Salida por pantalla

La función *print()* es la forma general de mostrar información por pantalla. Generalmente podemos mostrar texto y variables separándolos con comas:

```
v = "otro texto"
n = 10

print("Un texto",v,"y un número",n)
```

```
Un texto otro texto y un número 10
```

El método .format()

Es una funcionalidad de las cadenas de texto que nos permite formatear información en una cadena (variables o valores literales) cómodamente utilizando identificadores referenciados:

```
c = "Un texto '{}' y un número '{}'".format(v,n)
print(c)
```

```
Un texto 'otro texto' y un número '10'
```

También podemos referenciar a partir de la posición de los valores utilizando índices:

```
print( "Un texto '{1}' y un número '{0}'".format(v,n) )
print(c)
```

```
Un texto '10' y un número 'otro texto'
```

O podemos utilizar identificador con una clave y luego pasarlas en el format:

```
print( "Un texto '{v}' y un número '{n}'".format(n=n,v=v) )
```

```
Un texto 'otro texto' y un número '10'
```

```
print("{v},{v},{v}".format(v=v))
```

```
otro texto,otro texto,otro texto
```

Formateo avanzado

Este método soporta muchas técnicas de formateo, aquí algunos ejemplos.

Alineamiento a la derecha en 30 caracteres:

```
print( "{:>30}".format("palabra") )
```

```
                       palabra
```

Alineamiento a la izquierda en 30 caracteres (crea espacios a la derecha):

```
print( "{:30}".format("palabra") )
```

```
palabra
```

Alineamiento al centro en 30 caracteres:

```
print( "{:^30}".format("palabra") )
```

```
           palabra
```

Truncamiento a 3 caracteres:

```
print( "{:.3}".format("palabra") )
```

```
pal
```

Alineamiento a la derecha en 30 caracteres con truncamiento de 3:

```
print( "{:>30.3}".format("palabra") )
```

```
                           pal
```

Formateo de números enteros, rellenados con espacios:

```
print("{:4d}".format(10))
print("{:4d}".format(100))
```

```
print("{:4d}".format(1000))
```

```
  10
 100
1000
```

Formateo de números enteros, rellenados con ceros:

```
print("{:04d}".format(10))
print("{:04d}".format(100))
print("{:04d}".format(1000))
```

```
0010
0100
1000
```

Formateo de números flotantes, rellenados con espacios:

```
print("{:7.3f}".format(3.1415926))
print("{:7.3f}".format(153.21))
```

```
  3.142
153.210
```

Formateo de números flotantes, rellenados con ceros:

```
print("{:07.3f}".format(3.1415926))
print("{:07.3f}".format(153.21))
```

```
003.142
153.210
```

Format simplificado

La actualización de Python 3.6 trajo la novedad de poder concatenar variables y cadenas de una forma muy cómoda sin utilizar el *format()*.

Hasta ahora para concadenar hacíamos lo siguiente:

```
nombre = "Héctor"
texto = "Hola {}".format(nombre)
print(texto)
```

```
```

```
Hola Héctor
```

La nueva sintaxis nos permite ahorrarnos el método:

```
nombre = "Héctor"
texto = f"Hola {nombre}"
print(texto)
```

```
Hola Héctor
```

Sólo tenemos que indicar **f** antes de la cadena y sustituir las variables por sus nombre.

Última edición: 25 de Septiembre de 2018