File types – advantages, disadvantages & use cases

CSV (Comma-Separated Values):

- Advantages:
    - Simple and lightweight format.
    - Human-readable and widely supported.
    - Easy to import and export data from various applications.
    - Suitable for tabular data with a simple structure.
- Disadvantages:
    - Limited support for complex data structures.
    - Lacks standardization for data types and metadata.
    - Does not support nested objects or relationships.
- Common Use Cases:
    - Storing and exchanging tabular data (e.g., spreadsheets, database exports).
    - Log files and data transfer between systems.
    - Simple data storage and interchange in data analysis or data manipulation tasks.

JSON (JavaScript Object Notation):

- Advantages:
    - Lightweight and human-readable format.
    - Supports complex data structures and nested objects.
    - Widely supported by programming languages and APIs.
    - Offers flexibility in representing various data types.
- Disadvantages:
    - Slightly larger file size compared to more compact formats.
    - Not as efficient for tabular or highly structured data.
    - Requires parsing and serialization/deserialization.
- Common Use Cases:
    - Web APIs for data interchange.
    - Configuration files and settings storage.
    - NoSQL databases and document-based storage.

- Interoperability between different systems or programming languages.

SQLite:

- Advantages:
    - Self-contained and serverless relational database engine.
    - Lightweight and efficient for small to medium-sized databases.
    - Supports ACID transactions and SQL querying.
    - Good performance and cross-platform compatibility.
- Disadvantages:
    - Not suitable for high-traffic or large-scale applications.
    - Limited concurrent access for write-intensive scenarios.
    - Lacks some advanced features of enterprise-grade databases.
- Common Use Cases:
    - Desktop or mobile applications requiring local data storage.
    - Prototyping and small-scale applications.
    - Embeddable databases in applications.
    - Storing structured data with relational querying capabilities.

Pickle:

- Advantages:
    - Supports serialization/deserialization of complex Python objects.
    - Retains object relationships, methods, and class definitions.
    - Preserves data structure and state.
- Disadvantages:
    - Limited compatibility with other programming languages.
    - Not suitable for long-term storage or data exchange between systems.
    - Security risks when unpickling untrusted data.
- Common Use Cases:
    - Caching and memorization of computationally expensive results.
    - Storing and retrieving Python-specific data structures and objects.
    - Inter-process communication and sharing of Python objects.

SQLAlchemy:

- Advantages:

    - High-level and object-relational mapping (ORM) capabilities.
    - Provides an abstraction layer for working with databases.
    - Supports multiple database backends (e.g., SQLite, MySQL, PostgreSQL).
    - Offers query building and management utilities.
- Disadvantages:
    - Learning curve associated with ORM concepts and configuration.
    - Additional overhead and complexity compared to lower-level approaches.
    - May not cover all advanced database-specific features.
- Common Use Cases:
    - Large-scale applications requiring complex data modeling and relationships.
    - Data-driven web applications and APIs.
    - Mapping database tables to Python objects.
    - Cross-database compatibility and portability.