1 Python - Estructura léxica de Python

Estructura léxica de Python

La "extructura léxica" de un lenguaje de programación es el conjunto de reglas básicas que gobiernan el modo de escribir programas en ese lenguaje. Especifica detalles como el modo de definir variables, qué caracteres utilizar para indicar comentarios, cómo organizar los programas, etc.
Python es muy particular en lo que se refiere al diseño de los programas, especialmene en lo que respecta a las líneas y al sangrado.

1.1 Sumario

1.2 Líneas y sangrías

Para escribir código en Python deberíamos seguir la guía de estilo definida por la propia Comunidad Python. Esta se trata en el PEP 8 y aquí podemos ver un resumen donde, a continuación, enumeramos algunas de las normas que debemos tener más en cuenta:

- Un programa de Python se compone de un conjunto de líneas lógicas, cada una formada por una o más líneas físicas.

  Si en una línea aparece el símbolo **#** indica que todo lo que aparece a su derecha es un "comentario".
  Las líneas en blanco también son ignoradas por el intérprete de Python.

  > **¡Ojo!:** En una sesión interactiva, si pulsamos ***Enter*** sin escribir nada, ni siquiera un espacio en blanco, se termina una declaración de varias líneas.

- En Python NO es necesario añadir un delimitador para indicar el final de una línea (en otros lenguajes se utiliza el símbolo **;**).
- Cuando una sentencia es demasiado larga como para escribirse en una única línea (se recomienda que no sean mayores de 80 caracteres), se puede indicar que se sigue en la siguiente añadiendo el símbolo \ al final de la línea aún no terminada. Así y todo, : Python automáticamente une líneas físicamente adyacentes en una única línea lógica si se abre un paréntesis (**(**), un corchete (**[**) o una llave (**{**) hasta que estos sean cerrados.
- Python emplea el sangrado (*indentation*) para expresar un bloque de un programa.

  Al contrario que otros lenguajes, Python no emplea paréntesis, ni corchetes, ni llaves para delimitar los bloques.
  Todas las líneas físicas de un bloque deben tener la misma sangrado.
  La primera línea de un archivo Python no debe tener ningún tipo de sangrado.

- El estilo estándar de Python pide utilizar cuatro espacios para cada nivel de sangrado.

  Nunca se deben utilizar tabuladores (*tab*).
  Se recomienda utilizar un editor para que convierta los tabuladores en espacios (por ejemplo Pycharm).

1.3 Conjunto de caracteres

- Normalmente, los archivos de Python deben estar escritos totalmente con caracteres ASCII (códigos ASCII del 0 al 127).

  Podemos utilizar otros caracteres, pero sólo para comentarios *strings* (texto), siempre y cuando comencemos nuestro archivo Python con la siguiente línea:

```
# -*- coding: utf-8 -*-
```

1.4 Tokens

- Se denominan *tokens* a los componentes léxicos del lenguaje de programación.

  Los tipos de *token* son: identificadores, palabras clave, operadores, delimitadores y literales.
  Para separar los *tokens* se utilizan espacios.

1.5 Identificadores

- Un *identificador* es el nombre empleado para identificar una variable, una función, una clase, un módulo u otro objeto.

  Un identificador comienza con una letra (de la A a la Z o de la a a la z) o con un guión bajo (_) seguido de cero o más letras, guiones bajos y números.
  Python distingue mayúsculas de minúsculas.
  Python NO permite signos de puntuación como @, $ y %, excepto el guión bajo (_).

- Se recomienda nombrar las clases comenzando por una letra mayúscula y el resto de los identificadores por una letra minúscula.

  Si un identificador comienza por un guión bajo, significa que es "privado".

```
# Identificadores validos:
 radio
 RaDio
 contAlumnos
 Cont_Alumnos
 num1

# Identificadores NO validos:
 2E2
 Miércoles
 Cont-Alumnos
 Conjunto Primero
 print #palabras reservadas
 1_cantidad
```

  Recuerda que Python distingue las mayúsculas de las minúsculas, entonces por ejemplo **area**, **Area**, **AREA**, **aRea**, son 4 identificadores distintos. También comentar que, el último ejemplo, **aRea** no es recomendado, dado su poca legibilidad.

1.5.1 Palabras clave

- Python tiene 30 palabras reservadas que no podremos utilizar para ningún tipo de identificador:

and	del	for	is	raise
assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	with
def	finally	in	print	yield

1.5.2 Operadores

- Python utiliza caracteres no alfanuméricos y combinación de caracteres como operadores.

  Se reconocen los siguientes operadores: **+ - * / % ** // << >> & | ^ ~ < <= > >= <> != ==**

1.5.3 Delimitadores

- Python utiliza los siguientes símbolos y combinación de símbolos como delimitadores en expresiones, listas, diccionarios, etc: **( ) [ ] { , : . ` = ; += -= *= /= //= %= &= |= ^= >>= <<= **=**
- Los siguientes caracteres tienen un especial significado como parte de otros símbolos: **' " #** \
- Los caracteres **$** y **?** y todos los caracteres de control nunca podrán ser parte de un programa Python, excepto en comentarios y *strings*.
- La utilización de @ también está limitado en algunas versiones.

1.5.4 Literales

- Un *literal* es un número o un *string* que aparece directamente en un programa.

  Los siguientes son literales en Python:

```
42           # Entero
3.14         # Punto flotante
1.0j         # Imaginario
'hola'       # String
"mundo"      # Otro string
"""Buenas
noches"""    # String limitado con tres comillas
```

  Utilizando literales y delimitadores, se pueden crear distintos tipos de datos:

```
[ 42, 3.14, 'hola' ]    # Lista
( 100, 200, 300 )       # Tupla
{ 'x':42, 'y':3.14 }    # Diccionario
```

1.6 Declaraciones

- Se puede considerar un archivo fuente de Python como una secuencia de declaraciones.

1.6.1 Declaraciones simples

- Se trata de declaraciones que tienen sólo una declaración, sin ninguna más.

  Ocupan una linea lógica.
  Varias declaraciones simples pueden ocupar una única linea física si se separan éstas con ";". Pero esto no es normal hacerlo en Python.

1.6.2 Declaraciones compuestas

- Una sentencia compuesta contiene una o más sentencias simples y el control de su ejecución.

  Todas las cláusulas que componen la declaración compuesta deben estar alineadas en la misma sangría.
  Cada cláusula tiene un encabezado a partir de una palabra clave y debe terminar con dos puntos (:), seguida luego por un cuerpo, que es una secuencia de una o más sentencias.

  Cuando el cuerpo contiene varias sentencias (bloque), éstas deben estar colocadas en líneas lógicas con una sangría de cuatro espacios con respecto a la cabecera.
  El bloque léxicamente termina cuando el sangrado vuelve a la altura de la cabecera.

1.7 Problemas resueltos

1.- ¿Cuál de los identificadores siguientes es válido y cuál no?

casa	*mi-casa*	*mi*casa*	*micasa1*
_MES	*MES_1*	*MES%UNO*	mes$1
a980	*890a*	*_890*	*$a890*

Resolución:

Los identificadores **mi-casa**, **mi*casa**, **MES%UNO**, **mes$1** y **$a890** no son válidos porque utilizan sepadadores no aceptados por Pyhton. Por otra parte **890a** no es válido porque los identificadores no pueden empezar por un número.

2.- Indicar si los siguientes identificadores son el mismo para Python

Traca, *traca*, *traCa*.

Resolución:

Son distintos porque Python es sensible a los caracteres en minúscula y en mayúscula.

-- Volver