

Elementos de 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** o tokens del lenguaje.

Los elementos pueden ser de diferentes tipos:

- líneas y espacios
- comentarios
- palabras reservadas (keywords)
- funciones integradas (built-in functions)
- literales
- operadores
- delimitadores
- identificadores

Una descripción más detallada de los elementos del lenguaje la podemos consultar en

https://docs.python.org/3/reference/lexical_analysis.html

Para que un programa se pueda ejecutar, el programa debe ser **sintácticamente correcto**, es decir,

debe utilizar los elementos del lenguaje Python respetando su reglas de "ensamblaje".

En cualquier caso, que un programa se pueda ejecutar (sea sintácticamente correcto) 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** o indentación) indican un **nivel de agrupamiento**. El sangrado inicial es una de las características de Python que lo distinguen de otros lenguajes, que utilizan un carácter 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 el intérprete una instrucción con espacios iniciales, se mostraría un aviso de error de sintaxis:

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

Comentarios

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).

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

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

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 permiten definir 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 otros países que utilizan juegos de caracteres diferentes.

El primer carácter 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):

<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>

```
compile()    globals()    map()         reversed()   __import__()  
complex()    hasattr()    max()         round()      delattr()  
hash()       memoryview() set()
```

Estas palabras pueden utilizarse para nombrar variables, pero entonces las funciones ya no están 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.

Referencias

Apuntes generados a partir del curso [Introducción a la programación con Python](#) que se distribuye bajo una [Licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional \(CC BY-SA 4.0\)](#).