Lecture slides - Week 12

OOP - Abstraction

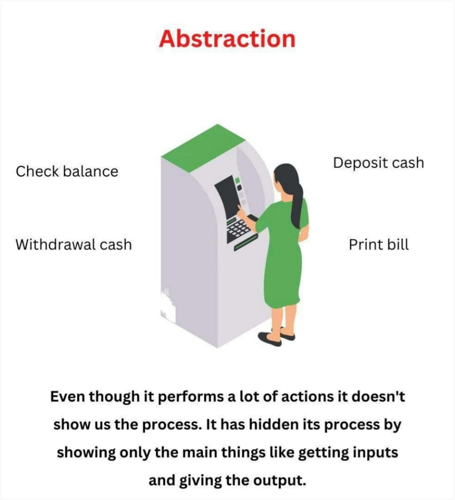
Dr. Aamir Akbar

Director of both AWKUM AI Lab and AWKUM Robotics, Final Year Projects (FYPs) coordinator, and lecturer at the department of Computer Science
Abdul Wali Khan University, Mardan (AWKUM)

Contents

Abstraction

Abstraction in real life and OOP i



Even though it performs a lot of actions it doesn't show us the process. It has hidden its process by showing only the main things like getting inputs and giving the output.

Abstraction in real life and OOP ii

`Abstraction` in OOP is the concept of hiding complex implementation details and showing only the necessary features of an object. It allows you to focus on what an object does instead of how it does it.

In Python, abstraction can be achieved using:

- **Encapsulation:** While not purely abstraction, encapsulation hides the internal state of an object from the outside world. This is achieved by using private variables and methods, denoted by double underscore `__` before their names.

- **Abstract Classes and Methods:** Python has a module named `abc` (Abstract Base Classes) that allows creating abstract classes and abstract methods using the `abstractmethod` decorator. These abstract methods must be implemented by any class that inherits from the abstract class.

Abstract Class and Method

Abstract Classes and Methods

```python
from abc import ABC, abstractmethod

class Animal(ABC):
    def __init__(self, name, health):
        self.name = name
        self.health = health

    @abstractmethod
    def attack(self):
        pass

    @abstractmethod
    def defence(self):
        pass

    def update_health(self, damage):
        self.health -= damage
```

In the above code, the abstract class Animal cannot be instantiated directly because its incomplete. Abstract methods within an abstract class lack implementation details in the base class itself, which must be implemented by any subclass that inherits from the abstract class.

```python
a = Animal("Lion", 100)  # This line will raise a TypeError
```

Person and Employee Abstract Classes

```python
from abc import ABC, abstractmethod

class Person(ABC):
    # Rest of the class code is skipped

class Employee(Person, ABC):
    # Rest of the class code is skipped

class Teacher(Employee):
    # Rest of the class code is skipped

class Staff(Employee):
    # Rest of the class code is skipped

class Student(Person):
    # Rest of the class code is skipped
```

Because Person and Employee are abstract classes, we cannot directly create objects (instances) of them.

```
person = Person()  # Error: Person is an abstract class
employee = Employee()  # Error: Employee is also abstract
```

Case Study: Shapes

Basic Shapes

Develop a `drawing application` that allows users to create shapes: Circle, Square, Rectangle and Triangle.

The application will draw these shapes on the GUI and also print the calculated area of the shape.