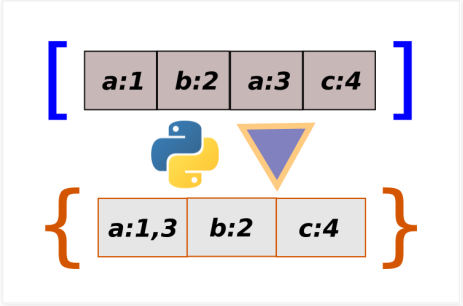
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

jueves, 23 de abril de 2015

defaultdict vs setdefault



El método **setdefault()** se utiliza con un diccionario común para recuperar el valor de una clave y, si ésta no existe, establecer un valor predeterminado:

```
# Establecer el valor por defecto de 'y' con 5:

diccionario = {'x':2, 'y':5}
print(diccionario.setdefault('y', 1))  # 5

# Mostrar los datos del diccionario:
# 'y' existe con valor 5 y ese es el valor mostrado

print(diccionario)  # {'x': 2, 'y': 5}

# Establecer el valor por defecto de 'z' en 3:

print(diccionario.setdefault('z', 3))

# Mostrar el valor de 'z':
# como no existía aparece su valor por defecto:

print(diccionario['z'])  # 3

# Mostrar los datos del diccionario:

print(diccionario)  # {'z': 3, 'y': 5, 'x': 2}
```

Sin embargo, **defaultdict**, del módulo **collections**, permite asignar un valor predeterminado por adelantado, justo en el momento de inicializar el contenedor:

```
import collections

def fvalpredeter():
    return 1

ddiccionario = collections.defaultdict(fvalpredeter, x=2, z=3)
print('ddiccionario:', ddiccionario)
print('x: ', ddiccionario['x']) # 2
print('y: ', ddiccionario['y']) # 1
print('z: ', ddiccionario['z']) # 3
print('w: ', ddiccionario['w']) # 1
print(dict(ddiccionario))  # {'z': 3, 'x': 2, 'w': 1, 'y': 1}

# La función fvalpredeter se podía haber expresado con
# una función lambda:

ddiccionario = collections.defaultdict(lambda: 1, x=2, z=3)
```

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

(Ver información sobre las funciones lambda).

La aplicación del ejemplo anterior es útil cuando todas las claves deben tener el mismo valor por defecto.

No obstante, el uso de **defaultdict** puede resultar más interesante cuando el valor por defecto se establece como del tipo ***list***, ***set*** o ***int***. La razón está en que estos tipos permiten añadir, contar y acumular datos:

```
# En el ejenplo siguiente defaultdict se utiliza con list
# para agrupar valores de claves que se repiten:

import collections
peces1 = [('dorada', 1), ('merluza', 2),
          ('dorada', 3), ('bacalao', 4),
          ('merluza', 1)]
ddiccionario1 = collections.defaultdict(list)
print(list(ddiccionario1))
for clave, valor in peces1:
    ddiccionario1[clave].append(valor)

list(ddiccionario1.items())
# [('dorada', [1, 3]), ('merluza', [2, 1]), ('bacalao', [4])]

# En el ejemplo siguiente defaultdict se utiliza con int
# para contar elementos repetidos:

peces2 = ['dorada', 'merluza', 'dorada',
          'merluza', 'merluza', 'bacalao']
cont_peces2 = collections.defaultdict(int)
for pez in peces2:
    cont_peces2[pez] += 1

print(dict(cont_peces2))
# {'merluza': 3, 'bacalao': 1, 'dorada': 2}

# En el siguiente ejemplo a partir de una cadena se
# obtiene una lista de tuplas ordenadas alfabéticamente
# con el número de veces que se repite cada carácter:

poppins = 'supercalifragilisticoexpialidoso'
ddiccionario = collections.defaultdict(int)
for clave in poppins:
    ddiccionario[clave] += 1

print(sorted(list(ddiccionario.items())))
# [('a', 3), ('c', 2), ('d', 1), ('e', 2), ('f', 1),
#  ('g', 1), ('i', 6), ('l', 3), ('o', 3), ('p', 2),
#  ('r', 2), ('s', 3), ('t', 1), ('u', 1), ('x', 1)]

# A continuación, defaultdict se utiliza con set
# para obtener una lista de elementos con los valores
# no repetidos:

peces3 = [('dorada', 1), ('merluza', 2),
          ('merluza', 2), ('dorada', 3)]
dict_peces3 = collections.defaultdict(set)

for clave, valor in peces3:
    dict_peces3[clave].add(valor)

print(list(dict_peces3.items()))
# [('dorada', {1, 3}), ('merluza', {2})]

# En el ejemplo siguiente defaultdict se utiliza con set
# para agrupar las posiciones que ocupan los elementos
# en una lista (el primer elemento se encuentra en
# la posición 0):

peces4 = ['dorada', 'merluza', 'dorada',
          'merluza', 'merluza', 'bacalao']
cont_peces4 = collections.defaultdict(set)
for indice, clave in enumerate(peces4):
    cont_peces4[clave].add(indice)

print(dict(cont_peces4))
# {'dorada': {0, 2}, 'bacalao': {5}, 'merluza': {1, 3, 4}}

cont_peces4['dorada']  # {0, 2}
```

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

```
dorada = list(cont_peces4['dorada'])
print(dorada)  # [0, 2]
```

ir al índice del tutorial de Python

Publicado por Pherkad en 16:17

Etiquetas: collections, Python3

Entrada más reciente Inicio Entrada antigua