

## Understanding try, except, and finally Blocks

### Definition

- **Error handling** allows a program to continue running even if an error occurs.
  - Python uses:
    - try → code that may cause an error
    - except → code that runs if an error occurs
    - finally → code that runs **always**, whether an error occurs or not
- 

### Basic Syntax

```
try:  
    # risky code  
except ErrorType:  
    # runs if error occurs  
finally:  
    # always runs
```

---

### Simple Example

```
try:  
    x = int("abc") # invalid conversion  
except ValueError:  
    print("Conversion failed")  
finally:  
    print("Program ended")
```

### Output

```
Conversion failed  
Program ended
```

---

### Practical Code Example

```
def divide(a, b):  
    try:  
        result = a / b  
        print("Result:", result)  
    except ZeroDivisionError:  
        print("Cannot divide by zero")  
    finally:  
        print("Division operation completed")  
  
divide(10, 2)  
divide(10, 0)
```

---

### Handling Common Errors

## **Definition**

Common errors are mistakes that occur during program execution.

Python raises **exceptions** for these errors.

---

## **Common Python Errors & Examples**

---

### **1. ZeroDivisionError**

Occurs when dividing by zero.

try:

```
    print(10 / 0)
except ZeroDivisionError:
    print("Division by zero is not allowed")
```

---

### **2. ValueError**

Occurs when incorrect data type is used.

try:

```
    age = int("twenty")
except ValueError:
    print("Invalid number format")
```

---

### **3. TypeError**

Occurs when incompatible data types are used.

try:

```
    print(10 + "5")
except TypeError:
    print("Cannot add integer and string")
```

---

### **4. IndexError**

Occurs when accessing an invalid list index.

try:

```
    nums = [1, 2, 3]
    print(nums[5])
except IndexError:
    print("Index out of range")
```

---

### **5. KeyError**

Occurs when accessing a missing dictionary key.

try:

```
    data = {"name": "Athar"}
    print(data["age"])
except KeyError:
```

```
print("Key not found in dictionary")
```

---

### Multiple Exceptions Example

```
try:  
    a = int(input("Enter number: "))  
    b = int(input("Enter divisor: "))  
    print(a / b)  
except ValueError:  
    print("Please enter valid numbers")  
except ZeroDivisionError:  
    print("Cannot divide by zero")
```

---

### Activity: Calculator Handling Division by Zero (Practical Program)

#### Definition

A **graceful program** handles errors without crashing and gives meaningful messages to users.

---

#### Practical Calculator Code

```
def calculator():  
    try:  
        num1 = float(input("Enter first number: "))  
        num2 = float(input("Enter second number: "))  
  
        result = num1 / num2  
        print("Result:", result)  
  
    except ZeroDivisionError:  
        print("Error: Division by zero is not allowed")  
  
    except ValueError:  
        print("Error: Please enter numeric values only")  
  
    finally:  
        print("Calculator operation finished")  
  
calculator()
```

---

#### Sample Runs

##### Input

```
Enter first number: 10  
Enter second number: 0
```

## **Output**

Error: Division by zero is not allowed

Calculator operation finished

---

## **Summary**

<b>Keyword</b>	<b>Purpose</b>
try	Write risky code
except	Handle specific errors
finally	Always executes
Exception	Runtime error
Graceful handling	Program doesn't crash

---