T01_Etiquetas_y_Palabras_Reservadas

January 29, 2021

1 Python de cero a experto

Autor: Luis Miguel de la Cruz Salas

1.1 Pythonico es más bonito

1.1.1 Etiquetas (variables).

- Son símbolos que permiten identificar la información que se almacena en la memoria de la computadora.
- Son nombres para los objetos.
- Se crean con ayuda del operador de asignación `=`.
- No se tiene que establecer explícitamente el tipo de dato.

Ejemplos válidos de etiquetas

[1]:
```
_luis = "Luis Miguel de la Cruz"
LuisXV = "Louis Michel de la Croix"
luigi = 25
luis_b = 0b01110 # Binario
luis_o = 0o12376 # Octal
luis_h = 0x12323 # Hexadecimal

# Sensibilidad a mayúsculas y minúsculas
pi = 3.14
PI = 31416e-4
Pi = 3.141592
```

[2]:
```
print(type(_luis), type(LuisXV))
print(type(luigi))
print(type(luis_b), type(luis_o), type(luis_h))
print(type(pi), type(PI), type(Pi))
```

```
<class 'str'> <class 'str'>
<class 'int'>
<class 'int'> <class 'int'> <class 'int'>
<class 'float'> <class 'float'> <class 'float'>
```

[3]:
```
ñero = 'Luismi'
ñero
```

[3]:
```
'Luismi'
```

[4]:
```
entero = int(1.0)
print(type(entero), id(entero), entero)
```

```
<class 'int'> 94473880517952 1
```

Unicode: estándar para la codificación de caracteres, que permite el tratamiento informático, la transmisión y visualización de textos de muchos idiomas y disciplinas técnicas. Unicode: universalidad, uniformidad y unicidad. Unicode define tres formas de codificación bajo el nombre UTF (Unicode transformation format): UTF8, UTF16, UTF32. Véase https://es.wikipedia.org/wiki/Unicode

[5]:
```
chr(294) #Función que transforma el código Unicode en el caracter␣
↪correspondiente
```

[5]:
```
'Ħ'
```

[6]:
```
ord('é') # Función inversa a chr()
```

[6]:
```
233
```

[7]:
```
u = chr(233)
print(u)
```

```
é
```

[8]:
```
Ħ = 3.141592 # unicode
print('{:04d} \t {} = {}'.format(ord('Ħ'), 'Ħ', Ħ)) # Impresión en decimal
print('{:04o} \t {} = {}'.format(ord('Ħ'), 'Ħ', Ħ)) # Impresión en octal
print('{:04x} \t {} = {}'.format(ord('Ħ'), 'Ħ', Ħ)) # Impresión en hexadecimal
```

```
0294     Ħ = 3.141592
0446     Ħ = 3.141592
0126     Ħ = 3.141592
```

[9]:
```
print(294,   chr(294))
print(0o446, chr(0o446))
print(0x126, chr(0x126))
```

```
294 Ħ
294 Ħ
294 Ħ
```

[10]:
```
México = 'El ombligo de la luna'
print(México)
```

```
El ombligo de la luna
```

[11]:
```
# ¿Qué codificación de carateres estoy usando?
import sys
sys.stdout.encoding
```

[11]:
```
'UTF-8'
```

[12]:
```
import unicodedata

u = chr(233) + chr(0x0bf2) + chr(3972) + chr(6000) + chr(13231)
print(u)

for i, c in enumerate(u):
    print(i, '{} {:04x} {}'.format(c, ord(c), unicodedata.category(c)), end=" ")
    print(unicodedata.name(c))
```

```
é
0 é 00e9 Ll LATIN SMALL LETTER E WITH ACUTE
1  0bf2 No TAMIL NUMBER ONE THOUSAND
2 0f84 Mn TIBETAN MARK HALANTA
3  1770 Lo TAGBANWA LETTER SA
4  33af So SQUARE RAD OVER S SQUARED
```

Véase: https://docs.python.org/3/howto/unicode.html

1.1.2 Asignación múltiple

[13]:
```
x = y = z = 25
```

[14]:
```
print(type(x), type(y), type(z))
```

```
<class 'int'> <class 'int'> <class 'int'>
```

[15]:
```
print(id(x), id(y), id(z))
```

```
94473880518720 94473880518720 94473880518720
```

[16]:
```
x, y, z = 'eje x', 'eje y', 50
```

[17]:
```
print(type(x), type(y), type(z))
```

```
<class 'str'> <class 'str'> <class 'int'>
```

[18]:
```
print(id(x), id(y), id(z))
```

```
140595088183536 140595088184880 94473880519520
```

Ejemplos inválidos de etiquetas

[ ]:
```
1luis = 20      # No se puede iniciar con un número
```

[ ]:
```
luis$ = 8.2323  # No puede contener caractéres especiales
```

[ ]:
```
for = 35        # Algunos nombres ya están reservados
```

1.1.3 Palabras reservadas

[19]:
```
help('keywords')
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

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

[ ]:
```

```