SESIÓN 17 · MÓDULO 2

Excepciones a fondo

Lanzar excepciones, crear las propias y controlar exactamente qué parte de un `try` se ejecuta.

Antonio Chanes Espigares · CC BY-NC-ND 4.0

Hoy

0:00	Corrección de la tarea 16.
0:15	Situación: un parte de incidencias; cada tipo de avería va a un técnico distinto.
0:25	**Predice**. El `finally` imprime antes que el `print` de fuera, aunque haya un `return`.
0:45	Construcción: `raise`, `else`, `finally`, orden de los `except`, varios tipos en un `except`, jerarquía hasta…
1:20	Ejercicios del capítulo.
1:45	Preguntas tipo examen y tarea 17.

Para empezar

Un parte de incidencias; cada tipo de avería va a un técnico distinto.

Predice

Escribe en papel qué crees que pasará. Después, lo ejecutamos.

```python
def f(x):
    try:
        return 10 / x
    except ZeroDivisionError:
        return "div0"
    finally:
        print("f", x)

print(f(2))
print(f(0))
```

Predice

Escribe en papel qué crees que pasará. Después, lo ejecutamos.

```python
def f(x):
    try:
        return 10 / x
    except ZeroDivisionError:
        return "div0"
    finally:
        print("f", x)

print(f(2))
print(f(0))
```

LO QUE SALE

`f 2` · `5.0` · `f 0` · `div0`

`finally` se ejecuta antes de que el valor devuelto llegue al `print`.

Lanzar tus propias excepciones

Excepciones · 3

```
def raiz(x):
    if x < 0:
        raise ValueError("número negativo")
    return x ** 0.5

try:
    r = raiz(-4)
except ValueError as e:
    r = None
    print("Error:", e)
print(r)
```

`else`, `finally` y el orden de los `except`

Excepciones · 4

```python
def dividir(a, b):
    try:
        resultado = a / b
    except ZeroDivisionError:
        print("No se puede dividir entre 0")
    else:
        print("Resultado:", resultado)    # solo si NO hubo error
    finally:
        print("Fin del intento")          # SIEMPRE, haya error o no

dividir(10, 2)
dividir(10, 0)
```

Tus propias clases de excepción

Excepciones · 5

```python
class NotaInvalida(Exception):
    pass

def comprobar(nota):
    if not 0 <= nota <= 10:
        raise NotaInvalida(f"{nota} no está entre 0 y 10")
    return nota

try:
    comprobar(12)
except NotaInvalida as e:
    print("Error:", e)
```

Varios tipos a la vez, la jerarquía entera y volver a lanzar

Excepciones · 6

```
def convierte(texto, divisor):
    try:
        return int(texto) / divisor
    except (ValueError, ZeroDivisionError) as e:
        print("No se puede:", type(e).__name__)
        return None

print(convierte("8", 2))
print(convierte("ocho", 2))
print(convierte("8", 0))
```

Manos a la obra

Abre el cuaderno de la sesión en Colab, con los ejemplos de hoy y estos ejercicios:

1. ¿Qué imprime?
2. Pedir un número
3. El orden. ¿Qué imprime este programa? ¿Cómo lo arreglarías?



Cuaderno en Colab

Tarea 17 · Un entero en su rango

1. Crea la clase `FueraDeRango`, hija de `ValueError`, cuyo constructor reciba el valor, el mínimo y el máximo, y prepare…
2. Escribe `a_entero(texto, minimo, maximo)`: convierte con `int` y, si el número no está en el rango, lanza…
3. Escribe `pedir_entero(pregunta, minimo, maximo)`, que pregunta con `input` hasta recibir un dato válido.
4. Usa la rama `else` del `try` para devolver el número.
5. ¿Qué pasaría si capturaras `ValueError` antes que `FueraDeRango`?

COMPROBACIÓN Si se escribe `doce`, luego `150` y luego `15`, con el rango 0–120, salen dos avisos distintos y se devuelve 15.

Antes de irte

Las **preguntas tipo examen** del capítulo, en directo: se corrigen al momento.



Preguntas del capítulo

LA PRÓXIMA SESIÓN

Sesión 18 · Clases y objetos

Programación en Python · Sesión 17 · Excepciones a fondo