

Files, exceptional handling, logging and memory management Questions text

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

ANS:- The main difference between interpreted and compiled language is that compiled language are converted into machine code before execution also its faster execution process and interpreted language are translated line by line during runtime ,which makes them comparatively slower .

Q.2/ What is exception handling in Python ?

ANS:- Exception handling is a special way to deal erros in python , so our code does not get crashed . we can use special keyword like try,except,finally for exception handling .

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

ANS:- The finally block in Python is used to write code that should run no matter what happens – whether an exception occurs or not.It is mainly used for cleanup tasks like closing a file, releasing resources, or ending a database connection.

Q.4/ What is logging in Python ?

ANS:- Logging is a special method in python , it can procces reloading message like errors, warnings, or normal execution details .

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

ANS:- Del method is the special method in python which use for clean up task when an object is deleted from memory.

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

ANS:- import module Imports the entire module, and we must use the module name to access its functions or classes.from module import name Imports only a specific function/class/variable from the module, and it can be used directly without the module name .

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

ANS:- we can handel multiple expection in python by using key-word like try,except, etc .

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

ANS:- The with statement in Python is used for resource management . It can manage files safely and automatically close them after use

Q.9/ What is the difference between multithreading and multiprocessing ?

ANS:- Multithreading Uses multiple threads within a single process. Threads share the same

memory, so it is lightweight but less stable if not handled properly.

Multiprocessing Uses multiple separate processes. Each process has its own memory, making it more powerful and stable but heavier.

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

ANS:- Logging in Python is better than print() because it helps to track errors, warnings, and program flow. It supports different log levels, can save logs to files, and makes debugging and monitoring easier .

Q.11/ What is memory management in Python ?

ANS:- Memory management in Python is the process of allocating and releasing memory automatically while a program runs.

Python has a built-in garbage collector that removes unused objects to free up memory. It also uses reference counting to keep track of how many variables point to an object.

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

ANS:- The basic steps in exception handling are:

try block - Write the code that may cause an error. except block - Handle the error if it occurs.

else block - Runs if no error occurs. finally block - Runs always, for cleanup tasks.

Q.13/ Why is memory management important in Python ?

ANS:- Memory management in Python is important because it ensures efficient use of RAM, prevents memory leaks, and makes programs run faster and more reliably.

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

ANS:- In python try and except are core building blocks for exception handling ,their role is to prevent the programm from csshing when an error occurs and instead handle it gracefully . In try block we the put risky code there and the except block handle the error if it happens.

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

ANS:- Python gaebage collection system is responsible for freeing up memory by removing objects that are no longer use . Python's garbage collection system is a two-part process that automatically manages memory to prevent leaks and improve efficiency. It primarily relies to cyclic garbage collector to handle more complex scenarios .

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

ANS:- The else block in a (try...except) statement is used to execute code that should only run if the code in the try block succeeds without raising an exception. Its primary purpose is to clearly

separate the logic that might fail from the logic that depends on a successful outcome, improving code readability .

Q.17/ What are the common logging levels in Python ?

ANS:- The common logging levels in python are actually warning ,debug ,info etc.

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

ANS:- The main difference between `os.fork()` and multiprocessing is that, `os.fork()` is a low-level system call available only on Unix/Linux and the `multiprocess` is a high-level Python module that allows process creation in a platform-independent way .

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

ANS:- The importance of closing a file in python is to

- 1/Save data,
- 2/Prevent corruption,
- 3/Avoid future access issues

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

ANS:- The main difference between `file.read()` and `file.readline()` is that ,when we want to read the entire file ,we can use `file.read()` method but when we want to read the file line by line we can use `file.readline()` method .

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

ANS:- The logging module is used to:

- Record program events, errors, and debugging info.
- Replace print with a professional logging system.

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

ANS:- The `os` module in Python is a built-in module that provides functions to interact with the operating system. When it comes to file handling, the `os` module is very useful, it can -

- 1/Create, remove, rename files and folders
- 2/Check file/directory existence and properties
- 3/List and manage file structures

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

ANS:- Challenges in Python memory management include -

- Cyclic references
- Memory leaks
- High memory usage for large datasets

Memory fragmentation
 Garbage collector overhead
 GIL limitations
 Developers not managing references properly

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

ANS:- In Python, you can raise an exception manually using the raise keyword. This is useful when you want to signal that something went wrong, even if Python itself doesn't throw an error automatically.

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

ANS:- Multithreading is important because it:-

Keeps applications responsive
 Handles I/O-bound tasks efficiently
 Allows concurrent execution of tasks
 Simplifies resource sharing within a program


PRACTICE QUESTION

Q.1/ How can you open a file for writing in Python and write a string to it ?

```
with open("assignment.txt", "w") as file:
    file.write("This is my assignment content.\n")
```

Q.2/ Write a Python program to read the contents of a file and print each line ?


```
try :
    with open("assignment.txt", "r") as file:
        for line in file:
            print(line, end='')
except FileNotFoundError:
    print("Error: The file does not exist.")
```

 This is my assignment content.

Q.3/ How would you handle a case where the file doesn't exist while trying to open it for reading ?


```
try:
    with open("srouce.txt", "r") as file:
        for line in file:
            print(line, end='')
except:
```

```
except FileNotFoundError:
    print("Error: The file does not exist.")
```

 Error: The file does not exist.


Q.4/ Write a Python script that reads from one file and writes its content to another file ?

```
with open("assignment.txt", "r") as source_file:
    content = source_file.read()
with open("destination.txt", "w") as dest_file:
    dest_file.write(content)
print("Content copied successfully!")
```

 Content copied successfully!


Q.5/ How would you catch and handle division by zero error in Python ?

```
try:
    numerator = 10
    denominator = 0
    result = numerator / denominator
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Cannot divide by zero.")
```

 Error: Cannot divide by zero.

Q.6/ Write a Python program that logs an error message to a log file when a division by zero exception occurs ?

```
import logging
logging.basicConfig(filename="error_log.txt", level=logging.ERROR,
                    format="%(asctime)s - %(levelname)s - %(message)s")
try:
    numerator = 10
    denominator = 0
    result = numerator / denominator
    print("Result:", result)
except ZeroDivisionError:
    logging.error("Division by zero occurred!")
    print("Error: Division by zero. Check log file for details.")
```

 ERROR:root:Division by zero occurred!
Error: Division by zero. Check log file for details.

Q.7/How do you log information at different levels (INFO, ERROR, WARNING) in Python using the logging module ?

```
import logging
logging.basicConfig(filename="app_log.txt", level=logging.INFO,
                    format="%(levelname)s - %(message)s")
logging.info("This is an info message.")
logging.warning("This is a warning message.")
logging.error("This is an error message.")
```

```
➞ WARNING:root:This is a warning message.
   ERROR:root:This is an error message.
```

Q.8/ Write a program to handle a file opening error using exception handling ?

```
try:
    with open("myfile.txt", "r") as file:
        content = file.read()
        print(content)
except FileNotFoundError:
    print("Error: The file does not exist.")
```

```
➞ Error: The file does not exist.
```

Q.9/ How can you read a file line by line and store its content in a list in Python ?

```
try :
    with open("myfile.txt", "r") as file:
        lines = file.readlines()
        print(lines)
except FileNotFoundError:
    print("Error: The file does not exist.")
```

```
➞ Error: The file does not exist.
```

Q.10/ How can you append data to an existing file in Python ?

```
with open("myfile.txt", "a") as file:
    file.write("This line will be added at the end.\n")
```

Q.11/ Write a Python program that uses a try-except block to handle an error when attempting to access a dictionary key that doesn't exist ?

```
student = {
    "name": "Soumya",
    "age": 20,
    "course": "Data Science"
}
try:
```

```
print("Student grade:", student["grade"])
except KeyError:
    print("Error: The key 'grade' does not exist in the dictionary.")
```

➞ Error: The key 'grade' does not exist in the dictionary.

Q.12/ Write a program that demonstrates using multiple except blocks to handle different types of exceptions ?

```
try:
    numbers = [10, 20, 30]
    index = int(input("Enter an index (0-2): "))
    divisor = int(input("Enter a number to divide: "))
    result = numbers[index] / divisor
    print("Result:", result)
except ZeroDivisionError:
    print("Error: You cannot divide by zero!")
except ValueError:
    print("Error: Please enter only numbers!")
except IndexError:
    print("Error: Index out of range! Choose 0, 1, or 2.")
```

➞ Enter an index (0-2): 6
Enter a number to divide: 9
Error: Index out of range! Choose 0, 1, or 2.

Q.13/ How would you check if a file exists before attempting to read it in Python ?


```
import os
filename = "example.txt"
if os.path.exists(filename):
    with open(filename, "r") as file:
        content = file.read()
        print("File content:\n", content)
else:
    print("Error: File does not exist!")
```

➞ Error: File does not exist!

Q.14/ Write a program that uses the logging module to log both informational and error messages ?


```
import logging
logging.basicConfig(level=logging.INFO)
try:
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    result = a / b
    logging.info("Division successful! Result = %f", result)
except ZeroDivisionError:
```

```
logging.error("Error: Division by zero!")
except ValueError:
    logging.error("Error: Please enter valid numbers!")
```

 Enter first number: 50
Enter second number: 25

Q.15/ Write a Python program that prints the content of a file and handles the case when the file is empty ?

```
try:
    filename = "example.txt" # file name
    with open(filename, "r") as file:
        content = file.read()
        if content.strip() == "":
            print("The file is empty.")
        else:
            print("File content:\n", content)
except FileNotFoundError:
    print("Error: The file does not exist.")
```

 Error: The file does not exist.

Q.16/ Demonstrate how to use memory profiling to check the memory usage of a small program ?


```
from memory_profiler import profile
@profile
def create_list():
    numbers = [i for i in range(1000000)]
    total = sum(numbers)
    print("Sum of numbers:", total)
if __name__ == "__main__":
    create_list()
```

 [Show hidden output](#)

Next steps: [Explain error](#)

Q.17/ Write a Python program to create and write a list of numbers to a file, one number per line ?

```
numbers = [10, 20, 30, 40, 50]
with open("numbers.txt", "w") as file:
    for num in numbers:
        file.write(str(num) + "\n")
print("Numbers written to numbers.txt successfully!")
```

 Numbers written to numbers.txt successfully!

Q.18/ How would you implement a basic logging setup that logs to a file with rotation after 1MB ?

```
import logging
from logging.handlers import RotatingFileHandler
handler = RotatingFileHandler("app.log", maxBytes=1_000_000, backupCount=2)
logging.basicConfig(level=logging.INFO, handlers=[handler])
for i in range(10000):
    logging.info(f"Message {i}")
```

Q.20/ Write a program that handles both IndexError and KeyError using a try-except block ?

```
try:
    numbers = [1, 2, 3]
    print("Accessing 5th element:", numbers[4])
    student = {"name": "Soumya", "age": 20}
    print("Student grade:", student["grade"])
except IndexError:
    print("Error: List index out of range!")
except KeyError:
    print("Error: Dictionary key not found!")
```

➞ Error: List index out of range!

Q.21/ Write a Python program that reads a file and prints the number of occurrences of a specific word ?

```
filename = "assignment.txt"
word_to_find = "python"

with open(filename, "r") as file:
    content = file.read().lower()
    count = content.split().count(word_to_find.lower())
    print(f"The word '{word_to_find}' occurred {count} times in the file.")
```

➞ The word 'python' occurred 0 times in the file.

Q.22/How can you check if a file is empty before attempting to read its contents ?

```
import os
filename = "pw.txt"
if os.path.exists(filename):
    if os.path.getsize(filename) == 0:
        print("The file is empty.")
    else:
        with open(filename, "r") as file:
            print(file.read())
```

```
else:  
    print("File does not exist!")
```

⇒ File does not exist!

Q.23/ Write a Python program that writes to a log file when an error occurs during file handling ?

```
import logging  
logging.basicConfig(filename="assignment.txt", level=logging.ERROR)
```

```
try:  
    with open("example.txt") as f:  
        print(f.read())  
except FileNotFoundError:  
    logging.error("File not found!")  
    print("Error: File does not exist!")
```

⇒ ERROR:root:File not found!
Error: File does not exist!