QUESTION 1 - Which of the following is NOT a state variable visibility level in solidity?
Answer: d. Exclusive
Justification: Solidity offers three visibility levels for state variables:

- **Public:** Accessible from anywhere, inside and outside the contract.
- **Internal:** Accessible within the contract and inherited contracts (if applicable).
- **Private:** Accessible only within the contract where it's declared.

There's no "Exclusive" visibility level in Solidity.

QUESTION 2 - In Solidity, the require statement:
Answer: b. Ensures certain logical conditions in the program are satisfied before execution continues beyond that point.
Justification: The require statement acts as a conditional check in Solidity. If the provided condition evaluates to false, the smart contract throws an error, halting further execution. This helps enforce specific rules and prevent unexpected behavior.
Option A (providing dependencies) is not the purpose of require.

QUESTION 3 - Which of the following might be used as a debugging method for Solidity?
Answer: Both A & B are correct
Justification: Solidity offers limited debugging capabilities due to its nature on the blockchain.

- print **statement:** Although not directly writing to the console, print allows storing debug information in the contract's storage for later retrieval through tools like Remix or truffle.
- **Events:** Emitting events with relevant data during contract execution provides valuable insights into the contract's state and function calls. Debuggers can then capture and analyze these events.

QUESTION 4 - A best-practice technique for retrieving external information such as stock prices inside a Solidity smart contract might be:
Answer: c. Use a data feed from a decentralized source such as Chainlink.
Justification: Chainlink (or similar decentralized oracles) is the best practice for external data like stock prices. It offers enhanced security against manipulation compared to scraping or APIs. Additionally, it maintains decentralization (a core blockchain principle) and frees you from managing data feeds within the smart contract itself.
options A and B are incorrect and also less ideals:

- **Scraping:** Scraping websites directly exposes the contract to potential changes in website structure or security issues.
- **APIs:** While potentially easier, relying on centralized APIs introduces a single point of failure and may not be as tamper-proof as decentralized oracles.