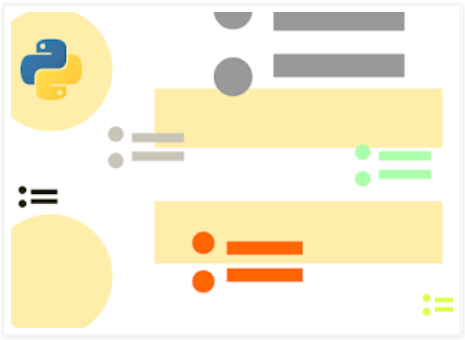
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

martes, 25 de febrero de 2020

Expresiones de asignación (:=)



Las **expresiones de asignación** es una novedad que incorpora la sintaxis de Python 3.8 para asignar valores a variables que son parte de una expresión, evitándose con ello el tener que inicializarlas con antelación.

Para este tipo de asignación se utiliza dentro de la expresión el operador morsa := (walrus *en inglés*) que añade claridad y simplicidad al código.

A continuación, varios casos de uso:

```
# Inicializar una variable antes de utilizarla
# es lo que siempre hemos hecho:

edad = 18
print(edad)  # 18

# Ahora se puede inicializar en una expresión
# que además es evaluada inmediatamente:

print(edad:=18)  # 18


# Antes, para guardar el valor de una
# función se asignaba previamente a una
# variable y después esta se podía utilizar:

def media(nota1, nota2):
    return (nota1 + nota2) / 2


notafinal = media(6, 8)
if notafinal >= 5:
    print('Aprobado con:', notafinal)

# Ahora las dos acciones se pueden
# hacer en una misma línea:

if (notafinal:=media(6, 8)) >= 5:
    print('Aprobado con', notafinal)


# Antes para hacer una asignación en
# un bucle se hacía dentro del mismo:

lista_compra = list()
while True:
    articulo = input('¿Qué necesitas comprar?: ')
    if articulo == '':
```

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

```
        print(lista_compra)
        break

    lista_compra.append(articulo)

# Ahora la misma asignación se puede hacer
# en la misma línea de while:

lista_compra = list()
while (articulo := input('Qué necesitas comprar?: ')) != '':
    lista_compra.append(articulo)

print(lista_compra)
```

Las expresiones de asignación también se pueden utilizar en listas de comprensión:

```
parcelas_m2 = [220, 320, 180, 430]
precios = [(precio_m2:=100) * sup for sup in parcelas_m2]
print(f'{precios} al precio de {precio_m2} € el m2')

# [22000, 32000, 18000, 43000] al precio de 100 € el m2
```

El uso de paréntesis en las expresiones de asignación es fundamental para delimitar exactamente el valor que se asigna a una variable:

```
# En el ejemplo que sigue a la variable no se
# asigna el valor 10, se asigna el resultado
# de comparar 10 y 5, es decir, True

if precio:=10 > 5:
    print(precio)  # True

# En este caso queda más claro que es 10 el
# valor asignado a la variable precio:

if (precio:= 10) > 5:
    print(precio)  # 10


# Los paréntesis también permiten anidar expresiones
# para una asignación múltiple:

(total:= (precio:=10) * 5)
print(precio, total)  # 10


# Pero atención, recuerda que existe una diferencia
# básica entre el uso de = y := en una asignación:

# En el siguiente ejemplo se asigna una tupla
# de 3 valores a la variable:

var1 = 0, 1, 2
print(var1)  # (0, 1, 2)

# Y en la siguiente expresión se asigna solo el
# primero de los valores: 0

(var1 := 0, 1, 2)
print(var1)  # 0

# Lo recomendable en estos casos es delimitar
# también por paréntesis los valores de la tupla:

(var1 := (0, 1, 2))
print(var1)

# Con delimitar solo los valores de la tupla no
# es suficiente. Hacerlo genera un error de
# sintaxis:

var1 := (0, 1, 2)  # SyntaxError
```

que permiten obtener de distintos modos números a...

Archivo

febrero 2020 (2)

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 23:44

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua