**1. What is the difference between interpreted and compiled languages?**

**Answer:**  
Interpreted languages execute code **line by line** at runtime (e.g., Python, JavaScript), whereas compiled languages convert the entire code into **machine code** before execution (e.g., C, Java). Interpreted languages are generally **easier to debug**, but compiled languages **run faster**.

**2. What is exception handling in Python?**

**Answer:**  
Exception handling in Python is a mechanism to **handle runtime errors** using try, except, finally, and else blocks. It prevents abrupt program termination and allows developers to manage errors gracefully.

**3. What is the purpose of the finally block in exception handling?**

**Answer:**  
The finally block in Python executes **regardless of whether an exception occurs** or not. It is commonly used for **cleanup operations** like closing files or releasing resources.

**Example:**

python

CopyEdit

try:

file = open("data.txt", "r")

content = file.read()

except FileNotFoundError:

print("File not found!")

finally:

file.close() # Ensures the file is closed even if an error occurs

**4. What is logging in Python?**

**Answer:**  
Logging in Python is a built-in module used to **record runtime events**, such as errors and debugging messages. It helps in **tracking issues** and improving maintainability.

**Example:**

python

CopyEdit

import logging

logging.basicConfig(level=logging.INFO)

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

**5. What is the significance of the \_\_del\_\_ method in Python?**

**Answer:**  
The \_\_del\_\_ method is a destructor method that **automatically executes when an object is deleted**. It is used for cleanup operations like closing connections.

**Example:**

python

CopyEdit

class Demo:

def \_\_del\_\_(self):

print("Object deleted!")

obj = Demo()

del obj # This calls \_\_del\_\_()

**6. What is the difference between import and from ... import in Python?**

**Answer:**

* import module\_name → Imports the entire module.
* from module\_name import function\_name → Imports only a specific function.

**Example:**

python

CopyEdit

import math

print(math.sqrt(16)) # Using the module name

from math import sqrt

print(sqrt(16)) # Direct access

**7. How can you handle multiple exceptions in Python?**

**Answer:**  
You can handle multiple exceptions using **multiple except blocks** or a **tuple** of exceptions.

**Example:**

python

CopyEdit

try:

x = int("abc")

except (ValueError, TypeError) as e:

print("Error occurred:", e)

**8. What is the purpose of the with statement when handling files in Python?**

**Answer:**  
The with statement **automatically closes** the file after execution, reducing the risk of resource leaks.

**Example:**

python

CopyEdit

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

content = file.read()

# No need to call file.close()

**9. What is the difference between multithreading and multiprocessing?**

**Answer:**

* **Multithreading** → Multiple threads share the same memory space (faster but has GIL issues).
* **Multiprocessing** → Creates separate memory spaces for each process (better for CPU-intensive tasks).

**Example:**

python

CopyEdit

from multiprocessing import Process

def task():

print("Running in separate process")

p = Process(target=task)

p.start()

**10. What are the advantages of using logging in a program?**

**Answer:**

* Helps in **debugging** and **monitoring**
* Records **runtime issues**
* Allows **different logging levels** (INFO, ERROR, DEBUG)

**11. What is memory management in Python?**

**Answer:**  
Memory management in Python includes **garbage collection** and **dynamic memory allocation** to free unused objects automatically.

**12. What are the basic steps involved in exception handling in Python?**

**Answer:**

1. **Try** → Code that may raise an exception
2. **Except** → Handles the exception
3. **Else** → Runs if no exception occurs
4. **Finally** → Always executes

**13. Why is memory management important in Python?**

**Answer:**  
Efficient memory management prevents **memory leaks** and ensures **optimal performance**.

**14. What is the role of try and except in exception handling?**

**Answer:**  
The try block contains the code that may raise an exception, while the except block catches and handles the error.

**15. How does Python's garbage collection system work?**

**Answer:**  
Python’s **garbage collector** removes objects with **zero references** to free up memory. It uses **reference counting** and **cyclic garbage collection**.

**16. What is the purpose of the else block in exception handling?**

**Answer:**  
The else block **executes only if no exception occurs** in the try block.

**Example:**

python

CopyEdit

try:

print(10 / 2)

except ZeroDivisionError:

print("Cannot divide by zero")

else:

print("No errors!")

**17. What are the common logging levels in Python?**

**Answer:**

* **DEBUG** → Detailed information
* **INFO** → General runtime info
* **WARNING** → Indications of potential issues
* **ERROR** → Errors that prevent execution
* **CRITICAL** → Serious errors causing system failure

**18. What is the difference between os.fork() and multiprocessing in Python?**

**Answer:**

* os.fork() creates a **child process** (Unix only).
* multiprocessing works across all OS and allows better control over processes.

**19. What is the importance of closing a file in Python?**

**Answer:**  
Closing a file **frees up system resources** and prevents **data corruption**.

**20. What is the difference between file.read() and file.readline() in Python?**

**Answer:**

* file.read() → Reads the **entire file**
* file.readline() → Reads **one line at a time**

**21. What is the logging module in Python used for?**

**Answer:**  
The logging module helps in **tracking runtime events** and debugging.

**22. What is the os module in Python used for in file handling?**

**Answer:**  
The os module provides functions for **interacting with the operating system**, like file handling and directory management.

**23. What are the challenges associated with memory management in Python?**

**Answer:**

* **Circular references**
* **High memory usage for large objects**
* **Garbage collector overhead**

**24. How do you raise an exception manually in Python?**

**Answer:**  
Using the raise keyword.

**Example:**

python

CopyEdit

if age < 18:

raise ValueError("Age must be 18 or above")

**25. Why is it important to use multithreading in certain applications?**

**Answer:**  
Multithreading is useful for **I/O-bound tasks** like web scraping and file handling because it allows multiple operations to run **concurrently**.

**1. How can you open a file for writing in Python and write a string to it?**

**Answer:**  
You can use the open() function with the "w" mode.

**Example:**

python

CopyEdit

with open("output.txt", "w") as file:

file.write("Hello, this is a test file.")

This will create (or overwrite) output.txt and write the string to it.

**2. Write a Python program to read the contents of a file and print each line.**

**Answer:**  
You can use a with statement to open and read a file line by line.

**Example:**

python

CopyEdit

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

for line in file:

print(line.strip())

This reads and prints each line from input.txt.

**3. How would you handle a case where the file doesn't exist while trying to open it for reading?**

**Answer:**  
Use a try-except block to catch FileNotFoundError.

**Example:**

python

CopyEdit

try:

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

print(file.read())

except FileNotFoundError:

print("Error: File not found.")

**4. Write a Python script that reads from one file and writes its content to another file.**

**Answer:**

python

CopyEdit

with open("source.txt", "r") as source, open("destination.txt", "w") as destination:

for line in source:

destination.write(line)

This reads content from source.txt and writes it to destination.txt.

**5. How would you catch and handle division by zero error in Python?**

**Answer:**  
Use a try-except block to catch ZeroDivisionError.

**Example:**

python

CopyEdit

try:

result = 10 / 0

except ZeroDivisionError:

print("Error: Division by zero is not allowed.")

**6. Write a Python program that logs an error message to a log file when a division by zero exception occurs.**

**Answer:**

python

CopyEdit

import logging

logging.basicConfig(filename="errors.log", level=logging.ERROR)

try:

result = 10 / 0

except ZeroDivisionError as e:

logging.error("Division by zero occurred: %s", str(e))

**7. How do you log information at different levels (INFO, ERROR, WARNING) in Python using the logging module?**

**Answer:**

python

CopyEdit

import logging

logging.basicConfig(level=logging.DEBUG)

logging.debug("This is a debug message")

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

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

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

logging.critical("This is a critical message")

**8. Write a program to handle a file opening error using exception handling.**

**Answer:**

python

CopyEdit

try:

with open("non\_existent.txt", "r") as file:

data = file.read()

except FileNotFoundError:

print("File not found error occurred.")

**9. How can you read a file line by line and store its content in a list in Python?**

**Answer:**

python

CopyEdit

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

lines = file.readlines()

print(lines)

**10. How can you append data to an existing file in Python?**

**Answer:**  
Use "a" mode to append data.

**Example:**

python

CopyEdit

with open("output.txt", "a") as file:

file.write("\nAppending a new line.")

**11. Write a Python program that uses a try-except block to handle an error when attempting to access a dictionary key that doesn't exist.**

**Answer:**

python

CopyEdit

data = {"name": "Alice"}

try:

print(data["age"])

except KeyError:

print("Key not found in dictionary.")

**12. Write a program that demonstrates using multiple except blocks to handle different types of exceptions.**

**Answer:**

python

CopyEdit

try:

x = int("abc")

y = 10 / 0

except ValueError:

print("Invalid conversion to integer.")

except ZeroDivisionError:

print("Cannot divide by zero.")

**13. How would you check if a file exists before attempting to read it in Python?**

**Answer:**  
Use the os.path.exists() function.

python

CopyEdit

import os

if os.path.exists("file.txt"):

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

print(file.read())

else:

print("File does not exist.")

**14. Write a program that uses the logging module to log both informational and error messages.**

**Answer:**

python

CopyEdit

import logging

logging.basicConfig(filename="app.log", level=logging.INFO)

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

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

**15. Write a Python program that prints the content of a file and handles the case when the file is empty.**

**Answer:**

python

CopyEdit

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

content = file.read()

if not content:

print("The file is empty.")

else:

print(content)

**16. Demonstrate how to use memory profiling to check the memory usage of a small program.**

**Answer:**

python

CopyEdit

from memory\_profiler import profile

@profile

def test\_function():

x = [i for i in range(100000)]

return x

test\_function()

**17. Write a Python program to create and write a list of numbers to a file, one number per line.**

**Answer:**

python

CopyEdit

numbers = [1, 2, 3, 4, 5]

with open("numbers.txt", "w") as file:

for num in numbers:

file.write(f"{num}\n")

**18. How would you implement a basic logging setup that logs to a file with rotation after 1MB?**

**Answer:**

python

CopyEdit

import logging

from logging.handlers import RotatingFileHandler

handler = RotatingFileHandler("app.log", maxBytes=1048576, backupCount=3)

logging.basicConfig(handlers=[handler], level=logging.INFO)

logging.info("This is a log message.")

**19. Write a program that handles both IndexError and KeyError using a try-except block.**

**Answer:**

python

CopyEdit

try:

lst = [1, 2, 3]

print(lst[5])

d = {"a": 1}

print(d["b"])

except IndexError:

print("Index out of range.")

except KeyError:

print("Key not found in dictionary.")

**20. How would you open a file and read its contents using a context manager in Python?**

**Answer:**

python

CopyEdit

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

content = file.read()

print(content)

**21. Write a Python program that reads a file and prints the number of occurrences of a specific word.**

**Answer:**

python

CopyEdit

word\_to\_find = "Python"

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

content = file.read()

print(f"The word '{word\_to\_find}' appears {content.count(word\_to\_find)} times.")

**22. How can you check if a file is empty before attempting to read its contents?**

**Answer:**

python

CopyEdit

import os

if os.stat("file.txt").st\_size == 0:

print("File is empty.")

else:

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

print(file.read())