CUSTOM ERRORS

Instead of creating new error types, just use **Python's built-in errors** with clear messages.

Invalid Age (Using ValueError)

```
def check_age(age):
    if age < 18 or age > 60:
        raise ValueError("Age must be between 18 and 60.")

check_age(15)  # ❌ Error
```

Wrong Type (Using TypeError)

```
def divide(a, b):
    if type(a) not in [int, float] or type(b) not in [int, float]:
        raise TypeError("Both values must be numbers.")
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b

divide(10, "2")  # ❌ Error
```

Missing Key (Using KeyError)

```
prices = {"apple": 100, "banana": 50}

def get_price(item):
    if item not in prices:
        raise KeyError("Item not found in price list.")
    return prices[item]

get_price("mango")  # ❌ Error
```

Out of Range Index (Using IndexError)

```
my_list = [1, 2, 3]

def get_item(index):
    if index >= len(my_list) or index < 0:
        raise IndexError("Index out of range.")
    return my_list[index]

get_item(5)  # ❌ Error
```

Empty Data (Using RuntimeError)

```
def process_data(data):
    if not data:
        raise RuntimeError("No data provided.")
    return "Processing done!"

process_data([])  # ❌ Error
```

Summary (Easy Rules)

✓ **ValueError** → Wrong number
✓ **TypeError** → Wrong type
✓ **KeyError** → Missing dictionary key
✓ **IndexError** → Wrong list index
✓ **RuntimeError** → Custom error