**Assignment: Python Theory Questions and Answers**

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

* **Interpreted Languages:**
  + Execute code line-by-line at runtime.
  + Slower execution compared to compiled languages.
  + Examples: Python, JavaScript.
* **Compiled Languages:**
  + Translate entire source code to machine code before execution.
  + Faster execution after compilation.
  + Examples: C, C++.

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

* Exception handling is a mechanism to handle runtime errors in a Python program.
* It prevents the program from crashing by catching exceptions and allowing recovery.
* Key keywords: try, except, else, finally.

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

* The finally block is always executed, regardless of whether an exception occurs or not.
* It is commonly used to release resources such as file handles or database connections.

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

* Logging is a way to track events while a program runs.
* It provides messages about errors, information, and warnings.
* Useful for debugging and monitoring.

**5. What is the significance of the del method in Python?**

* The \_\_del\_\_ method is called when an object is about to be destroyed.
* Used to perform cleanup actions like closing files or network connections.

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

* **import module:** Imports the entire module.
  + Example: import math.
* **from module import function:** Imports specific functions or classes from the module.
  + Example: from math import sqrt.

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

* Use multiple except blocks to handle different types of exceptions.
* Example:
* try:
* # Code block
* except ValueError:
* # Handle ValueError
* except KeyError:
* # Handle KeyError

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

* The with statement simplifies file handling by automatically closing the file.
* Ensures resource cleanup.
* Example syntax:
* with open('file.txt', 'r') as file:
* content = file.read()

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

* **Multithreading:**
  + Multiple threads within the same process.
  + Threads share the same memory space.
  + Better for I/O-bound tasks.
* **Multiprocessing:**
  + Multiple processes with separate memory spaces.
  + Better for CPU-bound tasks.

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

* Helps in debugging and error tracking.
* Provides information on the program's execution flow.
* Can store logs in files for future reference.
* Allows different log levels like INFO, WARNING, and ERROR.

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

* Python has an automatic memory management system.
* Includes dynamic allocation, garbage collection, and memory optimization.
* Managed by the Python Memory Manager.

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

1. **Try Block:** Code that may raise an exception.
2. **Except Block:** Handles the exception.
3. **Else Block:** Executes if no exception occurs.
4. **Finally Block:** Executes regardless of exceptions.

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

* Ensures efficient use of memory resources.
* Prevents memory leaks.
* Enhances program performance.

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

* **Try:** Defines a block of code that might raise an exception.
* **Except:** Handles specific exceptions and provides a recovery mechanism.

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

* Python uses a garbage collection system to free unused memory.
* Uses reference counting and cyclic garbage collection.
* Triggered automatically but can also be controlled using the gc module.

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

* The else block runs only if no exception is raised in the try block.
* Useful for executing code that should only run when no errors occur.

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

* **DEBUG:** Detailed information for debugging.
* **INFO:** General information about program execution.
* **WARNING:** Indicates potential problems.
* **ERROR:** A major issue that may disrupt the program.
* **CRITICAL:** Serious errors causing program failure.

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

* **os.fork():** Creates a child process on Unix-based systems.
  + Difficult to use and not cross-platform.
* **Multiprocessing:** Cross-platform and provides high-level API for process creation.

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

* Frees system resources.
* Prevents data corruption and file access issues.
* Ensures data is written and saved correctly.

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

* **file.read():** Reads the entire content of the file as a single string.
* **file.readline():** Reads one line at a time from the file.

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

* The logging module is used to record (log) events while a program runs.
* It helps developers track issues, debug code, and monitor application behavior.
* Provides various log levels like DEBUG, INFO, WARNING, ERROR, and CRITICAL.
* Supports log formatting and writing to external files for persistent records.

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

* The os module provides functions for interacting with the operating system.
* **File Handling Functions:**
  + os.remove(): Deletes a file.
  + os.rename(): Renames a file.
  + os.path.exists(): Checks if a file exists.
  + os.makedirs(): Creates directories.
  + os.listdir(): Lists files in a directory.

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

* **Reference Cycles:** Cyclic references can prevent garbage collection.
* **Memory Fragmentation:** Inefficient memory usage due to fragmented memory blocks.
* **High Memory Consumption:** Dynamic typing and large data structures may lead to excessive memory use.
* **Garbage Collection Delays:** Unpredictable garbage collection may cause delays in releasing memory.

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

* Use the raise keyword to trigger exceptions manually.
* Example:

python

CopyEdit

age = -1

if age < 0:

raise ValueError("Age cannot be negative")

* Custom exceptions can also be defined by subclassing Exception.

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

* **Improved Performance:** Helps execute I/O-bound tasks concurrently.
* **Better Resource Utilization:** Efficiently uses CPU resources for parallel processing.
* **Responsiveness:** Keeps applications responsive by offloading tasks to separate threads.
* **Real-Time Data Handling:** Suitable for real-time applications like chat systems and server requests.