
 No description has been provided for this image No description has been provided for this image

Python Basics I: Print, Comentarios y otros aspectos básicos

En este notebook vamos a profundizar un poquito más en el uso de la función `print`, además de ver cómo usar los comentarios, la importancia del flujo secuencial de ejecución de python y terminaremos con el uso del comando para liberar memoria, `del`.

Contenidos

- [Print](#)
- [Comentarios](#)
- [Flujos de ejecución](#)
- [Del](#)

Print

[al índice](#)

Hasta ahora hemos visto en Jupyter dos tipos de celdas:

- **Markdown:** básicamente texto para las explicaciones
- **Código:** donde insertamos el código Python y lo ejecutamos

Las celdas de código, no solo corren el código y realizan todas las operaciones que le indiquemos, sino que también tienen un output, una salida de ese código. Tenemos dos opciones para ver el output del código, o bien mediante `print`, o poniendo una variable al final de la celda. En este último caso, veremos el valor de esa variable

```
In [1]: ingresos = 1000
gastos = 100
beneficios = ingresos - gastos
```

```
In [2]: print(ingresos)
print(gastos)
print("Los beneficios han sido de:")
print(beneficios)
```

1000

100

Los beneficios han sido de:

900

Si hacemos un print de un texto, podemos concatenar valores de variables mediante `%s`.

```
In [5]: mes = "junio"
print("Los beneficios de %s han sido %s millones" %(mes,beneficios))
```

Los beneficios de junio han sido 900 millones

Otra opción si queremos imprimir por pantalla varios strings, es separarlos por comas en el `print`. Fíjate que le añades espacios en la salida cada vez que ponemos un valor nuevo entre comas.

```
In [7]: print("Los beneficios de %s han sido %s millones" %(beneficios, mes))
```

Los beneficios de 900 han sido junio millones

```
In [8]: mes = "agosto"
print("Los beneficios de ", mes, "han sido", beneficios, "millones")
```

Los beneficios de agosto han sido 900 millones

Comentarios

[al índice](#)

Se trata de texto que va junto con el código, y que el intérprete de Python ignora por completo. Muy útil para documentar y explicar el código

```
In [9]: # Comentario de Línea
# print("Esto lo ignora")

print("Esto sí lo ejecutará")
```

Esto sí lo ejecutará

```
In [10]: print("Fin del programa") #Esta línea indica que hemos terminado
```

Fin del programa

```
In [11]: '''
Comentario multilínea
Puedo usar varias líneas sin tener que repetir el #
```

```
'''
'''
Los comentarios multilíneas pueden ir
entre comillas dobles, y tampoco se ejecutan o interpretan
este
print("No se va a ejecutar")
'''
print("Este print se va a ejecutar")
```

Este print se va a ejecutar

In []:

IMPORTANTE. SIEMPRE comentar el código. Nunca sabes quién lo puede heredar. Te dejo en [este link](#) una guía interesante sobre cómo comentar tu código.

Flujos de ejecucion

[al indice](#)

Los programas de Python se ejecutan secuencialmente, por lo que el orden en el que escribas las operaciones es determinante

```
In [13]: ventas_jun_jul = ventas_junio + ventas_julio

ventas_junio = 100
ventas_julio = 150
```

```
-----
---
NameError                                Traceback (most recent call la
st)
Cell In[13], line 1
----> 1 ventas_jun_jul = ventas_junio + ventas_julio
      3 ventas_junio = 100
      4 ventas_julio = 150

NameError: name 'ventas_junio' is not defined
```

```
In [14]: ventas_junio
```

```
-----
---
NameError                                Traceback (most recent call la
st)
Cell In[14], line 1
----> 1 ventas_junio

NameError: name 'ventas_junio' is not defined
```

Da error, primero tenemos que declarar las ventas, y luego sumarlas.

```
In [15]: ventas_junio = 100  
ventas_julio = 150  
ventas_jun_jul = ventas_junio + ventas_julio
```

```
In [18]: print("Las ventas de junio a julio son", ventas_jun_jul)
```

Las ventas de junio a julio son 250

¿Cuándo acaba una línea de código? El salto de línea lo interpreta Python como una nueva instrucción. En muchos lenguajes, como Java hay que especificar el fin de la sentencia de código mediante `;`. En Python no es necesario, aunque se puede usar igualmente.

```
In [19]: altura = 1.80; peso = 75  
print(altura, peso)
```

1.8 75

```
In [20]: x,y,z = 12,23,45
```

```
In [ ]: x = 12  
y = 23  
z = 45
```

Del

[al índice](#)

Es la sentencia que usaremos para borrar una variable. La verdad, no se suelen borrar variables. Vamos desarrollando los programas de Python sin preocuparnos de limpiar aquellas variables que no usamos. Normalmente no borrarlas no suele ser un problema, pero cuando manejamos un gran volumen de datos, podemos sufrir problemas de rendimiento ya que **las variables ocupan memoria**.

Cuando las variables son las que hemos visto hasta ahora, no pasa nada, pero si son más pesadas, como por ejemplo datasets de imágenes que ocupan mucho, sí va a venir bien eliminar aquellas que no usemos.

```
In [21]: altura = 1.85  
del altura  
print(altura)
```

```
-----  
---  
NameError                                Traceback (most recent call last)  
Cell In[21], line 3  
      1 altura = 1.85  
      2 del altura  
----> 3 print(altura)  
  
NameError: name 'altura' is not defined
```

In []: