Tops Technology

Module 15) Advance Python Programming

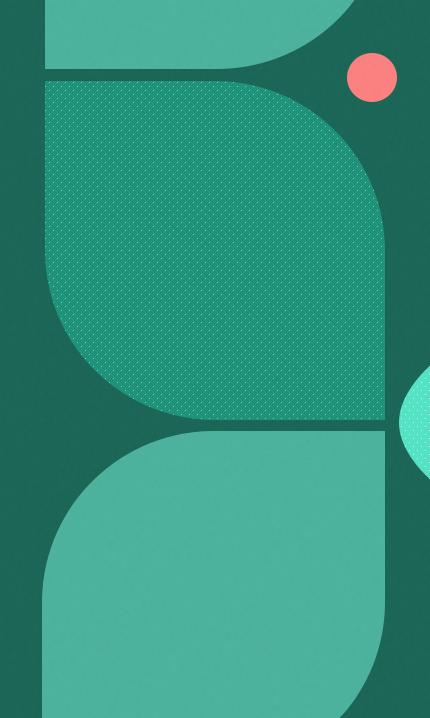
Presented By:

Nandni Vala

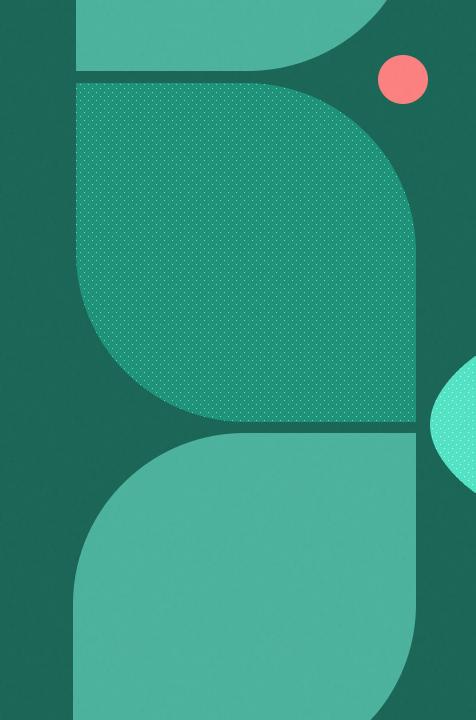
Exception Handling

1.Introduction to exceptions and how to handle them using try, except, and finally.

- An **exception** is an error that occurs during the execution of a program, disrupting its normal flow.
- > Examples of exceptions in Python:
- **ZeroDivisionError**: Dividing by zero.
- ➤ FileNotFoundError: Accessing a file that doesn't exist.
- ➤ ValueError: Passing an invalid value to a function.
- ➤ Python provides a mechanism to handle exceptions using the try, except, and finally blocks. This prevents the program from crashing and allows developers to gracefully handle errors.
- ➤ Syntax of Exception Handlingtry:
- # Code that might raise an exception
- > except SomeException:
- # Code to handle the exception
- > finally:
- # Code that runs no matter what (optional)



- > try Block: Code that might raise an exception.
- > except Block: Handles specific exceptions.
- ➤ finally Block: Executes cleanup code regardless of an exception.
- **Example**
- > try:
- \triangleright result = 10 / 0
- ➤ except ZeroDivisionError:
- print("Cannot divide by zero.")
- > finally:
- print("Execution complete.")
- **Output:**
- Cannot divide by zero.
- > Execution complete.



2. Understanding multiple exceptions and custom exceptions

- > Multiple Exceptions
- When a program can raise different types of exceptions, you can handle them using multiple except blocks or a single block with a tuple.
- **Example:**
- > try:
- value = int("abc") # This will raise a ValueError
- result = 10 / 0 # This will raise a ZeroDivisionError
- > except ValueError:
- print("Invalid input: Cannot convert to an integer.")
- > except ZeroDivisionError:
- print("Error: Division by zero.")
- > Output:
- Invalid input: Cannot convert to an integer.

- > Custom Exceptions
- ➤ You can define your own exceptions by subclassing the built-in Exception class. This is useful when the built-in exceptions don't suit your needs.
- **▶** Defining a Custom Exception
- ➤ class CustomError(Exception):
- """Custom exception class."""
- def __init__(self, message):
- > self.message = message
- super().__init__(self.message)
- **Exmple:**
- > def divide(a, b):
- \rightarrow if b == 0:
- raise CustomError("Cannot divide by zero.")
- return a / b
- 🗲 try:
- \triangleright result = divide(10, 0)
- > except CustomError as e:
- print(f"Custom Error: {e}")