OOPs: Episode-II

Manas Jyoti Das, PhD
Computer Science

Python

- Python is a dynamically typed language
- What does it mean
- Type (int, str, etc.) declarations are not required
- Can define a function without ever specifying what types the function is expecting as an argument
- Duck typing
- Code

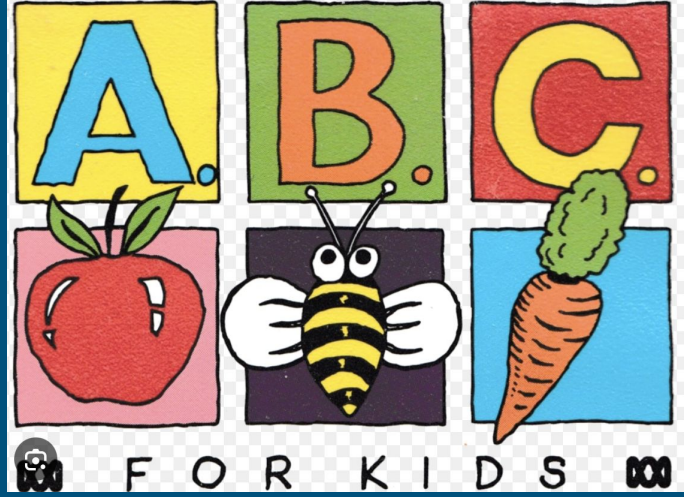
Duck typing

- Runtime errors: Unlike static typing where errors are caught at compile time, duck typing waits until runtime to verify if an object has the required methods.
- Lack of clarity: With duck typing, the contract between objects is implicit, relying on shared names and behavior.
- Debugging difficulties: Debugging code with runtime errors can be challenging as the source of the issue might not be evident immediately
- Potential for security vulnerabilities: Duck typing's flexibility can be exploited by malicious code that implements the required methods but with unintended behavior.

Duck typing solution

- Runtime Checks: `hasattr` or `isinstance`
- Interfaces or Abstract Base Classes:
  - In languages that support them, define interfaces or abstract base classes to outline expected behavior more formally.
- Design Patterns:
  - Utilize design patterns like the Strategy pattern to encapsulate different algorithms or behaviors behind a common interface, promoting flexibility and testability while reducing reliance on duck typing.

ABC



- Enforcing a Common API
- Code Reusability:
  - Common code and logic can be placed in the abstract base class, reducing redundancy and promoting code reuse in subclasses.
- Type Checking and Introspection:
  - Tools like isinstance and issubclass can be used to check if an object is an instance of a particular ABC or its subclass.
- Increased complexity
- Over-Engineering

Protocol

- Provide concrete implementation without inheriting from a base class
- Protocol classes still guide the structure of classes that conform to them, promoting consistency and readability.
- type checking, hint from IDE (added benefit)