Salida por pantalla. La función print()

En Informática, la **salida** de un programa son los datos que el programa proporciona al exterior. Aunque en los incios de la informática la salida más habitual era una **impresora**, hace muchos años que el dispositivo de salida más habitual es la **pantalla** del ordenador.

Salida por pantalla en el intérprete

Para que el intérprete muestre el valor de una variable, basta con escribir su nombre.

```
>>> a = 2
>>> a
2
```

También se puede conocer el valor de **varias variables** a la vez escribiéndolas entre comas (el intérprete las mostrará entre paréntesis), como muestra el siguiente ejemplo:

```
>>> a = b = 2
>>> c = "pepe"
>>> a
2
>>> c, b
('pepe', 2)
```

En un programa el comportamiento no es el mismo. Si creamos un programa con estas dos líneas:

```
a = 2
a
```

Al ejecutarlo no se mostrará nada por pantalla.

La función print()

En los programa, para mostrar texto o variables hay que utilizar la función print(). La función print() permite mostrar texto en pantalla. El texto a mostrar se escribe como argumento de la función:

```
print("Hola")
```

Salida por pantalla:

```
Hola
```

Las cadenas se pueden delimitar tanto por comillas dobles (") como por comillas simples (').

```
print('Hola')
```

Salida por pantalla

```
Hola                |
```

A la función `print()` se le pueden pasar varios valores separados por comas, es lo que se denominan **argumentos**. Los argumentos se muestran en el **mismo orden** y en la **misma línea**, separados por espacios:

```
print("Hola", "Adios")
```

Salida por pantalla:

```
Hola Adios
```

Al final de cada print(), Python añade automáticamente un salto de línea:

```
print("Hola")
print("Adios")
```

Salida por pantalla:

```
Hola
Adios
```

Para generar una línea en blanco, se puede escribir una orden print() sin argumentos.

```
print("Hola")
print()
print("Adios")
```

Salida por pantalla:

```
Hola

Adios
```

Si no se quiere que Python añada un salto de línea al final de un print(), se debe añadir al final el argumento end="":

```
print("Hola", end="")
print("Adios")
```

Salida por pantalla:

```
HolaAdios
```

En el ejemplo anterior, las dos cadenas se muestran pegadas. Si se quieren separar los argumentos en la salida, hay que incluir los espacios deseados (bien en la cadena, bien en el argumento end):

```
print("Hola. ",end="")
print("Adios")
print("Hasta pronto.", end=" ")
print("Buenas noches")
```

Como vimos anteriormente, para incluir comillas dentro de comillas, se puede escribir una contrabarra () antes de la comilla para que Python reconozca la comilla como carácter, no como delimitador de la cadena:

```
print("Un tipo le dice a otro: \"¿Cómo estás?\"")
print('Y el otro le contesta: \'¡Pues anda que tú!\'')
```

Salida por pantalla:

```
Un tipo le dice a otro: "¿Cómo estás?"
Y el otro le contesta: '¡Puesanda que tú!'
```

O escribir comillas distintas a las utilizadas como delimitador de la cadena:

```
print("Un tipo le dice a otro: '¿Cómo estás?'")
print('Y el otro le contesta: "¡Pues anda que tú!"')
```

Salida por pantalla:

```
Un tipo le dice a otro: '¿Cómo estás?'
Y el otro le contesta: "¡Pues anda que tú!"
```

La función print() permite incluir variables o expresiones como argmento, lo que nos permite combinar texto y variables:

```
nombre = "Pepe"
edad = 25
print("Me llamo",nombre, "y tengo", edad, "años.")
semanas = 4
print("En", semanas ,"semanas hay", 7 * semanas, "días.")
```

Salida por pantalla:

```
Me llamo Pepe y tengo 25 años.
En 4 semanas hay 28 días.
```

Se puede forzar que las variables se escriban como enteros o reales, utilizando las funciones de conversión de tipos `int()` o `float()`, respectivamente.

```
>>> print(int(2.9))
2
>>> print(float(2))
2.0
```

La función print() muestra los argumentos separados por espacios, lo que a veces no es conveniente . En el ejemplo siguiente el signo de exclamación se muestra separado de la palabra.

```
nombre = "Pepe"
print("¡Hola,", nombre, "!")
```

Salida:

```
¡Hola, Pepe !
```

Para eliminar ese espacio, se puede reducir el número de argumentos, concatenando texto y variables con el operador de concatenación suma (+):

```
nombre = "Pepe"
print("¡Hola,", nombre + "!")
print("¡Adios, " + nombre + "!")
```

Salida:

```
¡Hola, Pepe!
¡Adios, Pepe!
```

Pero hay que tener cuidado, porque no se pueden concatenar cadenas y números con el operador suma (+):

```
fecha = 2050
print("¡Feliz " + fecha + "!")
```

Salida:

```
Traceback (most recent call last):
  File "ejemplo.py", line 2, in
<module>
    print("¡Feliz " + fecha + "!")
TypeError: Can't convert 'int'
object to str
implicitly
```

En este caso, hay que convertir los números a cadenas con la función `str()` antes de concatenarlos:

```
fecha = 2050
print("¡Feliz", fecha, "!")
print("¡Feliz", str(fecha) + "!")
print("¡Feliz " + str(fecha) + "!")
```

Salida:

```
¡Feliz 2050 !
¡Feliz 2050!
¡Feliz 2050!
```

Referencias