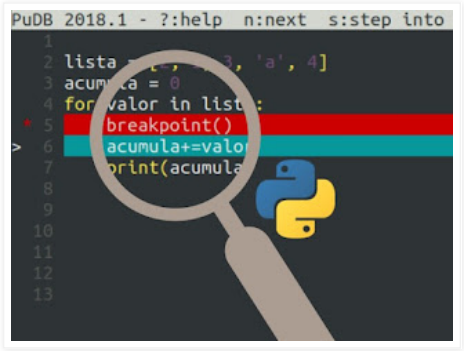
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

martes, 9 de julio de 2019

Facilitando la depuración de programas con breakpoint()



Python 3.7 incluye una mejora que facilita a los desarrolladores el acceso al depurador de programas de Python (**Pdb**). Antes para poder invocar al depurador era necesario incluir en un programa la siguiente línea:

```
import pdb; pdb.set_trace()
```

Esto ya no es necesario. Ahora para depurar el código simplemente tendremos que insertar la función **breakpoint()** en el lugar o lugares donde se quieran establecer los puntos de interrupción. Con este cambio se persigue facilitar la tarea, hacerla más cómoda e intuitiva.

Después, una vez iniciado el proceso de depuración todo funciona como hasta ahora: cuando se alcanza un punto de interrupción se hace una parada temporal y se devuelve el control al usuario para ejecutar el programa paso a paso (***n***) o hasta el siguiente punto de interrupción (***c***), consultar el valor de las variables, etc. Veamos un caso práctico.

El siguiente ejemplo contiene una lista de valores a sumar. Para ello, se recorren, uno a uno, los distintos elementos y se van sumando a la variable **acumula**, que inicialmente tiene el valor **0**. El programa **sumando.py** genera una excepción cuando se alcanza el cuarto elemento (**'a'**) por ser de tipo cadena; y por no poder sumarse a un valor numérico. En este caso para la depuración se incluye el punto de interrupción justo antes de la suma del elemento en curso para permitir evaluar su valor.

sumando.py:

```
lista = [2, 1, 3, 'a', 4]
acumula = 0
for valor in lista:
    breakpoint()
    acumula+=valor
    print(acumula)
```

Para iniciar la depuración, ejecutar:

```
$ python3 sumando.py
```

Una vez iniciada, el programa avanzará hasta el punto de interrupción. Después, si escribimos el nombre de alguna de las variables (**valor** o **acumula**) y presionamos **[return]** obtendremos su valor actual. A continuación, al pulsar **'c'** y **[return]** el programa mostrará el resultado de la primera suma y completará un ciclo hasta alcanzar de nuevo el punto de interrupción. Si repetimos la acción y vamos consultando la variable **valor** observaremos que el error se produce cuando se alcanza el cuarto elemento de la lista:

```
TypeError: unsupported operand type(s) for +=: 'int' and 'str'
```

Para obtener ayuda de otras opciones del depurador escribir **'h'** y **[return]**. Para ampliar la

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...

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

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...

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...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

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

información de ayuda de alguna opción escribir: **'h' [opción]** y **[return]**. Y para salir del depurador escribir **'exit'**, **'quit'** o **'q'** y **[return]**.

Para ejecutar el programa omitiendo la depuración:

```
$ PYTHONBREAKPOINT=0 python3 sumando.py
```

Y para ejecutar el programa con el depurador de consola PuDB:

```
$ PYTHONBREAKPOINT=pudb.set_trace python3 sumando.py
```

Relacionado:

- Solucionando errores con el depurador
- El depurador PuDB
- Editar y depurar scripts (IPython)

Ir al índice del tutorial de Python

Publicado por Pherkad en 12:32

Etiquetas: depurador, Python3

Entrada más reciente Inicio Entrada antigua

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

Archivo

julio 2019 (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