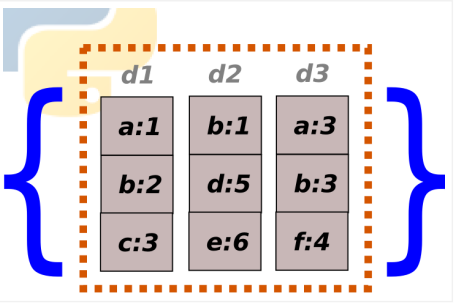
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

martes, 21 de abril de 2015

Combinar diccionarios con ChainMap



La clase **ChainMap** del módulo **collections** permite crear objetos como resultado de combinar varios diccionarios. En el objeto que se obtiene se hace un mapeo lógico de todas las claves existentes en todos los diccionarios, permitiéndose realizar ciertas operaciones, como la búsqueda de valores o comprobar la existencia de dichas claves. **ChainMap** está disponible desde Python 3.3.

```
import collections
diccionario1 = {'cafés':12, 'zumos':9}
diccionario2 = {'zumos':6, 'infusiones':8}

# Combinar dos diccionarios y obtener un objeto ChainMap:

diccionario3 = collections.ChainMap(diccionario1,
                                    diccionario2)

# Mostrar el objeto ChainMap:

print(diccionario3)
# ChainMap({'cafés': 12, 'zumos': 9},
#          {'infusiones': 8, 'zumos': 6})

# Accediendo al valor de varias claves:

print(diccionario3['cafés'])  # 12, del diccionario1
print(diccionario3['zumos'])  # 9, del diccionario1
print(diccionario3['infusiones'])  # 8, del diccionario2
```

Como puede observarse si se accede al valor de una clave que existe en varios diccionarios, se obtiene el valor que tenga en el primer diccionario donde se encuentre dicha clave.

De lo anterior se deduce que cuando se combinan varios diccionarios realmente no se produce una fusión de todas las asignaciones existentes, en el sentido más estricto del término. En realidad un objeto **ChainMap** mantiene todas las asignaciones vinculadas con su diccionario de origen y redefine las operaciones que se aplican a un diccionario común para poder explorar los elementos combinados. Además, aquellas operaciones que se aplican a un diccionario común funcionarán con un objeto **ChainMap**.

```
# Añadir nuevos pares a un objeto ChainMap:

diccionario3['cacaos'] = 23
diccionario3['granizados'] = 4

# Mostrar el objeto ChainMap:

print(diccionario3)
# ChainMap({'cafés': 12, 'granizados': 4,'zumos': 9,
#           'cacaos': 23}, {'infusiones': 8, 'zumos': 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

```
# Mostrar el número de pares en el objeto ChainMap:

print(len(diccionario3))  # Objeto ChainMap tiene 5 pares

# Mostrar el número de pares del diccionario diccionario1:

print(len(diccionario1))  # diccionario1 tiene 4 pares

# Borrar un par clave/valor si existe en
# el objeto ChainMap:

if "granizados" in diccionario3:
    del diccionario3['granizados'] # Si existe es borrado

if "granizados" not in diccionario3:
    print('No existe')  # Si no existe muestra mensaje

# Mostrar el objeto ChainMap:

print(diccionario3)
# ChainMap({'cafés': 12, 'zumos': 9, 'cacaos': 23},
#          {'infusiones': 8, 'zumos': 6})

# Obtener una lista actualizada con todos los
# diccionarios del ChainMap:

lista_diccionarios = diccionario3.maps
print(lista_diccionarios)
```

Por otro lado, un objeto **ChainMap** es particularmente útil cuando se ha construido a partir de varios diccionarios hijos que comparten pares con claves repetidas porque éstas pueden ser recorridas por descartes sucesivos:

```
import collections

# Crear un objeto ChainMap vacío:

pizzas = collections.ChainMap()
pizzas['tonno'] = 5

# Añadir un nuevo diccionario hijo:

pizzas = pizzas.new_child()
pizzas['tonno'] = 10

# Agregar un nuevo diccionario hijo:

pizzas = pizzas.new_child()
pizzas['tonno'] = 15

# Mostrar objeto ChainMap:

print(pizzas)
# ChainMap({'tonno': 15},
#          {'tonno': 10},
#          {'tonno': 5})

# Acceder al valor de una clave:

print(pizzas['tonno']) # 15

# Descartar el primer mapeo del objeto ChainMap
# y mostrar valor de clave

pizzas = pizzas.parents
print(pizzas['tonno']) # 10

# Descarta el siguiente mapeo del objeto ChainMap
# y mostrar valor de clave

pizzas = pizzas.parents
print(pizzas['tonno']) # 5

# Mostrar objeto ChainMap

print(pizzas)  # ChainMap({'tonno': 5})
```

Ir al índice del tutorial de Python

que permiten obtener de distintos modos números a...

Archivo

abril 2015 (6)

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

Publicado por Pherkad en 12:05

Etiquetas: collections, Python3

Entrada más reciente Inicio Entrada antigua