>> eval(codobjeto)
[1, 2, 3, 4, 5]

>>> print(codobjeto)
<code object <module> at 0xb5d8b340, file "lista", line 1>

>>> type(codobjeto)
code
```

- compile() y exec():

```
>>> codobjeto = compile('for t in range(11):print(t, t**2)', 'modcuadrado', 'exec')
>>> exec(codobjeto)
0 0
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100

>>> print(codobjeto)
<code object <module> at 0xb6865a20, file "modcuadrado", line 1>

>>> type(codobjeto)
code
```

simultáneamente varias operaciones en el mismo espacio de proceso se...

Archivo

marzo 2015 (2)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

Ir al índice del tutorial de Python

Publicado por Pherkad en 14:49

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua