📆 DAY 05 — Error Handling in Python

Goal : Write safe, crash-resistant code

1️⃣ What Is an Error / Exception?

An exception is a runtime problem.

Example:

```
x = 10 / 0

# output
ZeroDivisionError
```

without handling → program crashes.

2️⃣ Basic `try` / `except`

✅ Simple pattern

```
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

- Program does not crash
- Error handled cleanly

3️⃣ Catching Any Error (Be Careful)

```
try:
    x = int("abc")
except Exception as e:
    print("Error:", e)
```

⚠️ Use this only when you really must, otherwise catch specific errors.

4️⃣ Multiple `except` Blocks

```
try:
    x = int("abc")
    y = 10 / 0
except ValueError:
    print("Invalid number")
except ZeroDivisionError:
    print("Division by zero")
```

Python checks top to bottom.

5️⃣ `else` and `finally` (Very Useful)

`else` → runs if no error

`finally` → always runs

```
try:
    x = int("5")
except ValueError:
    print("Conversion failed")
else:
    print("Conversion successful:", x)
finally:
    print("Done")
```

6️⃣ Raising Your Own Errors (Important)

❌ Bad

```
if x < 0:
    print("Invalid")
```

✅ Good

```
if x < 0:
```

```
    raise ValueError("x must be non-negative")
```

Raising errors is **Professional code behavior**.

7️⃣ Custom Exceptions (ML-Friendly)

Step 1: Define exception

```
class DataValidationError(Exception):
    pass
```

Step 2: Use it

```
def train_model(X, y):
    if X is None or y is None:
        raise DataValidationError("Training data cannot be None")
```

- Clear error
- Easy debugging
- Clean ML pipelines

8️⃣ Real ML Engineer Example

```
def load_data(path):
    try:
        with open(path) as f:
            return f.read()
    except FileNotoundError:
        raise FileNotFoundError(f"Dataset not found at {path}")
```

This is **real production-style code.**

9️⃣ What NOT to Do ❌

❌ Silent failures:

```
try:
    x = 10 / 0
```

```
except:
    pass  # very bad
```

❌ Printing instead of raising:

```
print("Error occurred")  # useless for debugging
```