OOPs: Episode-1

Manas Jyoti Das, PhD
Computer Science

Python Decorators (@)

- Code reusability and modularity: You can define complex functionality once and then apply it to multiple functions without modifying their original code.
- Adding functionality without code modification: Unlike inheritance, decorators allow you to dynamically enhance functions without changing their source code.
- Enhanced code readability and expressiveness: Using decorators can make your code more concise and clear by expressing what's happening before the function definition.

@property decorator

- Hides internal implementation: Keeps internal data private and inaccessible from outside the class, maintaining data integrity and preventing accidental modifications.
- Change internal implementation without breaking existing code: Can modify property logic without affecting code that uses the property, making code more maintainable.
- Properties can be inherited by subclasses: Extends functionality and facilitates code reuse.
- Properties resemble regular attributes: Makes code more intuitive and easier to understand compared to explicit method calls.

Code...

Read the code carefully and make comments in the code. Otherwise you will forget what is happening in the code. Please try to understand the code.

Coding makes a human perfect. Code regularly or you will forget. Trust me in this one —**Adraham Lincon**

Dunder function

- Magic function or special methods
- double underscores at the beginning and end of their names (e.g., __init__, __len__)
- You typically don't invoke them yourself; Python calls them internally when specific events or operations occur
- Customizable interactions: By implementing dunder methods in your classes, you can create custom objects that behave naturally within Python's syntax and language constructs
- E.g. customize the __len__ to do something else. May be length in bytes.

Code...

Data focused class[Previous class]

- Emphasis: Primarily on storing and managing data.
- Methods: Relatively fewer or simpler, mainly for accessing, manipulating, or validating the data. They might directly interact with the attributes or simply expose them for external use.
- Example: A "Product" class in an e-commerce application might have attributes like name, price, description, and stock level. Its methods might involve updating information, checking availability, or calculating discounts.

@dataclass decorator

- Automatic methods: It automatically generates essential methods like __init__, __repr__, __eq__, and comparison operators
- Concise syntax: The clean and simple syntax makes data classes easier to read and understand compared to traditional classes with manually implemented methods.
- Consistent behavior: Automatically generated methods ensure consistent behavior and reduce potential errors with manual implementations.