Fundamentos de Programación

Tutorial: Herramientas para desarrolladores

Aprender Python como principiantes puede ser un poco abrumador. ¿Por donde empezar? ¿Dónde podéis consultar información? ¿Cómo relovéis errores? ¿Qué estilo usáis a la hora de programar? ¿Qué hacéis una vez que hayáis aprendido lo básico? En este tutorial, hemos compilado una lista de algunos de los mejores recursos de consulta y aprendizaje de Python para principiantes.

Cómo resolver errores y excepciones

Stack Overflow es una plataforma muy popular, con un formato estilo foro, donde se pueden obtener respuestas detalladas, consejos y trucos para solucionar cualquier problema relacionado con la programación. Os será muy útil para encontrar soluciones a problemas o errores en vuestro código. Sin embargo, evitad copiar código del foro directamente. El código publicado en *Stack Overflow* es gratuito, bajo el atributo *Creative Commons* y con licencia para compartir. Esto significa que si usáis código de *Stack Overflow*, estáis obligado a dar crédito al autor original y a hacer que vuestro código también esté disponible de forma gratuita. En lugar de copiar código, intentad usar la información que compartan los demás usuarios para aprender los conceptos y aplicarlos mientras escribís vuestro propio código original. Además la plataforma ofrece una serie de funcionalidades que os serán de gran utilidad para aprovechar *Stack Overflow* al máximo:

- **Preguntas relacionadas**: a la derecha de la página encontraréis un listado de preguntas enlazadas (*Linked*) relacionadas (*Related*). Si la pregunta que habéis encontrado o la respuesta no es satisfactoria, os aconsejo echar un ojo a estos listados, suelen ser muy útiles para navegar y finalmente encontrar el hilo que estáis buscando, además las preguntas relacionadas o *Related* también os pueden servir para profundizar en el tema que estáis explorando.
- **Tags**: si hacéis clic en una etiqueta o *tag* de los que aparecen debajo de una pregunta, llegaréis a una página que muestra todas las preguntas dentro de esa etiqueta. Os puede servir para buscar más información sobre una misma categoría o tema.
- **Puntuación de las respuestas**: fijaros en que las respuestas aparecen ordenadas según su número de votos o puntuación. Esto os resultará muy útil para diferenciar rápidamente las respuestas más útiles o fiables.

Antes de publicar cualquier duda en *Stack Overflow* buscadla y aseguraos de que nadie ha formulado la misma pregunta antes. Muchos de vosotros os encontráis en una fase muy inicial de aprendizaje dentro del mundo de la programación y es muy probable que las dudas que os surjan ya hayan sido tratadas en la plataforma. Buscar antes de preguntar es una solución más rápida y además es una buena práctica entre programadores. Si no encontráis la pregunta que buscáis y decidís abrir un hilo, tened en cuenta que las preguntas que son demasiado amplias, poco claras, incompletas o basadas principalmente en opiniones pueden ser cerradas por la comunidad hasta que se mejoren. Veamos cómo hacer buen uso de *Stack Overflow* a través de un ejercicio.

Si ejecutáis el siguiente código veréis que devuelve un error debido a una sintaxis incorrecta: `SyntaxError: invalid syntax`.

In [2]:

```
while True print('Hola mundo')
```

```
  File "<ipython-input-2-9343c17d223d>", line 1
    while True print('Hola mundo')
                    ^
SyntaxError: invalid syntax
```

Como quizás ya sabréis, Python es un lenguaje de programación que usa un intérprete para "traducir" vuestro código al lenguaje máquina. Cuando se ejecuta un script de Python, el intérprete evaluará vuestro código antes de que realmente lo ejecute. En caso de que el intérprete encuentre un error, detendrá el programa e informará del tipo de error y en qué parte del código ocurrió.

En el ejemplo anterior podéis ver que el intérprete de Python devolvió (advirtió sobre) un error de sintaxis o `SyntaxError`. Esto significa que el intérprete encontró un error en la forma en que se escribió este script (sintaxis) y, como consecuencia, detiene su ejecución inmediatamente.

En el mensaje de error, además de indicar el tipo de error que se ha detectado, veréis que el parser repite la línea infractora y muestra una pequeña "flecha" que apunta al punto más temprano de la línea donde se detectó el error. El error es causado por (o

muestra una pequeña "flecha" que apunta al punto más temprano de la línea donde se detectó el error. El error es causado por (o al menos detectado en) el comando que precede a la flecha. En este caso, el error se detecta en la función `print`. Esto se debe a que el intérprete intentó analizar la función `print` cuando la declaración del 'while' statement todavía no estaba terminada. Recordad que en Python, por convención, el uso de `while`, `for`, `if`, etc, se tiene que acompañar de `:`.

Además, fijaros en que el nombre del archivo y el número de línea se imprimen para que sepáis dónde buscar en caso de que la entrada provenga de un script con múltiples líneas o archivos.

Ejercicio 1

Ejecuta el siguiente código e indicad qué tipo de error habéis obtenido. ¿A qué elemento del código hace referencia este error y a qué crees que se debe?

In [ ]:

```
4 + spam*3
```

Respuesta: [Detalla aquí tu respuesta]

Cómo vemos en el ejercicio anterior, más allá de los errores de sintaxis, un código también nos puede retornar otros tipos de error. Esto se debe a que a veces una expresión es sintácticamente correcta pero aún así genera un error cuando se intenta ejecutar. Estos errores detectados durante la ejecución se denominan excepciones y no siempre son críticos.

Los errores anteriores son bastante intuitivos a la hora de reconocer sus causas y darles solución, sin embargo, otras veces nos encontraremos con errores que no sabemos qué significan exactamente o que no sabemos cómo resolver. ¿Qué debemos hacer en estos casos? Aquí es cuando es fundamental que el programador haga uso de los recursos online y el soporte de comunidades de programadores como *Stack Overflow*. Veamos un ejemplo.

El siguiente código nos devuelve un error de tipo `TypeError`:

In [ ]:

```
longitud = input('Indica la longitud en m:')
anchura = input('Indica la anchura en m: ')
metros_cuadrados = longitud * anchura;
print (metros_cuadrados)
```

Ejercicio 2

Busca el error entre comillas en un motor de búsqueda (ej. Google) o en *Stack Overflow* y lee detenidamente el primer resultado que obtengas.

¿Qué crees que significa este error y cómo podrías solucionarlo? En base a los resultados obtenidos en tu búsqueda, intenta reescribir el código para resolver el `TypeError`.

In [ ]:

```
## Respuesta: [Ecribe aquí tu respuesta]
```

Librerías

Más allá de los errores y las excepciones que nos podamos encontrar, también es importante que sepamos acceder a los recursos y a la documentación de las librerías (de software) que estamos usando o que necesitaremos usar en nuestro código. Una librería es una colección de archivos (denominados módulos) que contienen funciones para su uso por otros programas. También pueden contener valores de datos (por ejemplo, constantes numéricas) y otras cosas. La librería más usada en Python es la *Python Standard Library*, que está formada por un extenso conjunto de módulos que viene con la instalación de Python por defecto. Algunos de estos módulos son `random`, `math`, o `time` y contienen funciones como `math.cos(x)`, que retorna el coseno de un valor `x`, o `time.sleep(x)`, que pausa la ejecución durante `x` segundos. ¿Pero cómo podemos saber qué hace cada uno de estos módulos y funciones y cómo debemos usarlas? Tenemos varias opciones:

1- Podemos acceder a la información sobre las librerías a través de su documentación, como por ejemplo en el caso de la librería estándar de Python, a través de la página https://docs.python.org/3/library/index.html

2- Alternativamente podemos usar el comando `help(argumento)` para conocer el contenido de un módulo de biblioteca. Para hacer esto, siempre tened en cuenta importar la librería o módulo pertinente antes de hacer la llamada a la función

`help(argumento)`. Recordad también que deberéis pasar un argumento, si no se proporciona ningún argumento, el sistema de ayuda interactiva se inicia en la consola del intérprete y tendréis que introducir `quit` para salir. Por ejemplo, para usar la función `help(argumento)` para obtener información sobre la función print, escribiremos:

In [ ]:

```
help(print)
```

Ejercicio 3

Dadas las variables año, mes y día, ¿cómo generarías una fecha en el formato iso estándar? ¿Qué módulo de biblioteca estándar podría servirte? ¿Qué función seleccionarías de ese módulo? Investiga los módulos de la *Python Standard Library*, no dudes en usar un motor de búsqueda o *Stack Overflow* para profundizar, e intenta escribir un programa que use la función.

In [ ]:

```
año = 2016
mes = 10
dia = 22

## Respuesta:
```

Guía de estilo

Algo que puede ser muy confuso cuando nos iniciamos en un nuevo lenguaje de programación es leer el código que escriben otras personas. Para evitar confusiones y facilitar la legibilidad entre programadores de diferentes lugares, culturas y formaciones, es una buena práctica seguir unos estándares de estilo. PEP 8, a veces escrito PEP8 o PEP-8, es un documento que proporciona las pautas y las mejores prácticas sobre cómo escribir código Python. Fue escrito en 2001 por Guido van Rossum, Barry Warsaw y Nick Coghlan. El enfoque principal de PEP 8 es mejorar la legibilidad y la coherencia del código Python.

La buena noticia es que no tendréis que memorizar todo el contenido de la guía de memoria. Hay muchos complementos y herramientas de línea de comandos que verificarán si vuestro código es compatible con la guía de estilo PEP8 e incluso corregirán algunos por vosotros, por ejemplo, espacios en blanco finales. Pero no creáis que vuestro código es compatible con PEP8 si un plugin no os da ningún warning. Por ejemplo, estos plugins no pueden comprobar si vuestras variables tienen nombres significativos o si los nombres son coherentes. Por eso os recomendamos leer la Guía de estilo PEP8 de vez en cuando, para aseguraros de que no estáis pasando por alto ningún aspecto relacionado con el estilo.

Ejercicio 4

Por ejemplo, autopep8 se puede utilizar en Jupyter para reformatear/embellecer el código en la celda de código de un cuaderno. Sólo necesitaréis instalar el paquete correspondiente:

In [3]:

```
!pip install autopep8
```

```
/bin/sh: pip: command not found
```