

## Day 24



### Raising custom errors in Python:

#### What Is a Custom Error?

A custom error (custom exception) is a user-defined error that you create to represent specific problems in your program.

#### Why Raise Custom Errors?

- To make errors more meaningful
- To improve code readability
- To handle application-specific rules
- To separate logic errors from system errors

#### Using raise (Basics)

You can raise a built-in exception manually.

```
age = -5
if age < 0:
    raise ValueError("Age cannot be negative")
```

Example: a basic example.

```
1 age = -5
2 if age < 0:
3     raise ValueError("Age cannot be negative")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
⊙ PS E:\Python\Day21-30\Day24> python main.py
Traceback (most recent call last):
  File "E:\Python\Day21-30\Day24\main.py", line 3, in <module>
    raise ValueError("Age cannot be negative")
ValueError: Age cannot be negative
```

Example: Handling Custom Exceptions with try-except

```
10 class InvalidAgeError(Exception):
11     pass
12 try:
13     age = -10
14     if age < 0:
15         raise InvalidAgeError("Invalid age provided")
16 except InvalidAgeError as e:
17     print("Custom Error:", e)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\Python\Day21-30\Day24> python main.py
Custom Error: Invalid age provided
```

Summary:

- Custom errors are user-defined exceptions created for specific problems
- They are created by inheriting from Exception
- Use raise to trigger a custom error
- Custom errors make code clearer and more meaningful
- They must be defined before use
- Use try-except to handle custom errors gracefully
- Unhandled custom errors show a traceback (expected behavior)
- Name custom errors clearly (e.g., InvalidAgeError)

--The End--