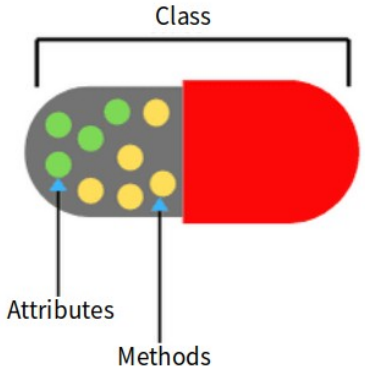
Abstraction and Encapsulation

Abstraction attempts to reduce program complexity by making generalizations about the most basic information and functionality of an object. Consider a car: it has four wheels, drives, and carries passengers. A *specific* car, like a Bugatti Veyron, is an *instance* of a car, not the abstract idea. A large part of designing programs that use OOP principles involves abstracting objects. Some define abstraction as "showing what an object can be or do".

Encapsulation is the practice of bundling data (attributes) and functionality (methods) inside of a single unit (e.g. a class). The mechanics of *how* a method does something are hidden from the user. An analogy is a pill capsule: the user does not need to know how the ingredients work together (complex chemical reactions), only how to use the pill (presumably by swallowing it).



Methods and attributes that are accessible within a class (using `self`) are **public**. In many programming languages, attributes and methods can be made **private**. This prevents them from being accessed directly, requiring setter and getter functions instead. Python does not allow truly private attributes and methods, but they can be "hidden" by prefixing the attribute or method name with two underscores, as in `__age` or `__calculate_score()`. Python *does* allow **protected** attributes and methods, prefixed with a single underscore, but this is more of a convention.

Encapsulation is related to abstraction, as it involves hiding unnecessary details from the user. Using setter and getter methods instead of accessing attributes directly is an example of encapsulation, since the user needs only to call these methods, and does not need to understand how they work internally. Also, if changes are made to the class (e.g. algorithms are modified), the methods can be rewritten so that the user does not notice these changes.

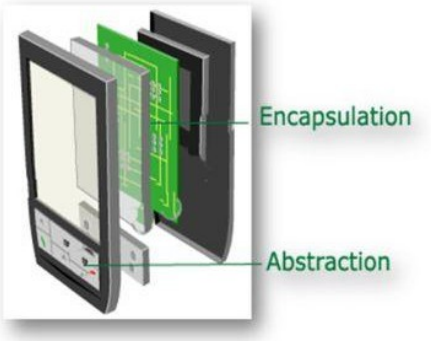
Example of "Private" Attributes and Methods

```
class CLASSNAME:
      def __init__(self, ARGUMENTS):
            # the following attributes are "private"
            self.__ATTRIBUTE1 = ARGUMENT1
            self.__ATTRIBUTE2 = ARGUMENT2
      # the following method is "private"
      def __METHODNAME(self, ARGUMENTS):
            # method code goes here
```

Abstraction and Encapsulation

Answer the following questions.

1. Explain how the following diagram represents encapsulation and abstraction.



2. How would you abstract a television? Describe the main methods and attributes that would be required.

Write programs that accomplish each task, using appropriate programming conventions.

3. Create a class, `Thingamajig`, with "private" attributes `x` and `y`. Initialize `x` and `y` with values of your choosing. Try to access these attributes without using setter and getter methods, to verify that they are "private". Write setter and getter methods to display and change their values.

4. A lightbulb has two states (on and off), and is able to toggle between these two states. Write a class, `Lightbulb`, with one "private" attribute (`state`) and two methods (`toggle` and `display`). These methods act as setter and getter respectively. Write a program that allows the user to switch between on and off, and displays the bulb's state after each switch.

5. Write a class, `Account`, that represents a typical bank account. Attributes include an account number and a balance. Methods should allow the user to deposit funds, withdraw funds, or display the balance of the account. Ensure that the user does not withdraw more money than is currently in the account!

6. Create a `Contact` class, which stores contact information for an individual (similar to one on a mobile phone). Information includes the contact's name (required), and zero or more of the following items: mobile phone number, home phone number, work phone number, and email address. Use a list to store contacts in your main program, and use a loop to obtain information to add or modify. Use write getter and setter functions to add each piece of information. Include the option to display the entire list of contacts, and their associated information.