Encapsulation in Python is achieved primarily through classes and the judicious

Here are a few examples demonstrating encapsulation in Python:

Example 1: Simple Encapsulation

This example shows a basic class with attributes and methods. We use a namin

```python
class Dog:
    def __init__(self, name, age):
        self._name = name  # Protected attribute (convention only)
        self._age = age    # Protected attribute (convention only)

    def get_name(self):
        return self._name

    def get_age(self):
        return self._age

    def set_age(self, new_age):
        if new_age > 0:
            self._age = new_age
        else:
            print("Age must be positive.")

    def bark(self):
        print("Woof!")

my_dog = Dog("Buddy", 3)
print(my_dog.get_name())  # Accessing name through getter method
print(my_dog.get_age())   # Accessing age through getter method
my_dog.set_age(5)         # Modifying age through setter method
print(my_dog.get_age())
my_dog.bark()

#Trying to access directly (although possible, it's discouraged)
print(my_dog._name) #This works, but breaks encapsulation principle
my_dog._age = -1    # This also works but violates data integrity
print(my_dog._age)
```

Example 2: More Robust Encapsulation (using properties)

Properties provide a cleaner and more Pythonic way to manage attribute access