Error Handling Basics

Overview

Error handling allows a program to **respond to unexpected conditions** (errors and exceptions) **without crashing**.

Goals:
- Gracefully recover or clean up when something goes wrong
- Provide informative messages to users or logs
- Maintain program flow where possible

Causes of Errors

- **Syntax Errors:** Missing colons, parentheses, or incorrect indentation.
- **Incorrect Indentation:** Python relies on indentation; a misplaced indent causes errors or changes logic.
- **Using Undefined Variables:** Forgetting to define variables before using them.
- **Type Mismatch:** Mixing incompatible types, like adding a string and an integer ("5" + 5).
- **Misusing Functions:** Calling functions incorrectly or not returning values when needed.
- **Case Sensitivity Mistakes:** Python is case-sensitive: Print() and print() are different.

Types of Errors

Category	Description	Examples
Syntax Errors	Mistakes in the structure of code	Missing colon
Runtime Errors	Exceptions raised while the program is running	Division by zero, accessing non-existent key in dict
Logical Errors	Code runs without crashing but yields incorrect results	Using and instead of or in condition

Handling errors: try variants

Techniques	Purpose
try – except	Catch and handle one or more specific exceptions.
try – except … except	Handle different exception types separately
try – except … else	Run code only if no exception occurred.
try – finally	Guarantee cleanup code runs, regardless of exceptions.
try – except – finally	Handle errors and always run cleanup.

Handling errors: try-except

Purpose: Catch and handle a specific type of error.

syntax

```
try:

    # some risky code

except SomeException as e:

    # handle the error
```

Handling errors: try-except

python

```python
try:
    age = int(input("Enter your age: "))
    print(f"In 5 years, you'll be {age + 5}.")

except ValueError as e:
    print("Error: Please enter a valid number.")
```

Output

```
Enter your age: five
ERROR!
Error: Please enter a valid number.
```

Handling errors: try-except…except

Purpose: Handle different exception types separately.

syntax

```
try:
    # some risky code

except FirstException:
    # handle first error

except SecondException:
    # handle second error
```

Handling errors: try-except…except

python

```python
try:
    items = ["apple", "banana"]
    index = int(input("Enter index to fetch fruit: "))
    print(f"You selected: {items[index]}")

except ValueError:
    print("Error: Index must be a number.")

except IndexError:
    print("Error: Index out of range.")
```

Output

```
Enter your age: five
ERROR!
Error: Please enter a valid number.
```

Handling errors: try-except…else

Purpose: Run code only if no exception occurred.

syntax

```
try:
    # some risky code

except Exception:
    # handle error here

else:
    # execute ONLY IF no exception occurs
```

Handling errors: try-except…else

python

```python
try:
    price = float(input("Enter item price: "))
    quantity = int(input("Enter quantity: "))
    total = price * quantity
except ValueError:
    print("Error: Please enter valid numbers.")
else:
    print(f"Total cost: ₹{total}")
```

Output

```
Enter item price: 5
Enter quantity: 4
Total cost: ₹20.0
```

Handling errors: try-finally

Purpose: Always run final cleanup code, regardless of whether an exception occurred or not.

syntax

```
try:
    # some risky code
finally:
    # code here ALWAYS runs, exception or not
```

Handling errors: try-finally

python

```python
try:
    result = 10 / 0
    print(“This won’t print")
finally:
    print(“End of operation.")
```

Output

```
End of operation.
ZeroDivisionError: division by zero
```

Handling errors: try-except-finally

Purpose: Handle errors and ensure cleanup code runs regardless of what happens

syntax

```
try:
    # some risky code
except Exception:
    # handle error here
finally:
    # code here ALWAYS runs, exception or not
```

Handling errors: try-except-finally

python

```python
try:
    num = int(input("Enter a number: "))
    result = 10 / num
    print(f"Result: {result}")
except ZeroDivisionError:
    print("You can't divide by zero.")
except ValueError:
    print("Invalid input. Please enter a number.")
finally:
    print(“Ending.")
```

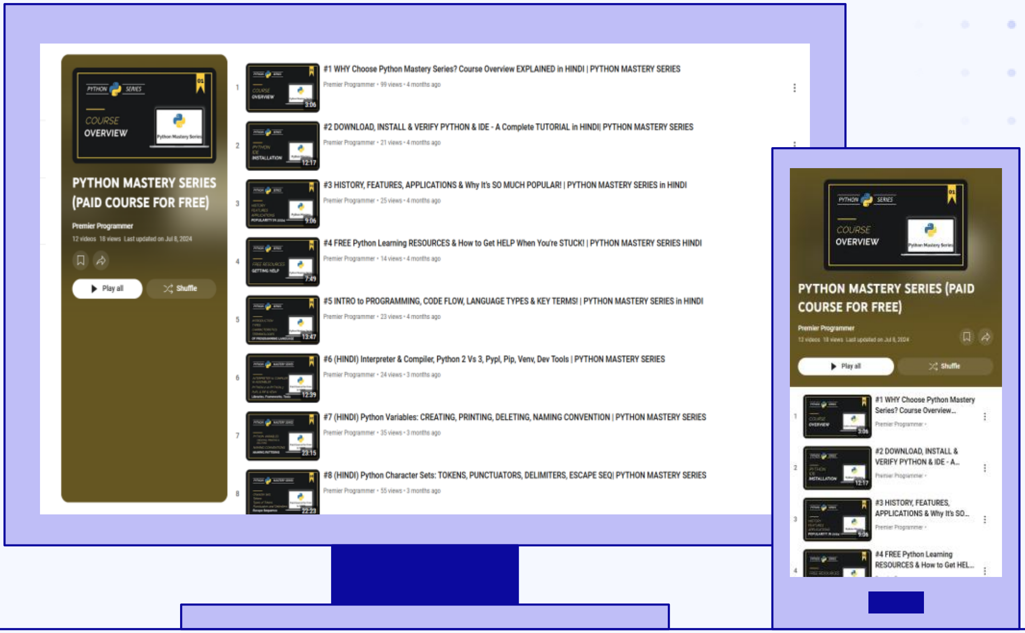
Output

```
Enter a number: 5
Result: 2.0
Ending.
```

```
Enter a number: 0
You can't divide by zero.
Ending.
```

WATCH

Level up your coding with each episode in this focused Python series.



PYTHON MASTERY SERIES (PAID COURSE FOR FREE)

Premier Programmer

12 videos 18 views Last updated on Jul 8, 2024

1 #1 WHY Choose Python Mastery Series? Course Overview EXPLAINED in HINDI | PYTHON MASTERY SERIES
Premier Programmer • 99 views • 4 months ago

2 #2 DOWNLOAD, INSTALL & VERIFY PYTHON & IDE - A Complete TUTORIAL in HINDI| PYTHON MASTERY SERIES
Premier Programmer • 21 views • 4 months ago

3 #3 HISTORY, FEATURES, APPLICATIONS & Why It's SO MUCH POPULAR! | PYTHON MASTERY SERIES in HINDI
Premier Programmer • 25 views • 4 months ago

4 #4 FREE Python Learning RESOURCES & How to Get HELP When You're STUCK! | PYTHON MASTERY SERIES HINDI
Premier Programmer • 14 views • 4 months ago

5 #5 INTRO to PROGRAMMING, CODE FLOW, LANGUAGE TYPES & KEY TERMS! | PYTHON MASTERY SERIES in HINDI
Premier Programmer • 23 views • 4 months ago

6 #6 (HINDI) Interpreter & Compiler, Python 2 Vs 3, Pypl, Pip, Venv, Dev Tools | PYTHON MASTERY SERIES
Premier Programmer • 24 views • 3 months ago

7 #7 (HINDI) Python Variables: CREATING, PRINTING, DELETING, NAMING CONVENTION | PYTHON MASTERY SERIES
Premier Programmer • 35 views • 3 months ago

8 #8 (HINDI) Python Character Sets: TOKENS, PUNCTUATORS, DELIMITERS, ESCAPE SEQ| PYTHON MASTERY SERIES
Premier Programmer • 55 views • 3 months ago

PYTHON MASTERY SERIES (PAID COURSE FOR FREE)
Premier Programmer
12 videos 18 views Last updated on Jul 8, 2024
1 #1 WHY Choose Python Mastery Series? Course Overview...
Premier Programmer •
2 #2 DOWNLOAD, INSTALL & VERIFY PYTHON & IDE - A...
Premier Programmer •
3 #3 HISTORY, FEATURES, APPLICATIONS & Why It's SO...
Premier Programmer •
4 #4 FREE Python Learning RESOURCES & How to Get HEL...

Next Video!

Raising Exceptions
Assertions