Name:- Vaibhav Rajabhau Padule
Email ID:- padulevaibhav18@gmail.com
Assignment Link:-
Files ,functional handling ,logging and memeory…
Drive Link:-
Files ,functional handling ,logging and memeory…
Github Link:-Code

Files, exceptional handling, logging and memory management Questions

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

Ans:- Interpreted languages are executed line-by-line by an interpreter at runtime (e.g., Python, JavaScript).

Compiled languages are translated into machine code beforehand by a compiler, creating an executable (e.g., C, C++).

✅ 2. What is exception handling in Python?

Ans:- Exception handling is a way to manage errors gracefully using blocks like try, except, else, and finally so programs don't crash unexpectedly.

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

Ans:- The finally block always runs, whether or not an exception occurred. It's used for cleanup actions, like closing files or releasing resources.

✅ 4. What is logging in Python?

Ans:-Logging is the process of recording events or messages during a program's execution for debugging, auditing, or monitoring purposes.

✅ 5. What is the significance of the _del_ method in Python?

Ans:-The _del_ method is a destructor called when an object is about to be destroyed. It allows you to define cleanup actions.

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

Ans:- import module imports the whole module; you access members as module.member.

from module import member imports specific members directly into your namespace.

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

Ans:-Use a tuple of exceptions:

```
try:
    # code
except (TypeError, ValueError) as e:
    print(e)
```

Or multiple except blocks:

```
try:
    # code
except TypeError:
    # handle TypeError
except ValueError:
    # handle ValueError
```

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

Ans:-It automatically manages file opening and closing, even if exceptions occur, ensuring proper resource management:

```
with open("file.txt") as f:
    data = f.read()
```

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

Ans:-Multithreading: Multiple threads share the same process memory space. Good for I/O-bound tasks.

Multiprocessing: Multiple processes have separate memory spaces. Better for CPU-bound tasks.

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

Helps debug issues

Records runtime information

Tracks application flow

Maintains audit trails

Offers flexible levels of detail (INFO, DEBUG, ERROR, etc.)

✅ 11. What is memory management in Python?

Ans:-Memory management refers to how Python allocates, tracks, and reclaims memory, using techniques like automatic garbage collection.

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

Ans:-Wrap code in a try block.

Handle exceptions in except blocks.

Optionally use else for code to run if no exception occurs.

Use finally for cleanup actions.

✅ 13. Why is memory management important in Python?

Ans:-Efficient memory management prevents:

Memory leaks

Program slowdowns

Crashes

Unnecessary resource consumption

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

Ans:-
try: Contains code that might raise an exception.

except: Contains code to handle the exception if one occurs.

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

Ans:-
Tracks object references.

Removes objects with zero references.

Uses a cyclic garbage collector to detect reference cycles.

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

Ans:-
The else block runs only if no exception was raised in the try block.

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

Ans:-
From lowest to highest severity:

DEBUG

INFO

WARNING

ERROR

CRITICAL

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

Ans:-

os.fork() creates a child process identical to the parent but is UNIX-only and lower-level.

multiprocessing is a cross-platform module offering a higher-level interface for creating separate processes.

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

Ans:-

Frees up system resources

Ensures data is properly written (flushes buffers)

Prevents data corruption

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

Ans:-

file.read(): Reads the entire file (or given number of bytes).

file.readline(): Reads one line at a time.

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

Ans:-

It provides a flexible way to:

Record events

Write logs to files, streams, or other outputs

Manage log levels and formatting

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

Ans:-

Interacting with the operating system

Performing file and directory operations (e.g. os.remove(), os.rename(), os.path.exists())

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

Ans:-

Circular references

Large memory consumption for complex objects

Difficulty predicting memory usage in large applications

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

Ans:-

Use the raise statement:

raise ValueError("An error occurred!")

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

Ans:-

Improves responsiveness in I/O-bound applications

Allows concurrent execution of tasks

Utilizes idle time during waiting operations