# Elementos de un programa de Python

 Por completar con ejemplos

En esta lección se tratan los elementos que componen un programa de Python

---

## Elementos de un programa

Un programa de Python es un fichero de texto (normalmente guardado con el juego de caracteres UTF-8) que contiene expresiones y sentencias del lenguaje Python. Esas expresiones y sentencias se consiguen combinando los elementos básicos del lenguaje.

El lenguaje Python está formado por elementos (*tokens*) de diferentes tipos:

- palabras reservadas (*keywords*)
- funciones integradas (*built-in functions*)
- literales
- operadores
- delimitadores
- identificadores

En esta lección se comentan los principales elementos del lenguaje. En la documentación de Python se puede consultar una descripción mucho más detallada y completa.

Para que un programa se pueda ejecutar, el programa debe ser sintácticamente correcto, es decir, utilizar los elementos del lenguaje Python respetando su reglas de "ensamblaje". Esas reglas se comentan en otras lecciones de este curso. Obviamente, que un programa se pueda ejecutar no significa que un programa vaya a realizar la tarea deseada, ni que lo vaya a hacer en todos los casos.

---

## Líneas y espacios

Un programa de Python está formado por líneas de texto.

```
radio = 5
area = 3.14159242 * radio ** 2
print(area)
```

Se recomienda que cada línea contenga una única instrucción, aunque puede haber varias instrucciones en una línea, separadas por un punto y coma (;).

```
radio = 5; area = 3.14159242 * radio ** 2
print(area)
```

Por motivos de legibilidad, se recomienda que las líneas no superen los 79 caracteres. Si una instrucción supera esa longitud, se puede dividir en varias líneas usando el caracter contrabarra (\):

```
radio = 5
area = 3.14159265358979323846 \
       * radio ** 2
print(area)
```

---

Los elementos del lenguaje se separan por espacios en blanco (normalmente, uno), aunque en algunos casos no se escriben espacios:

- entre los nombres de las funciones y el paréntesis
- antes de una coma (,)
- entre los delimitadores y su contenido (paréntesis, llaves, corchetes o comillas)

Los espacios no son significativos, es decir, da lo mismo un espacio que varios, excepto al principio de una línea.

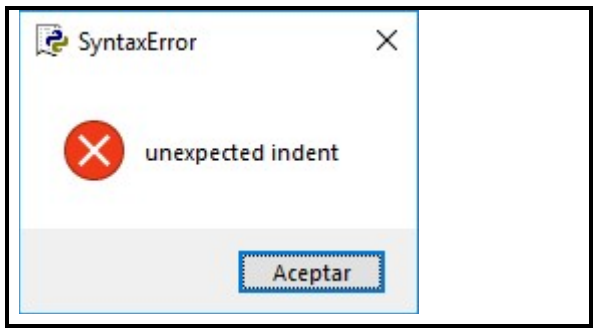
Los espacios al principio de una línea (el sangrado) indican un nivel de agrupamiento. El sangrado inicial es una de las características de Python que lo distinguen de otros lenguajes, que utilizan un caracter para delimitar agrupamientos (en muchos lenguajes se utilizan las llaves { }). Por ello, una línea no puede contener espacios iniciales, a menos que forme parte de un bloque de instrucciones o de una instrucción dividida en varias líneas.

Al ejecutar en IDLE una instrucción con espacios iniciales, se mostraría un aviso de error de sintaxis:

```
>>>     x = 42
SyntaxError: unexpected indent
```

Al ejecutar un programa, se mostraría una ventana con el mensaje de error de sintaxis:

```
    x = 42
```




---

El carácter almohadilla (#) marca el inicio de un comentario. Python ignora el resto de la línea (desde la almohadilla hasta el final de la línea).

```
area = 3.14159242 * radio ** 2       # Fórmula del área de un círculo
```

## Palabras reservadas (*keywords*)

Las palabras reservadas de Python son las que forman el núcleo del lenguaje Python. Son las siguientes:

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

Estas palabras no pueden utilizarse para nombrar otros elementos (variables, funciones, etc.), aunque pueden aparecer en cadenas de texto.

Las palabras reservadas async y await se incluyeron como tales por primera vez en Python 3.7 (publicado en junio de 2018).

---

## Literales

Los literales son los datos simples que Python es capaz de manejar:

- números: valores lógicos, enteros, decimales y complejos, en notación decimal, octal o hexadecimal
- cadenas de texto

---

## Operadores

Los operadores son los caracteres que definen operaciones matemáticas (lógicas y aritméticas). Son los siguientes:

```
+       -       *       **      /       //      %      @
<<      >>      &       |       ^       ~
<       >       <=      >=      ==      !=
```

---

## Delimitadores

Los delimitadores son los caracteres que permiten delimitar, separar o representar expresiones. Son los siguientes:

```
'       "       #       \
(       )       [       ]       {       }
,       :       .       ;       @       =       ->
+=      -=      *=      /=      //=     %=      @=
&=      |=      ^=      >>=     <<=     **=
```

## Identificadores

Los identificadores son las palabras que se utilizan para nombrar elementos creados por el usuario u otros usuarios. Esos elementos pueden ser variables u objetos que almacenan información, funciones que agrupan instrucciones, clases que combinan ambos, módulos que agrupan los elementos anteriores, etc.

Los identificadores están formados por letras (mayúsculas y minúsculas), números y el carácter guión bajo (_). Pueden ser caracteres Unicode, aunque normalmente se recomienda utilizar caracteres ASCII para evitar complicaciones a usuarios de oros países que utilizan juegos de caracteres diferentes.

El primer caracter del identificador debe ser una letra.

---

## Funciones integradas (*built-in functions*)

Una función es un bloque de instrucciones agrupadas, que permiten reutilizar partes de un programa.

Python incluye las siguientes funciones de forma predeterminada (es decir, estas funciones siempre están disponibles):

```
abs()         dict()      help()       min()      setattr()
all()         dir()       hex()        next()     slice()
any()         divmod()    id()         object()   sorted()
ascii()       enumerate() input()      oct()      staticmethod()
bin()         eval()      int()        open()     str()
bool()        exec()      isinstance() ord()      sum()
bytearray()   filter()    issubclass() pow()      super()
bytes()       float()     iter()       print()    tuple()
callable()    format()    len()        property() type()
chr()         frozenset() list()       range()    vars()
classmethod() getattr()   locals()     repr()     zip()
compile()     globals()   map()        reversed() __import__()
complex()     hasattr()   max()        round()
delattr()     hash()      memoryview() set()
```

Como se comenta en la lección Variables (1), los nombres de las funciones integradas se pueden utilizar para nombrar variables, pero entonces las funciones ya no estarán disponibles en el programa. Si se eliminan las variables, las funciones vuelven a estar disponibles.

---

## Funciones adicionales

Un programa puede definir nuevas funciones o redefinir las funciones integradas. Los nombres de las funciones no pueden coincidir con las palabras reservadas.

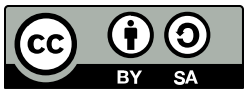
Un programa puede también importar nuevas funciones que se encuentran definidas en otros ficheros llamados módulos.

Python incluye una biblioteca de módulos (llamada Biblioteca estándar) especializados en todo tipo de tareas.

Además de la biblioteca estándar, existen miles de módulos escritos por diferentes programadores y accesibles en Internet. El principal repositorio de módulos es el Python Package Index (Índice de paquetes de Python), más conocido por PyPI.

---

Última modificación de esta página: 28 de junio de 2017




Esta página forma parte del curso **Introducción a la programación con Python** por Bartolomé Sintes Marco
que se distribuye bajo una Licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional (CC BY-SA 4.0).