Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

jueves, 30 de enero de 2014

Palabras reservadas. Variables. Cadenas.

Palabras reservadas de Python (35)

Las 35 palabras reservadas (keywords) de Python son:

and, as, assert, async, await, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield

No se pueden declarar variables, objetos, funciones y clases con estos términos. El siguiente código muestra la lista de palabras reservadas:

```
import keyword
print(keyword.kwlist)
```

Declarar variables

El signo igual "**=**" se utiliza para asignar números, booleanos, cadenas y expresiones a las variables de un programa. El tipo de la variable será el tipo del dato asignado: al declarar una variable no es necesario especificar el tipo de dato porque mientras se asigna el área de memoria necesaria el intérprete Python elige automáticamente el tipo más apropiado.

El nombre de una variable tiene que empezar por una letra del alfabeto o un guión bajo. En las siguientes posiciones podrán aparecer también números. Python distingue entre mayúsculas y minúsculas:

```
numero_entero = 5  # declara variable numérica
Numero_Entero = 5  # declara otra variable numérica distinta
x = y = z = 5  # asignación múltiple: x=5, y=5 y z=5
m, n = 5, 4 * 8  # asignación múltiple: m = 5 y n = 32
p1, p2 = (1, 2)  # asignación múltiple de tupla: p1 = 1 y p2 = 2
cadena = 'Python3'  # declara cadena alfanumérica
cadena = 'Pytonisos\tdel\tmundo\n'  # incluye tab y salto de línea
cadena = '''cadenas
            que ocupan
            varias líneas'''  # declara cadena de varias líneas'
numero_float1 = 23.45  # números con decimales
numero_float2 = 0.1e-3  # números con notación científica
numero_hexadecimal = 0x23  # número hexadecimal
booleano_verdad = True  # booleano: Verdadero
booleano_falso = False  # booleano: Falso
booleano_negar = not booleano_falso  # True, Verdadero
numero_complejo = 2 + 4j * 2  # (2+8j)

cadena1 = "4.5" # declara cadena1
cadena2 = "67" # declara cadena2
numero1 = float(cadena1) # Convierte cadena a flotante -> 4.5
numero2 = int(cadena2) # Convierte de cadena a entero -> 67

var_nula = None  # Declara variable con valor vacío o nulo
```

Asignar múltiples valores de una lista

```
a, *b = [1, 2, 3, 4]  # a toma primer valor y b lista con los demás
print('a', a)  # 1
print('b', b)  # [2, 3, 4]
```

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

Resultado:
a: 1
b: [2, 3, 4]

Variable especial _

En Python un guion bajo único '_' es una variable especial que almacena el valor de la última expresión evaluada en la ejecución de código, que puede utilizarse con posterioridad para consultar su valor, en asignaciones, en cálculos o simplemente cuando se deseen ignorar los valores obtenidos evitando con ello el tener que declarar una variable específica.

En el siguiente ejemplo la variable _ almacena en cada ciclo los valores del 0 al 9 que con posterioridad se van sumando a la variable **total**:

```
total = 0
for _ in range(10):
    total += _
    print(total, end=' ')  # 0 1 3 6 10 15 21 28 36 45
```

En la siguiente asignación múltiple la variable _ se utiliza para asignar todos los valores excepto el primero y el último que son los que interesan para un cálculo posterior:

```
var1, *_, var2 = (10, 20, 30, 40, 50, 60, 70)
print(var1+var2)  # 80
```

En una lista de comprensión la variable _ se puede utilizar para cálculos:

```
lista = [12, 15, 21, 17, 28]
cuadrados = [_**2 for _ in lista]
print(cuadrados)  # [144, 225, 441, 289, 784]
```

En el entorno interactivo Python se puede usar también para acceder al valor obtenido como resultado de evaluar la última expresión:

```
>>> 5 + 6
11
>>> _
11
```

Cadenas crudas: raw

Cualquier cadena que deba interpretarse tal como se escribe es una cadena raw o cruda. En una cadena raw se omiten los caracteres especiales expresados con la barra invertida "\". Las cadenas raw se escriben entrecomilladas y con el carácter "r" precediéndolas:

print('\\5', ' es igual que ', r'\5')

Operaciones básicas con variables

Varias variables que contienen cadenas pueden unirse (concatenarse) con el signo más "**+**". También, dos o más cadenas pueden unirse expresándolas entrecomilladas, una a continuación de la otra. Es más, cualquier cadena puede replicarse un número de veces con el operador "*****".

Con la función **type**() podemos conocer el tipo de datos de una variable y con la sentencia **del** borramos una variable de memoria. Una vez borrada si intentamos acceder a su valor se producirá un error.

Los métodos **upper**() y **lower**() devuelven una cadena en mayúsculas y minúsculas, respectivamente.

En las expresiones matemáticas es posible establecer la prioridad en la resolución de las operaciones mediante el uso de paréntesis.

Ejemplos:

```
cadena1 = 'Python'  # Declara cadena con 'Python'
cadena2 = 'Lenguaje ' + cadena1.upper()  # Lenguaje PYTHON
cadena3 = 'Lenguaje ' 'Python'  # Lenguaje Python
cadena3 = cadena1.lower() * 3  # pythonpythonpython
```

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

```
cadena4 = "Python para impacientes"
print(cadena4.title())  # Python Para Impacientes

# Uso de paréntesis en expresiones
total1 = ((24 - 10 + 2.3) * 4.3 / 2.1) ** 2

print(total1)  # 1113.9700907
print(type(cadena1))  # type str=""
print(type(total1))  # type float=""
del cadena1  # Borra la variable de memoria
print(cadena1)  # Genera un error porque no existe
```

Las funciones ord() y chr()

La función **ord()** devuelve el ordinal entero del carácter indicado y justo lo contrario hace la función **chr()** que devuelve el carácter (Unicode) que representa al número indicado.

```
ord('$')  # 36
ord('@')  # 64
ord('A')  # 65
ord('ñ')  # 241
ord('¥')  # 1200
ord('€')  # 8364
ord('娼')  # 23100

chr(36)  # '$'
chr(64)  # '@'
chr(65)  # 'A'
chr(241)  # 'ñ'
chr(1200)  # '¥'
chr(8364)  # '€'
chr(23100)  # '娼'
```

Métodos para evaluar cadenas

cadena.isalpha()

Devuelve verdadero (True) si todos los caracteres en la cadena son alfabéticos.

cadena.isalnum()

Devuelve verdadero (True) si todos los caracteres en la cadena son alfanuméricos.

cadena.isdecimal(), cadena.isdigit(), cadena.isnumeric()

Devuelve verdadero (True) si todos los caracteres en la cadena son números.

cadena.isspace()

Devuelve verdadero (True) si todos los caracteres son espacios en blanco.

cadena.islower(), cadena.isupper()

Devuelve verdadero (True) si todos los caracteres son minúsculas o mayúsculas, respectivamente.

cadena.istitle()

Devuelve verdadero (True) si el primer carácter de la cadena es mayúsculas y el resto minúsculas; o en el caso de que haya palabras separadas por espacios en blanco que cumplan la misma regla.

```
cadena1 = "Cuba"
cadena2 = "Costa Rica"
print(cadena1.isalpha())   # True
print(cadena1.isalnum())   # True
print(cadena1.isdecimal())   # False
print(cadena1.isspace())   # False
print(cadena2.istitle())   # True
if cadena1.isalpha():
    print("Todos los caracteres que contiene son alfabéticos")
```

Ir al índice del tutorial de Python

Publicado por Pherkad en 11:38

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua