Data Structure with Dynamic Attributes and Default Values- Python Code

1. DataNode Class

The `DataNode` class represents a node in the data structure. It allows attribute access to nested data using dot notation.

Class Description

- `DataNode`: A class representing a node in the Data structure.

Class Methods

- `__init__(self, data=None)`: Initializes a DataNode with optional data.
- `__getattr__(self, item)`: Handles attribute access for the DataNode.
- `__setattr__(self, key, value)`: Handles attribute assignment for the DataNode.
- `__delattr__(self, item)`: Handles attribute deletion for the DataNode.
- `__getitem__(self, item)`: Handles item access for the DataNode.
- `__setitem__(self, key, value)`: Handles item assignment for the DataNode.
- `__delitem__(self, item)`: Handles item deletion for the DataNode.
- `to_dict(self)`: Converts the DataNode to a dictionary.

2. Data Class

The `Data` class represents a dynamic data structure. It allows creating nested data structures and defining default values for attributes.

Class Description

- `Data`: A class representing a dynamic data structure.

Class Methods

- `__init__(self, **kwargs)`: Initializes a Data object with optional data and default_values.
- `from_dict(cls, data)`: Creates a Data object from a dictionary.
- `__getattr__(self, item)`: Handles attribute access for the Data object.
- `__setattr__(self, key, value)`: Handles attribute assignment for the Data object.
- `__delattr__(self, item)`: Handles attribute deletion for the Data object.
- `__getitem__(self, item)`: Handles item access for the Data object.
- `__setitem__(self, key, value)`: Handles item assignment for the Data object.
- `__delitem__(self, item)`: Handles item deletion for the Data object.
- `to_dict(self)`: Converts the Data object to a dictionary.

Example Usage

The provided example demonstrates the functionalities of the Data and DataNode classes.

Data Structure

- Data object `my_inst_1` loaded from a dictionary representing the data structure.
- Data object `my_inst_2` created using keyword arguments.

Dynamic Attribute Access

- Accessing nested data using dot notation (`my_inst_1.metadata.system.size`).

Default Values

- Retrieving a default value (`my_inst_1.metadata.system.height`), which is not set yet.
- Setting a default value for an attribute (`my_inst_1.metadata.system.height = 100`).
- Accessing the updated value (`my_inst_1.to_dict()['metadata']['system']['height']`).

Autocomplete

- Demonstrates the autocomplete feature by printing `my_inst_1.metadata`.

Output

- The expected outputs for each operation are provided as comments in the code.

Note: The code has been properly documented and structured to ensure clarity and maintainability.