Q: What is the difference between interpreted and compiled languages?
A: Interpreted languages execute code line by line using an interpreter, making debugging easier but execution slower (e.g., Python). Compiled languages are converted into machine code before execution, making them faster but less flexible (e.g., C, Java).

Q: What is exception handling in Python?
A: Exception handling allows a program to manage runtime errors using try, except, else, and finally blocks, preventing crashes.

Q: What is the purpose of the finally block in exception handling?
A: The finally block always executes whether an exception occurs or not. It is commonly used for cleanup operations like closing files or releasing resources.

Q: What is logging in Python?
A: Logging is the process of recording events, errors, and information during program execution using the logging module.

Q: What is the significance of the __del__ method in Python?
A: The __del__ method is a destructor method that is called when an object is about to be destroyed. It is often used for cleanup operations.

Q: What is the difference between import and from ... import in Python?
A: import module loads the entire module, requiring you to prefix functions with the module name. from module import function loads only specific items directly into the namespace.

Q: How can you handle multiple exceptions in Python?
A: You can handle multiple exceptions by writing multiple except blocks or by grouping exceptions in a tuple.

Q: What is the purpose of the with statement when handling files in Python?
A: The with statement ensures proper resource management by automatically closing the file after operations are done, even if an error occurs.

Q: What is the difference between multithreading and multiprocessing?
A: Multithreading allows multiple threads to run in the same process and share memory, useful for I/O-bound tasks. Multiprocessing uses multiple processes with separate memory space, better for CPU-bound tasks.

Q: What are the advantages of using logging in a program?
A: Logging provides better error tracking, debugging support, persistent records, and insights into program behavior.

Q: What is memory management in Python?
A: Memory management in Python involves allocating, using, and freeing memory for variables and objects, handled automatically by Python's garbage collector.

Q: What are the basic steps involved in exception handling in Python?
A: The steps are: place risky code in try block, handle errors in except block, optionally use else for no-error scenarios, and finally for cleanup.

Q: Why is memory management important in Python?
A: It prevents memory leaks, ensures efficient use of system resources, and keeps applications running smoothly.

Q: What is the role of try and except in exception handling?
A: The try block holds the code that may raise an exception, and the except block defines how to handle the error.

Q: How does Python's garbage collection system work?
A: Python uses reference counting and a cyclic garbage collector to automatically free unused memory.

Q: What is the purpose of the else block in exception handling?
A: The else block runs only if no exceptions are raised in the try block, often used for code that should run when everything works fine.

Q: What are the common logging levels in Python?
A: Common levels include DEBUG, INFO, WARNING, ERROR, and CRITICAL.

Q: What is the difference between os.fork() and multiprocessing in Python?
A: os.fork() creates a child process (Unix only) and duplicates the process. multiprocessing works across platforms and provides higher-level APIs for process management.

Q: What is the importance of closing a file in Python?
A: Closing a file releases system resources and ensures that all data is properly saved.

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.

Q: What is the logging module in Python used for?
A: The logging module provides a flexible way to record messages of different severity levels for monitoring and debugging.

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 operations, directory management, and path handling.

Q: What are the challenges associated with memory management in Python?
A: Challenges include handling circular references, memory fragmentation, and managing large data efficiently.

Q: How do you raise an exception manually in Python?
A: You can use the raise keyword with an Exception class, e.g., raise ValueError('Invalid input').

Q: Why is it important to use multithreading in certain applications?
A: Multithreading improves performance in I/O-bound tasks, allows concurrent execution, and keeps applications responsive.