DPP Solution : Understanding Data Types and Data Sources

Course: Data Science & Analytics

Topic: Data Types, Data Structures, and Datasources

Day 1: Understanding Data Types

Theory Questions

1. Explain the differences between primitive and non-primitive data types with examples.

Answer:

* ***Primitive Data Types:*** *These are the most basic building blocks of data. They usually store a single value directly in memory.*

  * *Examples:* `int` (Integer, e.g., 5), `float` (Decimal, e.g., 3.14), `bool` (Boolean, e.g., True), `str` (String, e.g., "Hello").*

* ***Non-Primitive Data Types:*** *These are more complex and are used to store collections of data. They are often objects that reference memory locations where the data is stored.*

  * *Examples:* `list` ([1, 2, 3]), `dictionary` ({"key": "value"}), `set`, `tuple`.*

2. Why is it important to choose the correct data type in a program? Give a real-life example.

Answer:

Choosing the right data type ensures memory efficiency, performance optimization, and data integrity.

* *Real-life Example:* If you are storing a user's age, using a `float` (e.g., 25.0000) is unnecessary and wasteful compared to an `int` (25). Conversely, if you store a phone number as an `int`, you might lose leading zeros; storing it as a `string` preserves the exact formatting required for dialing.*

3. How do memory allocation and data type influence program performance? Explain with examples.

Answer:

* ***Memory:*** *Primitive types generally consume a fixed, small amount of memory. Non-primitive types (like lists) consume more memory as they store references to objects and have overhead for dynamic resizing.*

* ***Performance:*** *Operations on primitive types are generally faster because they map to CPU-level instructions. Iterating over a massive linked list (non-primitive) is slower than accessing an array index due to memory locality (cache hits).*

4. Describe the key characteristics of a string data type and give a scenario where it is commonly used.

Answer:

* ***Characteristics:*** *Strings are* ***immutable*** *(cannot be changed in place) and are ordered sequences of characters. They support indexing and slicing operations.*

* ***Scenario:*** *Storing a user's email address, a product SKU code, or a URL.*

Practical Assignment

Task: Write a Python program that accepts different data types, identifies them, and stores them in a dictionary.

- **Solution File:** `day1_data_types.py`

Day 2: Data Structures and Their Applications

Theory Questions

1. What is the difference between an array and a linked list? Provide a real-world example where each would be used.

Answer:

* ***Array:*** *Stores elements in contiguous (side-by-side) memory locations. It allows fast access (O(1)) but slow insertion/deletion (O(n)) because elements must shift.*

  * **Use Case:* Storing pixel data in an image (matrix) where random access to specific pixels is frequent.*

* ***Linked List:*** *Elements (nodes) are scattered in memory, linked by pointers. It allows fast insertion/deletion (O(1) if the pointer is known) but slow access (O(n)) because you must traverse the chain.*

  * **Use Case:* Implementing a music playlist where songs are frequently added or removed from the middle of the list.*

2. Explain the working of a stack using the LIFO principle with a practical scenario.

Answer:

* ***Principle: LIFO*** *(Last In, First Out). The last element added to the stack is the first one removed.*

* ***Scenario:*** *The "Undo" feature in a text editor. The last action you performed is the first one to be undone.*

3. Compare and contrast hash maps and arrays. When would you prefer one over the other?

Answer:

* ***Hash Map:*** *Uses key-value pairs. Lookup is O(1) on average. Preferred when data is associated with unique identifiers (keys) rather than numerical indices.*

* ***Array:*** *Uses numerical indices. Preferred when order matters, data is sequential, or memory overhead must be minimized.*

4. How do queues work, and where are they used in real-world applications like customer service systems?

Answer:

* ***Principle: FIFO*** *(First In, First Out).*

* ***Scenario:*** *A customer service call center. The first customer to call is the first one to be connected to an agent.*

Practical Assignment

Task: Implement a simple ticketing system using a queue in Python.

- **Solution File:** `day2_ticketing_system.py`

Day 3: Structured, Semi-structured, and Unstructured Data

Theory Questions

1. Define structured, semi-structured, and unstructured data. Provide examples of each from real-life scenarios.

Answer:

* ***Structured:*** *Highly organized data that fits into tables (Rows/Columns). *Example:* SQL Database, Excel sheets.*

* ***Semi-structured:*** *Contains tags or markers to separate elements but enforces no rigid schema. *Example:* JSON, XML, HTML.*

* ***Unstructured:*** *No pre-defined data model. *Example:* Images, Audio files, PDF documents, Social media posts.*

2. Why is it challenging to process unstructured data? Mention tools that are commonly used for this purpose.

Answer:

* ***Challenge:*** *Computers cannot natively "understand" the content (e.g., pixels in an image or intonation in audio) without complex processing. It requires heavy storage and advanced algorithms to extract value.*

* ***Tools:*** *NLP (Natural Language Processing) for text, OpenCV for images, TensorFlow/PyTorch for deep learning analysis.*

3. How does semi-structured data strike a balance between structured and unstructured data? Give examples.

Answer:

It offers the flexibility of unstructured data (you can add new fields easily without breaking the system) while maintaining the hierarchical organization of structured data.

* *Example:* A NoSQL database (MongoDB) storing user profiles where some users have a "social_links" field and others do not.*

Practical Assignment

Task: Write a Python script to load and display structured and semi-structured data.

- **Solution File:** `day3_data_analysis.py`

Day 4: Real-Life Project – E-commerce System

Project Overview

Scenario: Building a backend for an e-commerce system handling Product Catalogs, User Reviews, and Product Images.

Tasks & Solutions

1. Design the structure for the product catalog (Structured Data).

Solution: *We used a List of Dictionaries to represent the catalog, simulating a database table structure.*

Refer to `day4_ecommerce.py` (Class `EcommerceBackend`)

2. Simulate user review data in JSON format and parse it (Semi-structured Data).

Solution: *We parsed a JSON string containing nested review objects and calculated average ratings per product.*

**Refer to `day4_ecommerce.py` (Method `analyze_reviews`)*

3. Outline a process to handle product images (Unstructured Data).

Process Flow:

1. ***Upload:*** *User uploads an image via the frontend.*
2. ***Validation:*** *Backend checks file type (jpg/png) and size.*
3. ***Storage:*** *The actual image file is uploaded to an Object Store (e.g., AWS S3, Google Cloud Storage) to keep the database light.*
4. ***Metadata:*** *Extract metadata (resolution, size) and generate a public URL.*
5. ***Database Entry:*** *Store the *URL* (e.g., `https://bucket.s3.com/img1.jpg`) in the SQL database linked to the Product ID.*

Day 5: Interview Preparation

Theory Questions

1. How would you explain the concept of data types to a non-technical interviewer? Give an analogy.

Answer:

Think of data types as different kinds of containers in a kitchen:

* ***Int:*** *A jar that only holds whole cookies.*
* ***String:*** *A notebook where you write text.*
* ***List:*** *A shopping basket where you can put many different items in order.*
* **Analogy:* If you try to pour water (text) into a wire-mesh basket (integer variable), it won't work properly. You need the right container for the right content.*

2. Discuss a scenario where choosing the wrong data structure could lead to inefficiency.

Answer:

Imagine a library with 1 million books.

* ***Inefficient (Linked List):*** *If you store books in a list and want to find one, you might have to walk through the shelves checking every single book one by one (O(n)).*

* ***Efficient (Hash Map/Index):*** *If you use a Hash Map (or database index), you can look up the book instantly by its ISBN number (O(1)).*

3. Explain how APIs enable the exchange of semi-structured data and why they are widely used.

Answer:

APIs (Application Programming Interfaces) act as messengers between different software systems. They typically use ***JSON*** *(semi-structured) because:*

1. *It is lightweight (fast to transmit).*
2. *It is human-readable.*
3. *It is language-independent (Python, JavaScript, and Java can all read JSON easily).*

Practical Assignment

Task: Mock Interview Preparation.

- **Deliverable:** Review the answers above and the code in `day1` through `day4` to prepare for the mock interview.

Shivansh Yadav