

# **PYTHON WORKBOOK – SECTION 7**

## ***Error & Exception Handling***

Programmer's Hub – by CodeWithVivek  
<https://www.youtube.com/@code-with-vivek>

## 7.1 — Introduction

### Section Objective

By the end of this section, learners will:

- Understand runtime errors and exceptions
- Handle errors gracefully using try, except, else, finally
- Raise custom exceptions
- Write robust, production-ready Python code

### Quick Concept

An **error** occurs when Python cannot execute code.

An **exception** is a runtime error that can be handled.

Common exceptions:

- ZeroDivisionError
  - ValueError
  - TypeError
  - IndexError
  - KeyError
- 

### Try This

Run the code below and observe the output:

```
x = 10
```

```
y = 0
```

```
print(x / y)
```

Write the error you see:

---

### Key Insight

Python stops execution when an exception occurs — unless it is handled.

## 7.2 — The try...except block

### Syntax

try:

    # code which may cause error

except:

    # error handling code

---

### Try This

try:

    num = int(input("Enter a number: "))

    print(10 / num)

except:

    print("Something went wrong!")

What happens if user enters 0?

---

### Debug This

The following code crashes. Fix it.

num = int(input("Enter a number: "))

print(10 / num)

# Write corrected code:

## Handling Specific Exceptions

### Example

try:

```
x = int("abc")
```

except ValueError:

```
print("Invalid number")
```

---

### Try This

Modify the program to handle **ZeroDivisionError** separately.

# Write here

---

### Best Practice

Always catch **specific exceptions** instead of using bare except.

---

### else & finally

#### Example

try:

```
num = int(input("Enter number: "))
```

```
print(10 / num)
```

except Exception as e:

```
print("Error:", e)
```

else:

```
print("Execution successful")
```

finally:

```
print("Program ended")
```

**Try This**

Identify when else executes:

---

**Debug This**

Why does this code fail?

try:

```
    print(10 / 0)
```

finally:

```
    print("Done")
```

# Explain here:

Code with Vivek

### 7.3 — Raising Exceptions

#### Example

```
age = int(input("Enter age: "))  
  
if age < 18:  
    raise ValueError("Age must be 18 or above")
```

---

#### Try This

Add exception handling to the above code.

# Write here

Code with Vivek

## 7.4 — Custom Exceptions

### Example

```
class AgeTooSmall(Exception):  
    pass
```

---

### Your Turn

Create a custom exception for **negative balance**.

# Write here

Code with Vivek

**Practice Problems****Problem 1**

Write a program that: Takes two numbers, handles division errors and prints result safely.

# Write here

---

**Problem 2**

Write a program that: Takes list index from user and handles invalid index

Code with Vivek

**MINI ASSIGNMENT - SAFE CALCULATOR****Objective**

Create a calculator that:

- Handles invalid input
- Handles division by zero
- Never crashes

**Features**

- Menu-driven
- Uses try-except
- Displays friendly error messages

Code with Vivek