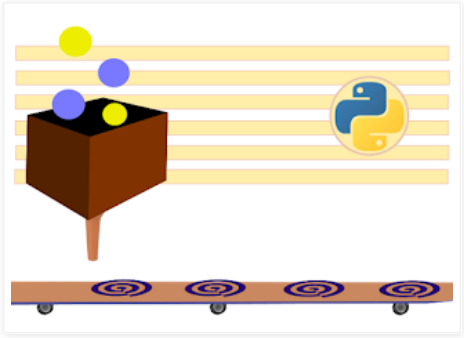
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

domingo, 23 de febrero de 2020

Funciones que imponen nombrar o no parámetros



Cuando se llama a una función Python con parámetros se pueden pasar los valores atendiendo a la posición de los parámetros o referenciando el nombre de los mismos.

A partir de Python 3.8 al declarar una función además puede establecerse qué parámetros, cuando se llame a la función, deben indicarse con su nombre y cuáles no.

Una opción consiste en introducir una barra inclinada hacia la derecha '/' en la definición de la función, entre los parámetros, para obligar a que todos aquellos que se encuentren a su izquierda no puedan referenciar su nombre cuando la función es llamada:

```
# En este primer ejemplo se muestran los modos habituales
# de llamar a una función (por referencia posicional o por
# nombre):

def funcion1(x, y, z):
    return x + y + z

print(funcion1(1, 2, 3))  # 6
print(funcion1(x=1, y=2, z=3))  # 6
print(funcion1(z=3, x=1, y=2))  # 6


# En los siguientes casos todos los parámetros a la
# izquierda de la barra '/' deben indicarse sin nombrar.
# Si no es así se producirá una excepción de
# tipo TypeError:

def funcion2(x, y, z, /):
    return x + y + z

print(funcion2(1, 2, 3))  # 6 (es correcta la llamada)
print(funcion2(x=1, y=2, z=3))  # Genera el error siguiente:
# TypeError: funcion2() got some positional-only arguments
# passed as keyword arguments: 'x, y, z'


def funcion3(x, /, y, z):
    return x + y + z

print(funcion3(x=1, y=2, z=3))  # Genera el error siguiente:
# TypeError: funcion3() got some positional-only arguments
# passed as keyword arguments: 'x'
# El parámetro 'x' por estar a la izquierda de la barra '/'
# necesariamente debe indicarse sin nombrar:
print(funcion3(1, z=2, y=3))  # 6
```

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...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones

Otra posibilidad es incluir un asterisco '*' entre los parámetros para imponer que todos los que se encuentren a su derecha tengan que referenciar su nombre:

```
def funcion4(x, y, *, z):
    return x + y + z

print(funcion4(x=1, y=2, z=3))  # 6 (es correcta esta llamada)
print(funcion4(1, 2, 3))  # Genera la siguiente excepción:
# TypeError: funcion4() takes 2 positional arguments
# but 3 were given
# El parámetro 'z' por estar a la derecha del '*' debe
# ser referenciado con su nombre:
print(funcion4(1, 2, z=3))  # 6
```

Esta nueva funcionalidad puede resultar de utilidad cuando entre distintas versiones de una API existan funciones que han variado el número de sus parámetros y, por razones de compatibilidad, se prefiera imponer a los desarrolladores el nombrar o no a conveniencia determinados parámetros.

Relacionado:

- Funciones
- Programación Orientada a Objetos
- Programación funcional: funciones de orden superior
- Diccionario de variables locales y globales

Publicado por Pherkad en 4:41

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua

que permiten obtener de distintos modos números a...

Archivo

febrero 2020 (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