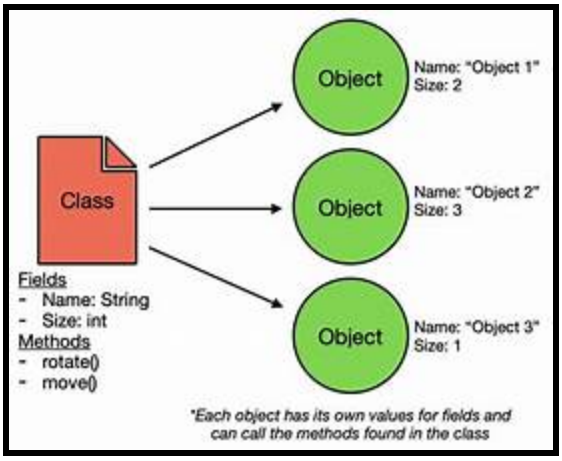
অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং ঃ

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এক ধরনের যুক্তিযুক্ত পরিকল্পনা , যার মাধ্যমে পর্যাপ্ত ডেটা ও Instruction এর সমন্বয়ে একটি চলক তৈরি করা হয় এবং প্রয়োজনই ডেটা প্রদান করে চলকটি ব্যবহার করে এমন একটি প্রক্রিয়া সকল কাজ সম্পাদন করা হয় | এই বিশেষ চলকটিকে বলা হয় "অবজেক্ট" |

OOP প্রোগ্রাম মানেই কতকগুলো অবজেক্টের সমষ্টি যার কিছু নির্দিষ্ট বৈশিষ্ট্য আছে , এসকল বৈশিষ্ট্য সমূহকে বলা হয় **Attribute** বা **behaviou**r | এসকল **Attribute** এর মাধ্যমেই অবজেক্ট স্বতন্ত্র পরিচয় লাভ করে |

সাধারণভাবে প্রোগ্রামিং কোড বা Instruction এবং ডেটার সমন্বিত উপস্থাপনাই হচ্ছে Object | প্রয়োজনীয় ডেটা প্রদান করলে Object নিজ নিয়মেই শব কাজ সম্পাদন করে | সাব- প্রোগ্রামের মতন Object কে অন্য প্রোগ্রামের ব্যবহারের জন্য সংরক্ষণ করা যায় |

Class হচ্ছে object এঁর জন্য একটি template প্রদান করে যার মাধ্যমে object তার কাজ সম্পর্কে ধারনা লাভ করে |
একটি Class এর Behaviour বা Attribute কে রিপ্রেসেন্ট করার জন্য object ব্যবহৃত হয় , তাই Object কে Class এর Instance বলা হয় |




```python
#Class is a blueprint for creating object.

class Student():
    name ="Tisha"
    roll = "1967"
    section = "Science"

#Object Creation
s1 = Student()
print(s1.name,'\t',s1.roll,'\t',s1.section)
```

চিত্র ঃ Python ল্যাঙ্গুয়েজ এর সাহায্যে Class Implementattion

উপরোক্ত চিত্রে Student নামে একটি Class তৈরি করা হয়েছে এবং Class এর তিনটি Attribute (name ,roll ,section) বিদ্যমান | Class এর এই Attribute সমূহ অধিগত করার জন্য প্রয়োজন একটি object , এর প্রেক্ষিতে s1 নামের একটি object তৈরি করা হয়েছে এবং " . " (ডট অপারেটর) দ্বারা Attribute সমূহকে শো করা যাচ্ছে |

Abstraction ঃ Abstraction সংশ্লিষ্ট Class এবং Object এর একটি সাধারণ ধারনা প্রদান করে | এখানে Object শুধুমাত্র Class এর Attribute সমূহকে প্রকাশ করে ,কিন্তু কিভাবে কাজ করে তা প্রকাশ ব্যতীত |

```
class Shape:
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2

class Square(Shape):
    def __init__(self, side_length):
        self.side_length = side_length

    def area(self):
        return self.side_length ** 2

# Using the abstraction
circle = Circle(5)
square = Square(4)

print("Circle Area:", circle.area())
print("Square Area:", square.area())
```

```
Circle Area: 78.5
Square Area: 16
```

Polymorphism ঃ পলিমরফিজম এর মাধ্যমে একটি Class এর সংশ্লিষ্ট Object কে ভিন্ন ভিন্ন ভাবে প্রকাশ করা যায় | এখানে **object , Class** এর **Attribute** অনুযায়ী পরিবর্তন হয় |

```python
class Adder:
    def add(self, x, y):
        return x + y

class Multiplier:
    def add(self, x, y):
        return x * y

def perform_addition(obj, a, b):
    return obj.add(a, b)

adder = Adder()
multiplier = Multiplier()

print(perform_addition(adder, 3, 4))        # Outputs: 7
print(perform_addition(multiplier, 3, 4))   # Outputs: 12
```

```
7
12
```

Encapsulation ঃ Encapsulation এর অর্থ হচ্ছে কোন variable এর ডেটা এবং Instruction একত্রিত অবস্থায় থাকে | Encapsulation এর মাধ্যমে ডেটা পরিবর্তন অথবা পরিবর্ধন এর অধিকার নিয়ন্ত্রণ করা হয় |
এই প্রক্রিয়াকে "Data Hiding " অথবা "Information Hiding " বলা হয় |

In [4]:
```
class BankAccount:
    def __init__(self, balance=0):
        self.__balance = balance  # Encapsulated variable

    def get_balance(self):
        return self.__balance

    def deposit(self, amount):
        self.__balance += amount

    def withdraw(self, amount):
        if amount <= self.__balance:
            self.__balance -= amount
        else:
            print("Insufficient funds")

account = BankAccount(1000)
print("Initial Balance:", account.get_balance())
account.deposit(500)
print("After Deposit:", account.get_balance())
account.withdraw(200)
print("After Withdrawal:", account.get_balance())
```

```
Initial Balance: 1000
After Deposit: 1500
After Withdrawal: 1300
```

Inheritance: Inheritance ল্যাঙ্গুয়েজ প্রদত্ত এমন একটি পাওয়ার যার মাধ্যমে কোন Class কে পরিবর্ধন বা Extend করে নতুন আরেকটি Class তৈরি করা যায় | Inheritance এর ফলে একটি Class এর Attribute নিয়ে নতুন Class তৈরি করা সম্ভভ | একটি Class এর Object তৈরি করা হলে তার মধ্যে Extended Class এর সকল বৈশিষ্ট্য (Attribute) object বর্তমান থাকে |

```
:  class Animal:
       def speak(self):
           pass

   class Dog(Animal):
       def speak(self):
           return "Woof!"

   class Cat(Animal):
       def speak(self):
           return "Meow!"

   dog = Dog()
   cat = Cat()

   print(dog.speak())
   print(cat.speak())
```

```
Woof!
Meow!
```