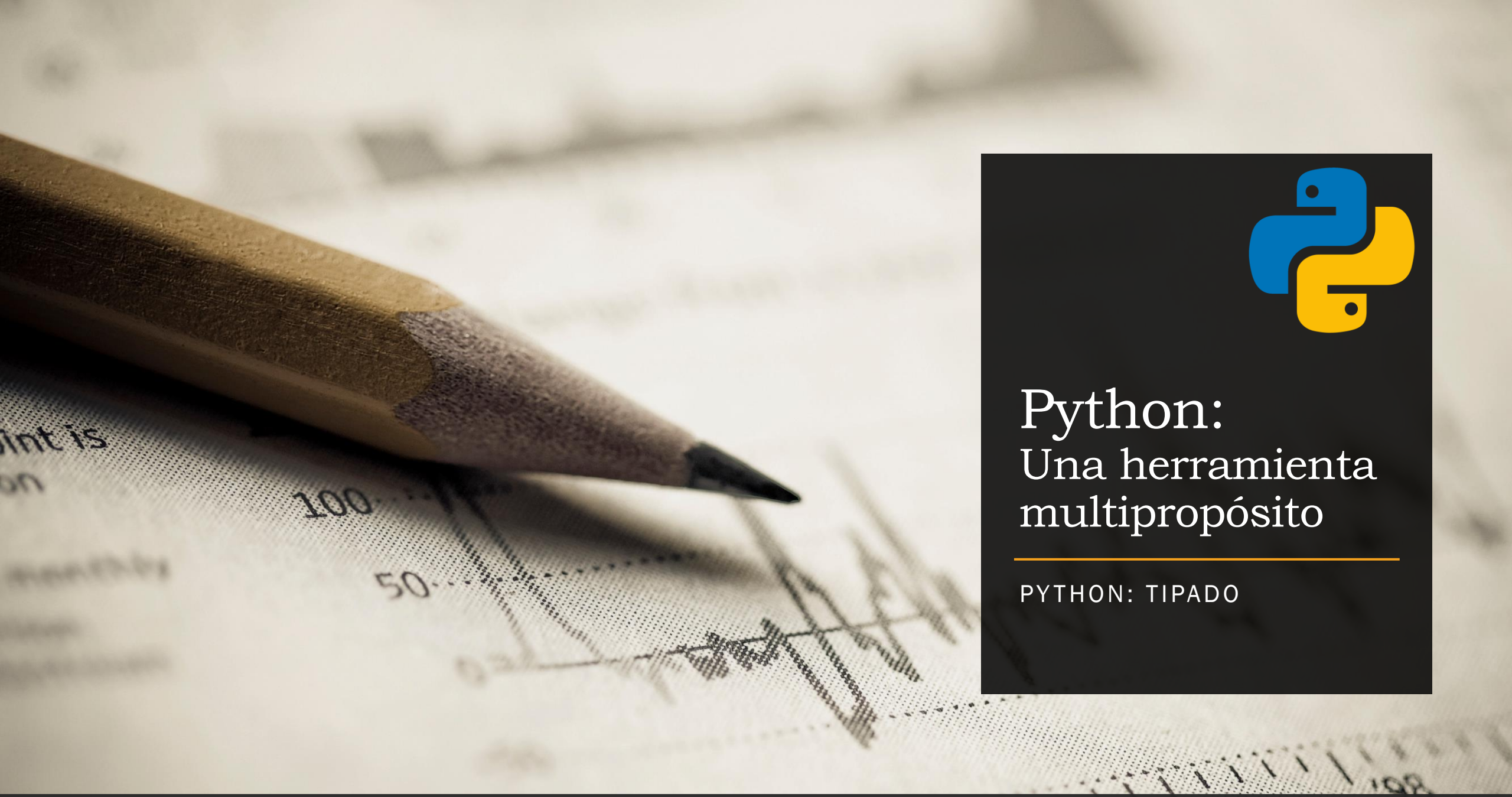

Python:
Una herramienta multipropósito

PYTHON: TIPADO

Recordando

Python es un lenguaje dinámico, es decir, las variables pueden acceder a cualquier tipo y no es necesario que el mismo se preserve durante la duración del script siendo ejecutado.

a = 1

a = “abcd”

Anotaciones de tipo

A pesar de que Python no verifica el tipado de las variables durante la ejecución del código, a veces es importante proveer anotaciones, o indicaciones que permitan al usuario entender las características del código siendo interpretado.

Asimismo, permite a herramientas automáticas detectar errores que el usuario puede haber ignorado.

Anotaciones de tipo

Veamos como en C por ejemplo podemos ver que los tipos vienen por defecto, y de hecho no podemos evitarlos ya que son chequeados por el compilador al momento de transformar el código.

```c
#include<stdio.h>
int main()
{
   char hello[] = "hello world!";
   printf("%s\n", hello);
   return 0;
}
```

Python no los requiere, y de hecho permitirá que el código corra hasta el momento en que exista un problema (por ejemplo aplicar una operación matemática a una cadena de texto).

Anotaciones de tipo

Hay ventajas para la falta de tipos ya que permite que el código sea más simple, y en algunos casos reduzca su complejidad.

Sin embargo, hay beneficios de tener tipos ya que permite que nuestro código sea explícito y no tenga ambigüedad. Es decir, podemos prevenir que una variable anotada como entero no se convierta en cadena de texto por accidente, o evitar pasar argumentos incorrectos a nuestras funciones.

Anotaciones de tipo

En Python las anotaciones de tipo no son como en lenguajes como C, en donde usarlas de manera equivocada terminará la ejecución del programa con errores, nunca habrán errores!!!!

¿Entonces para qué usarlas?

Porque como vimos antes, declarar tipos permite que nuestro código sea más explícito y claro.

Usándolas, herramientas conocidas como **linters** pueden detectar que el código no esté haciendo algo que no debe.

Anotaciones de tipo

Desde Python 3.9, podemos definir tipado en las propiedades de nuestro código de la siguiente manera:

```python
cadena_de_texto: str = "hello world!"

def sumar(x: int, y: int) -> int:
    return x + y

valor_sumado: int = sumar(7, 4)
```

Como podemos notar, mediante los dos puntos podemos determinar que clase de variable tenemos, en el caso de las funciones, podemos asimismo indicar el tipo retornado por la misma **->**.

Anotaciones de tipo: Alerta

Usar anotaciones de tipo permite a nuestro IDE, *linter*, o entorno de desarrollo señalar problemas en nuestro código a manera de **alerta temprana**, antes de que el mismo llegue a las manos de sus usuarios.

La siguiente clase aprenderemos como automatizar estos chequeos durante nuestro proceso de desarrollo.

Combinando tipos

Además de los tipos básicos int, str, float, dict, etc. También podemos combinarlos para crear definiciones de tipos más complejas. Por ejemplo, dos una variable que querremos sea de varios tipos.

```
int_o_cadena: str | int = 40
```

Un mundo más complicado

Sin embargo, esto es un mundo más complicado.

Podemos crear uniones, y muchas clases de tipos dinámicos, tipos condicionales.

Para aprender más, dirigirse a

typing — Support for type hints — Python 3.10.5 documentation

Conclusión

Preguntas?