1. **Error Handling**:
   * Python uses try, except, else, and finally blocks to handle exceptions.
   * Syntax:

python

Copy code

try:

# Code that may raise an exception

except ExceptionType as e:

# Code to handle the exception

else:

# Code to run if no exception occurs

finally:

# Code to run regardless of exceptions

1. **Common Exceptions**:
   * FileNotFoundError
   * KeyError
   * ValueError
   * IndexError
2. **Logging**:
   * Use Python’s logging module to track events in your applications.
   * Logging levels:
     + DEBUG: Detailed information for diagnosing problems.
     + INFO: Confirmation that things are working as expected.
     + WARNING: Indication that something unexpected happened.
     + ERROR: A serious problem preventing some functionality.
     + CRITICAL: A severe error causing the program to crash.
3. **Log File Management**:
   * Automatically save logs to files.
   * Rotate logs to prevent large file sizes using logging.handlers.

**Practical Exercises**

**1. Basic Error Handling**

* Script:

python

Copy code

try:

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

print(f"The number you entered is {num}.")

except ValueError as e:

print(f"Invalid input. Error: {e}")

* Test with both valid and invalid inputs (e.g., "10", "abc").

**2. Handle FileNotFoundError**

* Objective: Check if a file exists before reading it.
* Script:

python

Copy code

try:

with open("nonexistent\_file.txt", "r") as file:

print(file.read())

except FileNotFoundError:

print("The file does not exist. Please check the file name.")

* Output:

sql

Copy code

The file does not exist. Please check the file name.

**3. Use else and finally**

* Script:

python

Copy code

try:

with open("servers.txt", "r") as file:

print("File content:\n", file.read())

except FileNotFoundError:

print("File not found.")

else:

print("File read successfully.")

finally:

print("End of file operation.")

**4. Set Up Basic Logging**

* Script:

python

Copy code

import logging

logging.basicConfig(level=logging.INFO)

logging.info("This is an INFO message.")

logging.warning("This is a WARNING message.")

logging.error("This is an ERROR message.")

* Output:

vbnet

Copy code

INFO:root:This is an INFO message.

WARNING:root:This is a WARNING message.

ERROR:root:This is an ERROR message.

**5. Save Logs to a File**

* Objective: Redirect logs to a file.
* Script:

python

Copy code

import logging

logging.basicConfig(

filename="app.log",

level=logging.DEBUG,

format="%(asctime)s - %(levelname)s - %(message)s"

)

logging.debug("Debugging application.")

logging.info("Application started.")

logging.warning("Low disk space.")

logging.error("Unable to connect to the database.")

logging.critical("Application crashed!")

* Check app.log for entries.

**6. Rotate Logs Using logging.handlers**

* Objective: Automatically manage log file size.
* Script:

python

Copy code

import logging

from logging.handlers import RotatingFileHandler

handler = RotatingFileHandler("rotating\_app.log", maxBytes=200, backupCount=3)

logging.basicConfig(

level=logging.INFO,

format="%(asctime)s - %(levelname)s - %(message)s",

handlers=[handler]

)

for i in range(100):

logging.info(f"Log message {i}")

* Output:
  + Logs will be rotated when the file reaches 200 bytes, with backups stored as rotating\_app.log.1, rotating\_app.log.2, etc.

**7. Challenge: Monitor a File for Errors and Log Them**

* Objective: Read a file and log errors into another file.
* Script:

python

Copy code

import logging

logging.basicConfig(

filename="error\_monitor.log",

level=logging.ERROR,

format="%(asctime)s - %(levelname)s - %(message)s"

)

try:

with open("app.log", "r") as log\_file:

for line in log\_file:

if "ERROR" in line or "CRITICAL" in line:

logging.error(f"Error detected: {line.strip()}")

except FileNotFoundError:

logging.error("The log file does not exist.")

* Output:
  + Errors and critical logs from app.log will be saved to error\_monitor.log.

**8. Bonus: Logging with Custom Functions**

* Objective: Create reusable logging functionality.
* Script:

python

Copy code

import logging

def setup\_logger(name, log\_file, level=logging.INFO):

logger = logging.getLogger(name)

formatter = logging.Formatter("%(asctime)s - %(levelname)s - %(message)s")

file\_handler = logging.FileHandler(log\_file)

file\_handler.setFormatter(formatter)

logger.setLevel(level)

logger.addHandler(file\_handler)

return logger

# Create loggers

app\_logger = setup\_logger("app\_logger", "app.log")

error\_logger = setup\_logger("error\_logger", "errors.log", level=logging.ERROR)

# Use loggers

app\_logger.info("Application initialized.")

app\_logger.warning("Low memory warning.")

error\_logger.error("Database connection failed.")

* Check app.log and errors.log for separate logging outputs.