DATA SCIENCE COURSE TUTORIAL # 38

3.25 Encapsulation in Object Oriented Programming

What Is Encapsulation

Encapsulation means wrapping data and methods inside a single unit called a class. It protects data from direct access and allows controlled interaction. Encapsulation helps secure data and maintain clean program structure.

Why Encapsulation Is Important

Encapsulation provides data protection. It allows controlled access using methods. It prevents accidental changes to important variables. It improves code safety and structure.

Public, Protected and Private Variables

Python uses naming conventions to show how variables should be accessed.

Public Variables Accessible everywhere.

```
self.name = "Ali"
```

Protected Variables Use a single underscore. Meant for internal use.

```
self._age = 20
```

Private Variables Use double underscore. Cannot be accessed directly outside the class.

```
self.__salary = 50000
```

Example of Encapsulation

```
class Person:
    def __init__(self, name, age, salary):
        self.name = name
        self._age = age
        self.__salary = salary

p = Person("Ali", 22, 60000)
```

```
print(p.name)
print(p._age)
```

Private variable usage will give an error if accessed directly.

```
print(p.__salary)
```

This will produce an error.

Accessing Private Variables with Methods

Private variables can be accessed using getter and setter methods.

Getter Method Returns the value of a private variable.

```
class Person:
    def __init__(self, name, age, salary):
        self.name = name
        self._age = age
        self.__salary = salary

    def get_salary(self):
        return self.__salary
```

Usage:

```
p = Person("Aisha", 24, 80000)
print(p.get_salary())
```

Setter Method

Setter method is used to change the value of a private variable.

```
class Person:
    def __init__(self, name, age, salary):
        self.name = name
        self._age = age
        self.__salary = salary

    def set_salary(self, amount):
        if amount > 0:
            self.__salary = amount
```

```python
        else:
            print("Invalid amount.")
```

Usage:

```python
p = Person("Ali", 23, 50000)
p.set_salary(70000)
print(p.get_salary())
```

Encapsulation with Methods

Encapsulation also protects methods by marking them as protected or private.

```python
class Demo:
    def public_method(self):
        print("Public method.")

    def _protected_method(self):
        print("Protected method.")

    def __private_method(self):
        print("Private method.")
```

Benefits of Encapsulation

Encapsulation protects data by preventing unauthorized access. It improves code security and structure. It provides control by using getter and setter methods. It makes large programs easier to manage.

Summary

- Encapsulation means binding data and methods inside a class.
- Data protection is done using public, protected and private variables.
- Getter and setter methods allow controlled access.
- Name mangling protects private data by preventing direct access.
- Encapsulation improves security, organization and clarity of code.