Python OS Library – Full Practical Mastery Test

All questions are 100% practical and focused on real-world data analytics automation. No theory. Complete every task using Python's os library.

1. Print the current working directory.
2. Change the working directory to a folder named data_project.
3. Verify programmatically that the directory change was successful.
4. Print the absolute path of the script being executed.
5. Join the path data/raw/sales.csv using os.path.join.
6. Convert a relative path into an absolute path.
7. Extract only the directory name from a full file path.
8. Extract only the file name from a full file path.
9. Split a path into directory and file name.
10. Normalize a path containing mixed slashes.
11. Create raw, processed, and reports directories.
12. Create folders only if they do not exist.
13. Check whether raw folder exists before reading files.
14. Create a nested directory in one command.
15. Delete an empty directory programmatically.
16. Rename processed directory to cleaned.
17. List all subdirectories inside project directory.
18. Count how many folders exist inside a directory.
19. Detect and print only directories.
20. Detect and print only files.
21. Check if sales.csv exists before loading it.
22. Validate that a file is readable.
23. Validate that a file is writable.
24. Get file size in bytes and convert it to MB.
25. Reject files larger than 50MB.
26. Identify whether a path is a file or folder.
27. Find the extension of a file.
28. Check if a file is empty.
29. Compare two files and identify the larger one.
30. Print the last modified timestamp of a file.
31. List all CSV files in a directory.
32. Move all CSV files from raw to processed.
33. Rename all CSV files by appending today's date.
34. Delete all temporary .tmp files.
35. Write all file names into a log file.
36. Create a backup folder and move old files into it.
37. Separate files into folders based on extension.
38. Identify duplicate file names in a directory.
39. Rename all files to lowercase.
40. Replace spaces in file names with underscores.
41. Read an environment variable named DB_PASSWORD.
42. Set a temporary environment variable in Python.
43. Check if an environment variable exists.
44. Load a project base directory from env variable.
45. Print all environment variables starting with PY.
46. Use env variable to control input file location.
47. Switch between DEV and PROD paths using env variables.
48. Validate required env variables at startup.
49. Mask sensitive env variables in logs.
50. Exit script if required env variable missing.
51. Run system command to list directory contents.
52. Capture output of system command.
53. Create folder using OS command.
54. Delete folder using OS command.
55. Execute Python script from another script.

56. Detect the operating system.
57. Write OS-specific path handling logic.
58. Pause execution for user input.
59. Clear terminal screen programmatically.
60. Return exit status codes.
61. Log file creation events.
62. Log file deletion events.
63. Create daily execution log file.
64. Log errors to separate file.
65. Capture system errors during file operations.
66. Track number of files processed.
67. Log execution start and end time.
68. Calculate and log script runtime.
69. Create alert log if no files found.
70. Append logs without overwriting.
71. Validate incoming data folder structure.
72. Stop execution if required folders missing.
73. Archive processed files automatically.
74. Auto-create daily report folders.
75. Delete files older than 30 days.
76. Ensure raw data is not modified.
77. Prevent multiple script instances.
78. Generate summary report of file operations.
79. Detect partial file downloads.
80. Create reusable file management utility.
81. Build config-driven file mover using env variables.
82. Create safe file deletion function.
83. Recover from partial failures.
84. Build CLI-style file organizer.
85. Create cross-platform file pipeline.
86. Implement dry-run mode.
87. Implement rollback for failures.
88. Create checksum-based validation.
89. Handle permission errors gracefully.
90. Build automated data ingestion watcher.
91. Build complete data ingestion automation pipeline.
92. Generate final summary report with stats.