Python Programming

Módulo 2

Lanzar excepciones

Lanzar excepciones

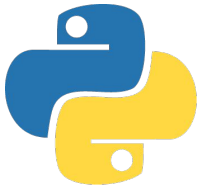
Ahora volvamos al ejemplo de nuestra rudimentaria función `sumar()`, que se veía así:

```
def sumar(a, b):
     return a + b
```

Como se trata de una operación aritmética, lo ideal sería poder chequear que los argumentos sean números enteros (`int`) o de coma flotante (`float`). Si bien dos colecciones del mismo tipo pueden concatenarse vía el operador **+**, no es la intención de nuestra función.

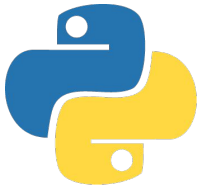
Utilizando la función incorporada `isinstance()` podemos chequear si un objeto es de un tipo de dato determinado.

```
def sumar(a, b):
    if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):
        raise TypeError("Se requieren dos numeros.")
    return a + b

print(sumar(7, 5)) #Imprime 12.
out: 12

print(sumar(2.5, 3.5))  # Imprime 6.
out: 6.0

print(sumar([1, 2], [3, 4]))  # Lanza la excepción.
out:
raise TypeError("Se requieren dos numeros.")
TypeError: Se requieren dos numeros.
```



El segundo argumento de `isinstance()` puede ser un tipo de dato o una tupla. Si es un tipo de dato, retorna `True` si el primer argumento efectivamente coincide con ese tipo de dato; si es una tupla, retorna `True` si el primer argumento coincide con alguno de los tipos de datos contenidos en ella.

Por convención, se estila lanzar `TypeError` cuando se quiere indicar que se obtuvo un argumento de un tipo inesperado, y `ValueError` cuando el tipo es correcto pero el valor no lo es.



¡Sigamos trabajando!