Python Exceptions

What Are Exceptions?

Exceptions in Python occur when an **error** disrupts the normal flow of a program. Handling them properly prevents crashes and improves robustness.

Common Exceptions in Python

Here are some frequently encountered exceptions:

Exception	Description 📖
`ZeroDivisionError`	Division by zero (🚫➗0)
`TypeError`	Invalid operation between incompatible types (×📏)
`ValueError`	Function receives an argument of the right type but invalid value (⚠️🎭)
`IndexError`	Trying to access an invalid list index (📜×)
`KeyError`	Accessing a missing dictionary key (🔑🚫)
`AttributeError`	Calling an invalid attribute on an object (🔄🚫)
`FileNotFoundError`	Trying to open a non-existent file (📂×)
`ImportError`	Module import failure (📦🔄×)

Handling Exceptions With `try-except`

Use `try-except` to catch errors and prevent program crashes.

```
try:
    x = 10 / 0  # 🚨 ZeroDivisionError
except ZeroDivisionError:
    print("You cannot divide by zero! 🚫")
```

Output:

```
You cannot divide by zero! 🚫
```

Catching Multiple Exceptions

```
try:
    num = int("Hello")  # 🚨 ValueError
except (ValueError, TypeError):
    print("Invalid input! ⚠️")
```

Output:

```
Invalid input! ⚠️
```

Using `else` and `finally`

- `else` runs if no exception occurs ✅
- `finally` runs **always** (cleanup code)

```
try:
    file = open("data.txt", "r")
    content = file.read()
except FileNotFoundError:
    print("File not found! 📂❌")
else:
    print("File read successfully! ✅")
finally:
    print("Execution complete. 🔄")
```

Output (if file is missing):

```

```

```
File not found! 📂❌
Execution complete. 🔄
```

Raising Exceptions (`raise`)

Use `raise` to create custom exceptions.

```python
def check_age(age):
    if age < 18:
        raise ValueError("Age must be 18 or older! 🚨")
    return "Access granted ✅"

try:
    print(check_age(15))
except ValueError as e:
    print(f"Error: {e}")
```

Output:

```
Error: Age must be 18 or older! 🚨
```

Custom Exceptions

Define custom exceptions by extending `Exception`.

```python
class CustomError(Exception):
    """Custom exception example."""
    pass

try:
    raise CustomError("Something went wrong! ✖")
except CustomError as e:
    print(e)
```

Output:

```
Something went wrong! ✖
```

Best Practices for Exception Handling

✔ Catch **specific** exceptions instead of using a generic `except:` block 🎯
✔ Use **logging** instead of `print()` for production-ready error tracking 📝
✔ Keep `try-except` blocks minimal to avoid hiding real errors 🛠
✔ Avoid `bare except:` unless re-raising the exception 🚨

Key Takeaways

✔ **Exceptions** help prevent program crashes
✔ Use `try-except` to handle **anticipated errors**
✔ Always clean up with `finally` 🔄
✔ Raise **custom exceptions** for better debugging