Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

viernes, 31 de enero de 2014

Control del flujo

Control del flujo: if

La sentencia **if** se utiliza para controlar el flujo en la ejecución de un programa. Básicamente, si se cumple una condición puede hacerse que se ejecute un bloque de código, que estará sangrado, y si no se cumple o se dan otras posibilidades puede hacerse que se ejecuten otros bloques, igualmente sangrados. Para ello, se apoya en las clausulas **elif** que evalúa otras condiciones y en **else**, que en caso de no cumplirse ninguna de las condiciones anteriores ofrece la solución final. Varios ejemplos:

```
# Evaluar distintos valores numéricos

if edad <= 12:  # se evalúa la primera condición...
    precio = 2  # y si edad es menor igual que 12, precio = 2
elif 13 <= edad <= 18:  # en caso contrario se evalúa...
    precio = 3  # si edad está entre 13 y 18, precio = 3
else:  # en cualquier otro caso...
    precio = 4  # precio = 4
print('A Pagar: ' + str(precio) + ' €')  # Muestra importe

# Evaluar en una sola línea

print('par' if edad % 2 == 0 else 'impar')

# Evaluar si un valor está entre varios posibles

tecla = 'S'
if tecla in('s', 'S', 'y', 'Y'):
    print('Ha seleccionado: Sí')

# Evaluar variables booleanas

respuesta = True
if respuesta:  # Evalúa si respuesta es True
    print('Sí, es verdad')
else:
    print('Es falso')

# Evaluar variables por tipo de dato que contienen

var1 = "Python 3 para impacientes"
var2 = 3
var3 = 3.14
var4 = True
var5 = [1, 2, 3]
var6 = ('a', 'b', 'c')
var7 = {'a':1, 'b':2, 'c':3}

if type(var1) is str:
    print("'var1' es una cadena")

if type(var2) is int:
    print("'var2' es una número entero")

if type(var3) is float:
    print("'var3' es un número con decimales")

if type(var4) is bool:
    print("'var4' es un booleano")

if type(var4):
    print("'var4' es un booleano")
```

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

```
if type(var5) is list:
    print("'var5' es una lista")

if type(var6) is tuple:
    print("'var6' es una tupla")

if type(var7) is dict:
    print("'var7' es un diccionario")

# Evaluar si cadena, lista o diccionario están vacíos

var1 = ""
var2 = []
var3 = {}

if not var1:
    print("Cadena vacía")

if not var2:
    print("Lista sin elementos")

if not var3:
    print("Diccionario sin claves/valores")

# Evaluar si una variable no tiene ningún valor

var4 = None
if not var4:
    print("No tiene ningún valor")
```

Control del flujo con bucles: while

El bucle más elemental suele emplearse cuando un programa necesita repetir un número de veces un proceso hasta que una variable alcanza un valor o hasta que se cumple alguna condición predeterminada. Por ello, en muchos casos, los bucles incorporan variables que actúan como contadores que van cambiando su valor en cada ciclo o variables que representan un cambio de estado (p.e. verdadero y falso).

La sentencia **while** incluye la condición que debe cumplirse para que se ejecute el bloque de código que contiene, aunque en cualquier momento podemos salir de un bucle con **break** o cancelar el ciclo actual y continuar con **continue** la ejecución del ciclo siguiente.

```
# Ejemplo de bucle con 'break'

contador = 0
limite = 5
while contador < 11:  # el bucle termina cuando contador=10
    if contador == limite:  # o cuando alcance el valor limite
        break
    else:
        contador += 1  # contador se incrementa en 1
        print(contador, limite)

# Ejemplo de bucle con 'continue' y 'break'

x=0
y=0
limite = 5
while True:
    y+=1
    if y!=limite:
        x+=y
    else:
        break
    if y!=3:
        continue
    print(x,y)
```

En un bucle infinito en el lugar de una condición se escribirá **True** o el valor **1**. Las instrucciones del bucle se ejecutarán indefinidamente hasta que se fuerce su fin con un **break**.

```
contador = 0
while True:
    contador += 1
    if contador == 10:  # cuando valor de contador sea 10...
        break  #  ...terminará la ejecución del bucle
```

que permiten obtener de distintos modos números a...

Archivo

enero 2014 (10)

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

Control del flujo con bucles: for...in

Otra forma de construir un bucle consiste en recorrer los elementos de un lista con la sentencia **for...in**. La construcción utiliza una variable que en cada ciclo toma el valor de un nuevo elemento de la lista de la clausula **in**. El bucle terminará cuando se alcance el último elemento de la lista o cuando se fuerce su fin con **break**.

```
loteria = [12019, 23023, 90326, 40506, 89450, 21023, 13237]
for numero in loteria:
    if numero == 21023:
        print('Tiene el primer premio:', numero)
        break
```

Con **for** también se puede usar la clausula **in range** que, en su uso más elemental, recorre una secuencia de valores (ver más opciones).

```
for num in range(10):
    print(num)  # recorre del 0 al 9
```

Relacionado: Bucles eficaces con Itertools

Ir al índice del tutorial de Python

Publicado por Pherkad en 9:22

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua