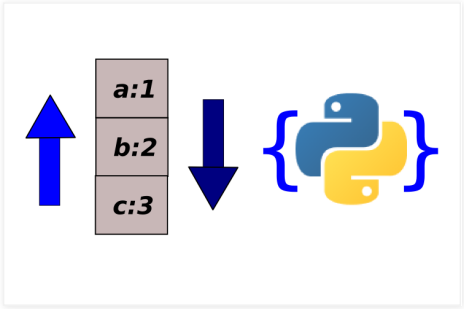
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

miércoles, 15 de abril de 2015

Con OrderedDict el orden ha llegado



Un diccionario en Python es un objeto de la clase **dict** bastante peculiar porque sus pares de claves/valor cuando son recorridos no tienen porque aparecer en el mismo orden en que fueron agregados. Es más, si recorremos un diccionario varias veces y mostramos los pares existentes es muy probable que aparezcan cada vez en distinto orden.

Después de ejecutar tres veces el siguiente ejemplo:

```
dicc = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5, 'f':6, 'g':7}

for clave in dicc.keys():
    print(clave, "->" , dicc[clave], end="|")
```

El resultado obtenido ha sido:

1) a -> 1|f -> 6|g -> 7|e -> 5|c -> 3|b -> 2|d -> 4|
2) f -> 6|d -> 4|a -> 1|b -> 2|g -> 7|c -> 3|e -> 5|
3) f -> 6|b -> 2|c -> 3|g -> 7|a -> 1|d -> 4|e -> 5|

El concepto de orden en un diccionario está fuera de lugar. Sin embargo, el objeto **OrderedDict** del módulo **collections** pertenece a una subclase de la clase **dict** que sí permite mantener las claves de un diccionario en el mismo orden en que fueron insertadas. Además, si una nueva entrada sobrescribe una existente la posición de inserción original se mantiene; pero si previamente es eliminada y después se vuelve a agregar pasará a ocupar la última posición.

```
import collections

escritores = {'Borges':'Argentina',
              'Coehlo':'Brasil',
              'Lorca':'España',
              'Márquez':'Colombia',
              'Neruda':'Chile',
              'Rulfo':'México',
              'Saramago':'Portugal'}

# Crear un objeto OrderedDict:
odescritores = collections.OrderedDict(escritores)

# Suprimir par clave/valor por el final y
# asignar tupla con datos:
escritor_ultimo = odescritores.popitem()

# Muestra tupla:
print(escritor_ultimo)  # ('Saramago', 'Portugal')

# Muestra elementos manteniendo orden original
# sin elemento borrado
print(odescritores)
```

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

```
# Suprimir par clave/valor por el principio y
# asignar tupla con datos:
escritor_primero = odescritores.popitem(last=False)

# Muestra tupla
print(escritor_primero) # ('Borges', 'Argentina')

# Muestra elementos manteniendo orden original
# sin elemento borrado
print(odescritores)

# Suprimir par clave/valor por el final y
# asignar datos a dos variables:
escritor, pais = odescritores.popitem()

print(escritor, pais)  # Rulfo México

# Muestra elementos manteniendo orden original
# sin elemento borrado
print(odescritores)

# OrderedDict([('Lorca', 'España'),
#              ('Coehlo', 'Brasil'),
#              ('Neruda', 'Chile'),
#              ('Saramago', 'Portugal')])

# Obtener un diccionario a partir de objeto OrderedDict:
diccionario_desordenado = dict(odescritores)

# Los elementos aparecerán a su libre albedrío:
print(diccionario_desordenado)
# {'Lorca': 'España', 'Coehlo': 'Brasil',
#  'Saramago': 'Portugal', 'Neruda': 'Chile'}

# Mover clave/valor al final del objeto OrderedDict:
# Mueve al final (si existe) el par de clave 'Coehlo'
odescritores.move_to_end('Coehlo')

print(odescritores)
# OrderedDict([('Lorca', 'España'), ('Neruda', 'Chile'),
#              ('Saramago', 'Portugal'), ('Coehlo', 'Brasil')])

# Mover un par clave/valor al principio del objeto OrderedDict:
# Mueve al principio (si existe) el par
odescritores.move_to_end('Coehlo', last=False)

# Muestra datos con par de Coehlo' al comienzo
print(odescritores)

# Añadir nuevos pares al final del objeto OrderedDict:
odescritores['Martí'] = 'Cuba'   # Añadir nuevo par
odescritores['Benedetti'] = 'Uruguay' # Añadir nuevo par

print(odescritores)
# Muestra: OrderedDict([('Coehlo', 'Brasil'),
#                       ('Lorca', 'España'),
#                       ('Neruda', 'Chile'),
#                       ('Saramago', 'Portugal'),
#                       ('Martí', 'Cuba'),
#                       ('Benedetti', 'Uruguay')])

# Obtener lista ordenada de tuplas del objeto OrderedDict:
lista_ordenada = sorted(odescritores.items())

print(lista_ordenada)
# [('Benedetti', 'Uruguay'), ('Coehlo', 'Brasil'),
# ('Lorca', 'España'), ('Martí', 'Cuba'),
# ('Neruda', 'Chile'), ('Saramago', 'Portugal')]

# Obtener lista ordenada de claves del objeto OrderedDict:
lista_claves = sorted(odescritores.keys())
print(lista_claves)
# ['Benedetti', 'Coehlo', 'Lorca',
#  'Martí', 'Neruda', 'Saramago']

# Mostrar todos los pares del objeto OrderedDict
# ordenados por las claves:
for clave in sorted(odescritores.keys()):
    print(clave, odescritores[clave])

# Obtener lista ordenada de valores del objeto OrderedDict:
lista_valores = sorted(odescritores.values())
```

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

```
print(lista_valores)
# ['Brasil', 'Chile', 'Cuba',
#  'España', 'Portugal', 'Uruguay']

# Obtener lista ordenada inversa de valores:
lista_valores = sorted(odescritores.values(), reverse=True)
print(lista_valores)
# ['Uruguay', 'Portugal', 'España',
# 'Cuba', 'Chile', 'Brasil']

# Agregar a un objeto OrderedDict los elementos de
# otro objeto OrdereDict:
odescritoras = collections.OrderedDict()
odescritoras['Allende'] = 'Chile'
odescritoras['Laforet'] = 'España'
odescritoras['Pizarnik'] = 'Argentina'
odescritoras['Mastretta'] = 'México'
odescritores.update(odescritoras)
```

Comparando diccionarios y objetos OrderedDict

Si se comparan dos diccionarios el resultado será ***True*** cuando contengan los mismos pares de claves/valores sin importar la posición que ocupen:

```
dicc1 = {'a':1, 'b':2}
dicc2 = {'b':2, 'a':1}
print(dicc1 == dicc2)  # True
```

En cambio, si se comparan dos objetos OrderedDict el resultado será ***True*** cuando contengan los mismos pares de claves/valores y además ocupen la misma posición:

```
odicc1 = collections.OrderedDict()
odicc1['a'] = 1
odicc1['b'] = 2
odicc2 = collections.OrderedDict()
odicc2['b'] = 2
odicc2['a'] = 1
print(odicc1 == odicc2)  # False
```

Importante: A partir de Python 3.6 la implementación de los diccionarios ha cambiado, manteniéndose en todo momento el orden en que fueron agregados los elementos a un diccionario cuando son recorridos o consultados.

Ir al índice del tutorial de Python

Publicado por Pherkad en 11:27

Etiquetas: collections, Python3

Entrada más reciente Inicio Entrada antigua

2014-2020 | Alejandro Suárez Lamadrid y Antonio Suárez Jiménez, Andalucía - España
. Tema Sencillo. Con la tecnología de Blogger.