File and Exception Handling

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

A: Interpreted languages execute code line by line at runtime (e.g., Python), while compiled languages convert the entire code into machine language before execution (e.g., C++).

1. Q: What is exception handling in Python?

A: Exception handling in Python is the process of responding to the occurrence of exceptions – anomalous or exceptional conditions requiring special processing.

1. Q: What is the purpose of the finally block in exception handling?

A: The finally block is used to execute code after the try and except blocks, regardless of whether an exception was raised or not.

1. Q: What is logging in Python?

A: Logging is a means of tracking events that happen when some software runs. Python's logging module provides a flexible framework for emitting log messages from Python programs.

1. Q: What is the significance of the \_\_del\_\_ method in Python?

A: The \_\_del\_\_ method is a destructor method which is called when an object is about to be destroyed.

1. Q: What is the difference between import and from ... import in Python?

A: 'import module' imports the entire module, whereas 'from module import function' imports specific functions or classes from a module.

1. Q: How can you handle multiple exceptions in Python?

A: You can handle multiple exceptions by using a tuple in a single except block or writing multiple except blocks for different exceptions.

1. Q: What is the purpose of the with statement when handling files in Python?

A: The with statement simplifies exception handling by automatically closing the file once it is no longer needed.

1. Q: What is the difference between multithreading and multiprocessing?

A: Multithreading allows concurrent execution in a single process, whereas multiprocessing uses multiple processes to run code in parallel.

1. Q: What are the advantages of using logging in a program?

A: Logging provides a way to track events, debug applications, and understand the program's behavior without interrupting the flow with print statements.

1. Q: What is memory management in Python?

A: Memory management in Python involves allocating and deallocating memory dynamically using a built-in garbage collector.

1. Q: What are the basic steps involved in exception handling in Python?

A: The basic steps include using try, except, else, and finally blocks to catch and handle exceptions gracefully.

1. Q: Why is memory management important in Python?

A: Efficient memory management ensures optimal performance, prevents memory leaks, and maintains program stability.

1. Q: What is the role of try and except in exception handling?

A: The try block allows testing of code for errors, while the except block handles the error if it occurs.

1. Q: How does Python's garbage collection system work?

A: Python uses reference counting and a cyclic garbage collector to detect and collect unused objects in memory.

1. Q: What is the purpose of the else block in exception handling?

A: The else block runs if no exceptions are raised in the try block, useful for code that should run only if the try is successful.

1. Q: What are the common logging levels in Python?

A: The common levels are DEBUG, INFO, WARNING, ERROR, and CRITICAL.

1. Q: What is the difference between os.fork() and multiprocessing in Python?

A: os.fork() creates a new child process in Unix-based systems, while multiprocessing is a platform-independent module for creating processes.

1. Q: What is the importance of closing a file in Python?

A: Closing a file ensures that all data is properly written and resources are freed.

1. Q: What is the difference between file.read() and file.readline() in Python?

A: file.read() reads the entire file as a string, while file.readline() reads one line at a time.

1. Q: What is the logging module in Python used for?

A: The logging module is used to log messages for tracking and debugging code execution.

1. Q: What is the os module in Python used for in file handling?

A: The os module provides functions to interact with the operating system, such as file and directory operations.

1. Q: What are the challenges associated with memory management in Python?

A: Challenges include circular references and non-deterministic finalization of objects.

1. Q: How do you raise an exception manually in Python?

A: Use the raise keyword followed by an exception type, e.g., raise ValueError('Invalid input').

1. Q: Why is it important to use multithreading in certain applications?

A: Multithreading is important for applications requiring concurrent operations without blocking, such as UI responsiveness or I/O operations.