# Puntos Clave: Excepciones

1. En Python, existe una distinción entre dos tipos de errores:

- **Errores de sintaxis** (errores de análisis), que ocurren cuando el analizador encuentra una sentencia de código que no es correcta. Por ejemplo:

El intentar ejecutar la siguiente línea:

```
print("Hola, ¡Mundo!)
```

Provocará un error del tipo *SyntaxError*, y da como resultado el siguiente (o similar) mensaje que se muestra en la consola:

```
  File "main.py", line 1

    print("Hola, ¡Mundo!)
                         ^
SyntaxError: EOL while scanning string literal
```

**salida**

Presta atención a la flecha: indica el lugar donde el analizador de Python ha tenido problemas. En este caso, la doble comilla es la que falta. ¿Lo notaste?

- **Excepciones**, ocurren incluso cuando una sentencia o expresión es sintácticamente correcta. Estos son los errores que se detectan durante la ejecución, cuando tu código da como resultado un error que no es *incondicionalmente fatal*. Por ejemplo:

El intentar ejecutar la siguiente línea:

```
print(1/0)
```

Provocará una excepción *ZeroDivisionError*, y da como resultado el siguiente (o similar) mensaje que se muestra en la consola:

```
Traceback (most recent call last):
  File "main.py", line 1, in
    print(1/0)
ZeroDivisionError: division by zero
```

**salida**

Presta atención a la última línea del mensaje de error; en realidad, te dice lo que sucedió. Existen muchos diferentes tipos de excepciones,
como *ZeroDivisionError*, *NameError*, *TypeError*, y muchas mas; y esta parte del mensaje te informa qué tipo de excepción se ha generado. Las líneas anteriores muestran el contexto en el que ha ocurrido la excepción.

2. Puedes "capturar" y manejar excepciones en Python usando el bloque *try-except*. Por lo tanto, si tienes la sospecha de que cualquier fragmento de código en particular puede generar una excepción, puedes escribir el código que la manejará con elegancia y no interrumpirá el programa. Observa el ejemplo:

```
while True:
    try:
        number = int(input("Ingresa un número entero: "))
        print(number/2)
        break
    except:
        print("Advertencia: el valor ingresado no es un número
válido. Intenta de nuevo...")
```

El código anterior le pide al usuario que ingrese un valor hasta que el valor ingresado sea un número entero válido. Si el usuario ingresa un valor que no se puede convertir a un int, el programa imprimirá en la consola `Advertencia: el valor ingresado no es un número válido. Intenta de nuevo...`, y pide al usuario que ingrese un número nuevamente. Veamos que sucede en dicho caso.

1. El programa entra en el bucle *while*.
2. El bloque `try` se ejecuta y el usuario ingresa un valor incorrecto, por ejemplo: `¡hola!`.
3. Se genera una excepción y el resto del código del bloque `try` se omite. El programa salta al bloque `except`, lo ejecuta, y luego se sigue al código que se encuentra después del bloque *try-except*.

Si el usuario ingresa un valor correcto y no se genera ninguna excepción, las instrucciones subsiguientes al bloque *try*, son ejecutadas. En este caso, los excepts no se ejecutan.

3. Puedes manejar múltiples excepciones en tu bloque de código. Analiza los siguientes ejemplos:

```
while True:
    try:
        number = int(input("Ingresa un número entero: "))
        print(5/number)
```

```
        break
    except ValueError:
        print("Valor incorrecto.")
    except ZeroDivisionError:
        print("Lo siento. No puedo dividir entre cero.")
    except:
        print("No se que hacer...")
```

Puedes utilizar varios bloques *except* dentro de una sentencia *try*, y especificar nombres de excepciones. Si se ejecuta alguno de los `except`, los otros se omitirán. Recuerda: puedes especificar una excepción integrada solo una vez. Además, no olvides que la excepción por **defecto** (o genérica), es decir, a la que no se le especifica nombre, debe ser colocada **al final** (utiliza las excepciones más específicas primero, y las más generales al último).

También puedes especificar y manejar múltiples excepciones integradas dentro de un solo bloque *except*:

```
while True:
    try:
        number = int(input("Ingresa un número entero: "))
        print(5/number)
        break
    except (ValueError, ZeroDivisionError):
        print("Valor incorrecto o se ha roto la regla de división
entre cero.")
    except:
        print("Lo siento, algo salió mal...")
```

4. Algunas de las excepciones integradas más útiles de Python son: *ZeroDivisionError*, *ValueError*, *TypeError*, *AttributeError*, y *SyntaxError*. Una excepción más que, en nuestra opinión, merece tu atención es la excepción *KeyboardInterrupt*, que se genera cuando el usuario presiona la tecla de interrupción (CTRL-C o Eliminar). Ejecuta el código anterior y presiona la combinación de teclas para ver qué sucede.

Para obtener más información sobre las excepciones integradas de Python, consulta la documentación oficial de Python aquí.

5. Por último, pero no menos importante, debes recordar cómo probar y depurar tu código. Utiliza técnicas de depuración como depuración de *impresión*; si es posible, pide a alguien que lea tu código y te ayude a encontrar errores o mejorarlo; intenta aislar el fragmento de código que es problemático y susceptible a errores, **prueba tus funciones** aplicando valores de argumento predecibles, y trata de **manejar** las situaciones en las que alguien ingresa valores incorrectos; **comenta** las partes del código que ocultan el problema. Finalmente, toma descansos y vuelve a tu código después de un tiempo con un par de ojos nuevos.

**Ejercicio**

¿Cuál es la salida del siguiente programa si el usuario ingresa un `0`?

```
try:
    value = int(input("Ingresa un número entero: "))
    print(value/value)
except ValueError:
    print("Entrada incorrecta...")
except ZeroDivisionError:
    print("Entrada errónea...")
except:
    print("¡Buuuu!")
```

➔ Entrada errónea…