Exception Handling Assignment

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

Interpreted languages (like Python) execute code line-by-line using an interpreter. Compiled languages (like C++) convert the entire code to machine language before execution, which runs faster but requires compilation.

---

2. What is exception handling in Python?

Exception handling allows you to manage errors during runtime using **try**, **except**, **else**, and **finally** blocks, preventing the program from crashing when an error occurs.

---

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

The **finally** block is always executed after the **try** and **except** blocks. It is used for cleanup actions like closing files or releasing resources.

---

4. What is logging in Python?

Logging in Python is used to record messages about program execution, errors, or status. It helps in debugging, monitoring, and auditing programs.

---

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

The **__del__** method is a destructor called automatically when an object is deleted. It helps in resource cleanup like closing files or network connections.

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

import module loads the entire module. **from module import item** imports only specific items (functions, classes) from the module for direct use.

---

7. How can you handle multiple exceptions in Python?

You can use multiple **except** blocks for different exceptions or group them in a tuple like: **except (TypeError, ValueError):**.

---

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

The **with** statement automatically manages file opening and closing. It ensures the file is closed properly, even if an error occurs during file operations.

---

9. What is the difference between multithreading and multiprocessing?

Multithreading runs multiple threads in a single process and shares memory. Multiprocessing runs separate processes with independent memory, ideal for CPU-bound tasks.

---

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

Logging helps monitor the flow of execution, track errors, and diagnose problems without stopping the program. It provides valuable runtime information.

11. What is memory management in Python?

Memory management in Python is automatic. It uses reference counting and garbage collection to allocate and free memory as needed during program execution.

---

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

Write risky code inside **try**, handle errors in **except**, run safe code in **else**, and clean up using **finally**. This prevents program crashes.

---

13. Why is memory management important in Python?

Good memory management avoids memory leaks, improves performance, and ensures that unused objects are removed to free up system resources.

---

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

The **try** block contains code that may raise an error, while **except** handles the error gracefully, preventing program termination.

---

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

Python’s garbage collector uses reference counting and cyclic garbage collection to identify and delete unused objects, freeing memory automatically.

---

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. It’s used to separate error-free logic from error handling.

17. What are the common logging levels in Python?

Python's logging levels are: **DEBUG**, **INFO**, **WARNING**, **ERROR**, and **CRITICAL**. Each level shows the severity or importance of log messages.

---

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

os.fork() creates a child process (Unix-only). **multiprocessing** works across platforms and provides a higher-level API for process-based parallelism.

---

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

Closing a file ensures that all data is properly saved and system resources are freed. It also prevents file corruption and access issues.

---

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

file.read() reads the entire file content at once as a string. **file.readline()** reads just one line from the file each time it’s called.

---

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

The **logging** module is used to track events during a program's execution. It helps in debugging, performance monitoring, and error reporting.

---

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

The **os** module allows interaction with the operating system for tasks like file creation, deletion, path handling, and directory navigation.

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

Challenges include identifying memory leaks, managing circular references, and optimizing memory usage in large or long-running programs.

---

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

Use the **raise** keyword with an exception type like this:

```
raise ValueError("Invalid input")
```

---

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

Multithreading is useful in I/O-bound programs (like file or network operations) because it allows concurrent task execution, improving responsiveness and speed.