Programming Fundamental

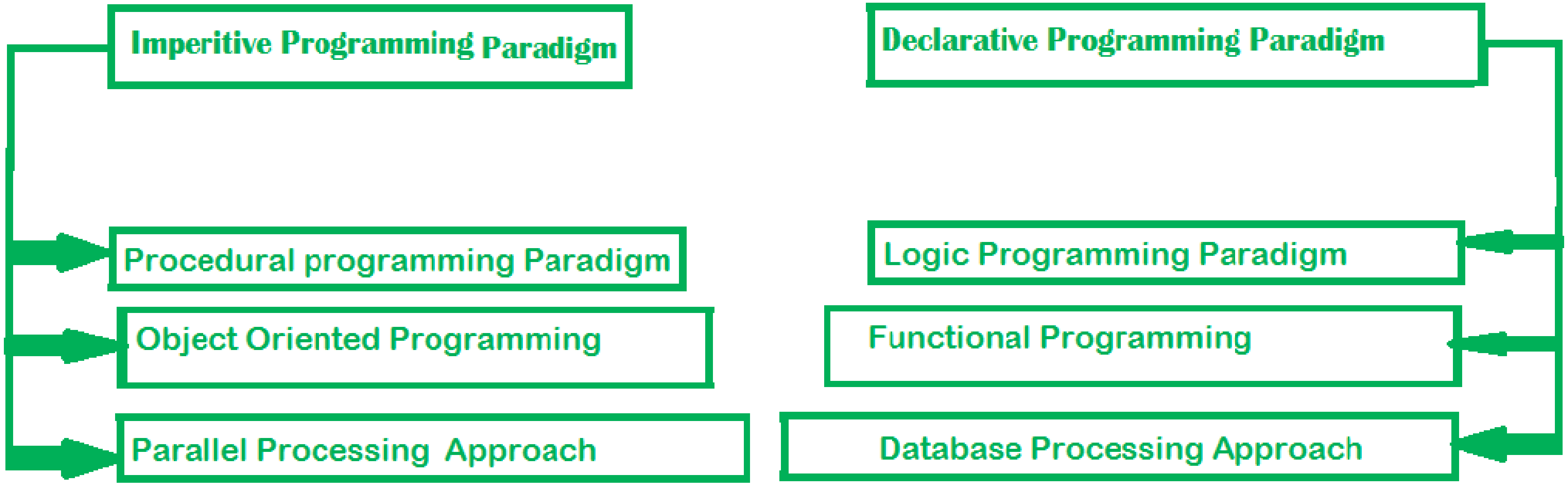
Programming Paradigms

L02 – Session 04

Programming Paradigms

- Paradigm can also be termed as a method to solve some problems or do some tasks.
- A programming paradigm is an approach to solve the problem using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach.
- There are lots of programming languages that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is paradigms.
- Apart from varieties of programming languages, there are lots of paradigms to fulfill each and every demand.

Programming Paradigms



Imperitive Programming Paradigm

Procedural programming Paradigm

Object Oriented Programming

Parallel Processing Approach

Declarative Programming Paradigm

Logic Programming Paradigm

Functional Programming

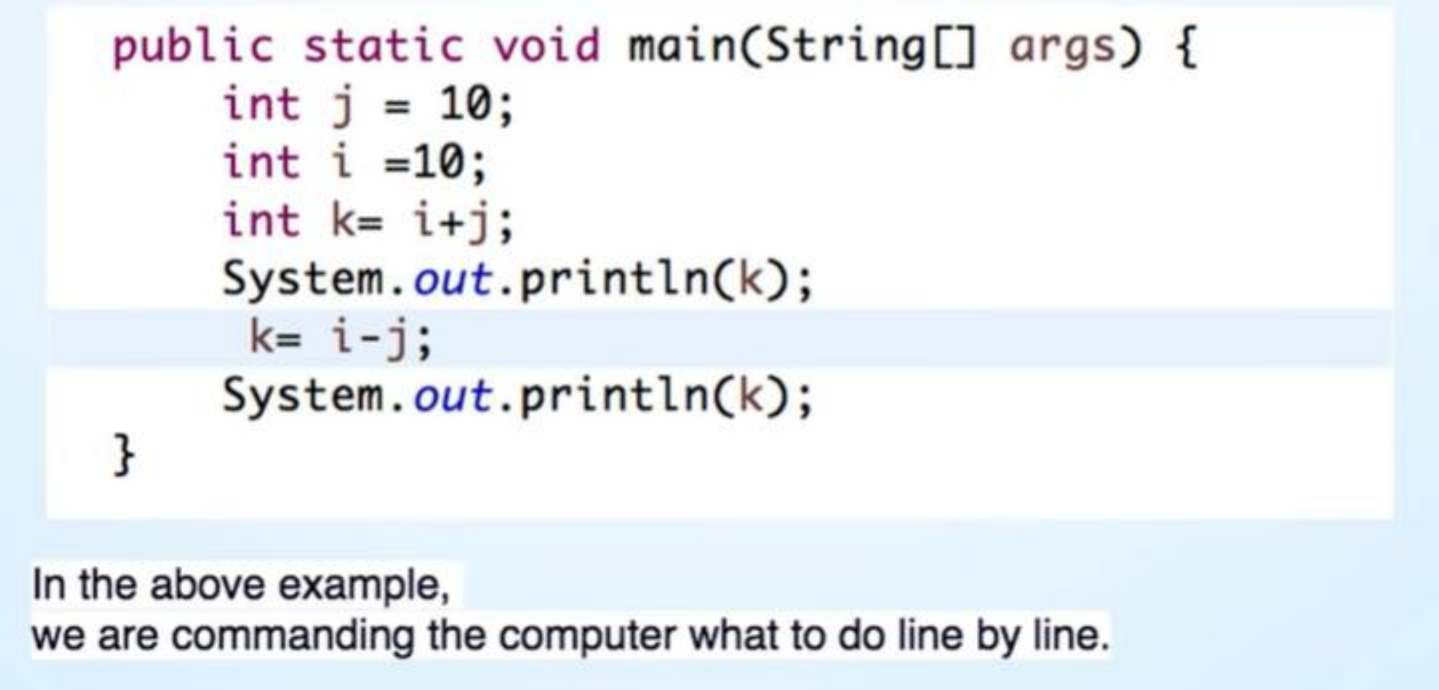
Database Processing Approach

Python supports three types of Programming Paradigms

- Object Oriented programming paradigms
- Procedure Oriented programming paradigms
- Functional programming paradigms

Imperative Programming

- Imperative programming tells the computer to do the task by giving a set of command or instruction in a particular order. Programmer has to explicitly defines the order of the execution of the instruction



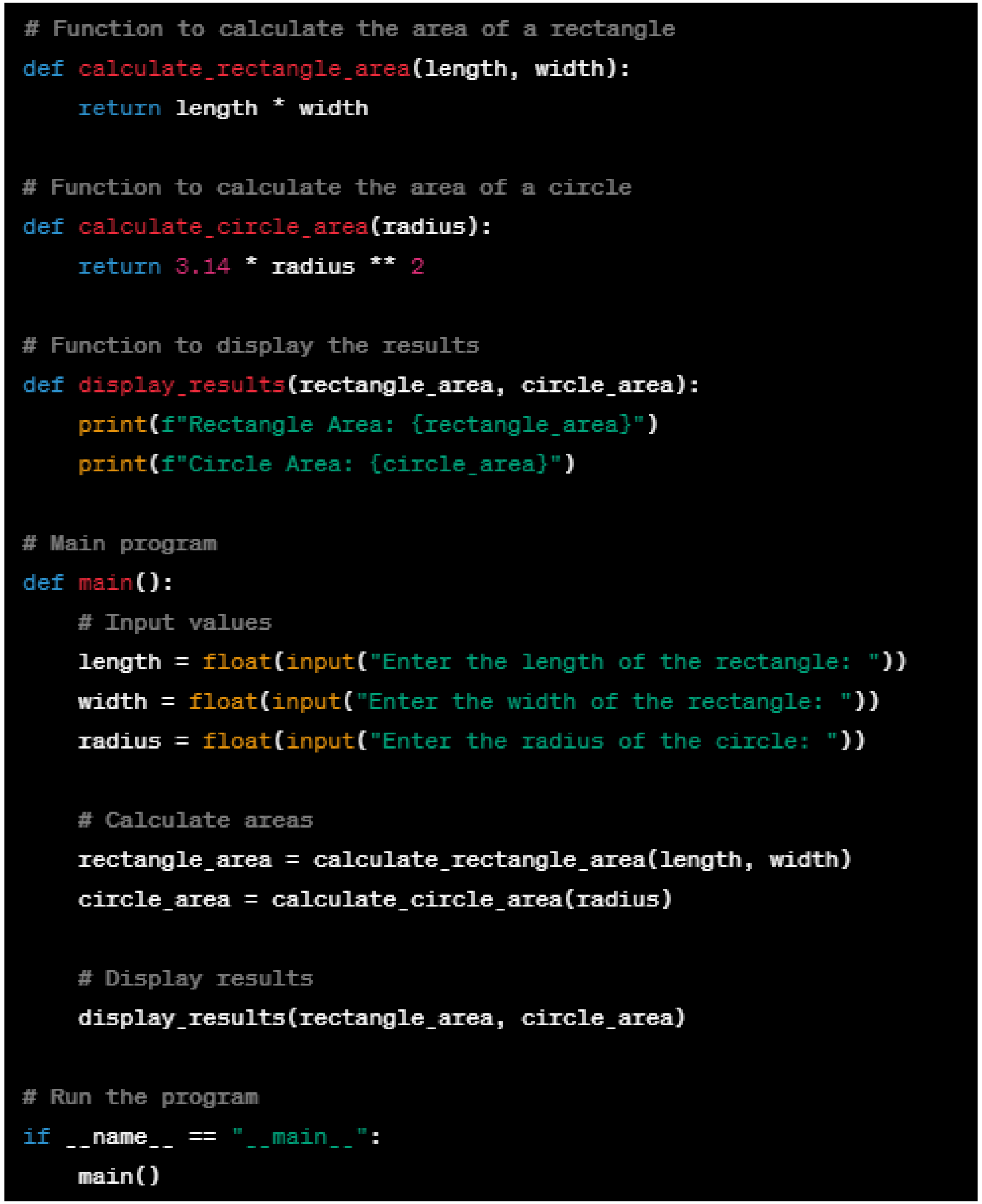
```
public static void main(String[] args) {
    int j = 10;
    int i =10;
    int k= i+j;
    System.out.println(k);
     k= i-j;
    System.out.println(k);
}
```

In the above example,
we are commanding the computer what to do line by line.

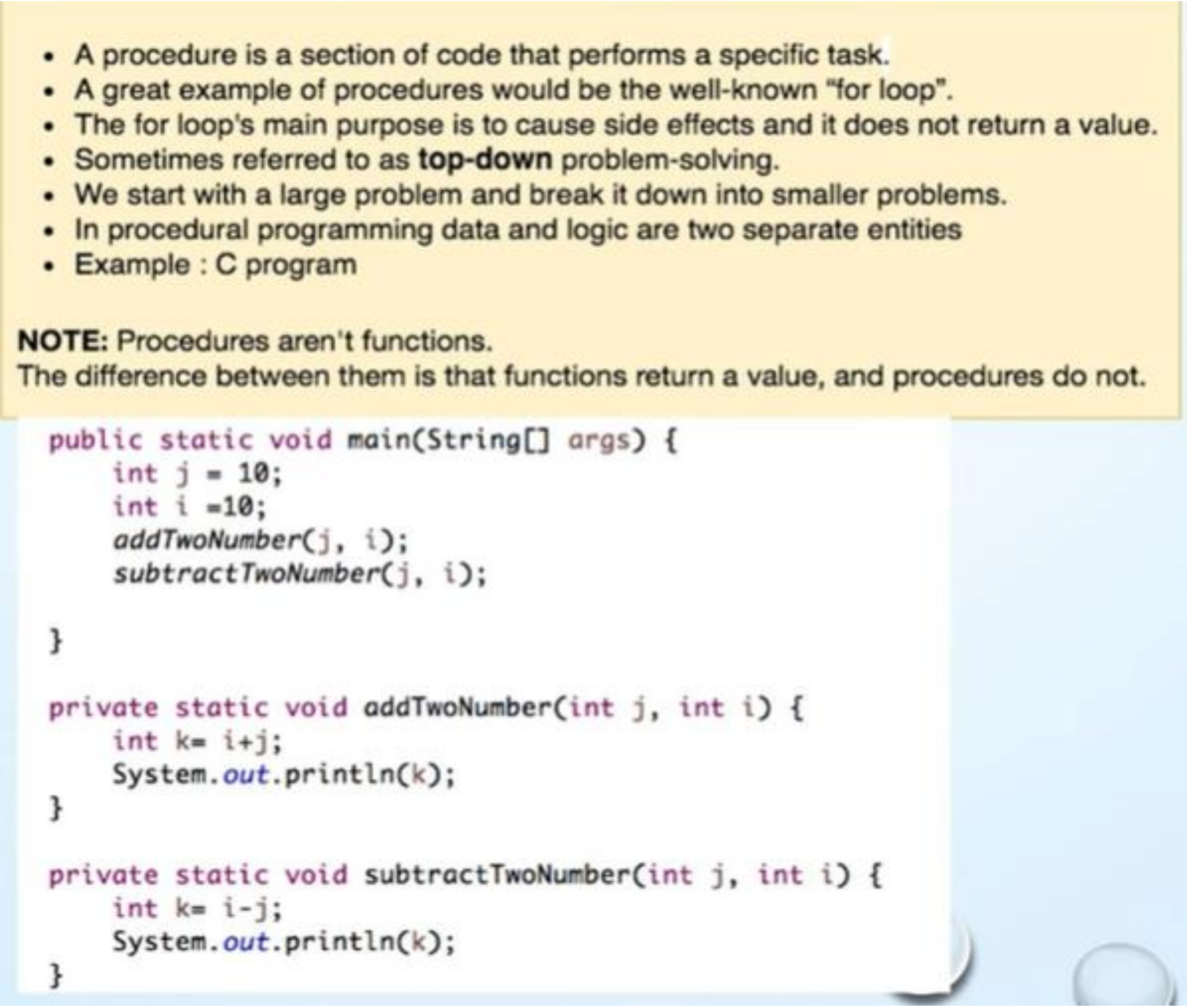
Procedural programming paradigms

In Procedure Oriented programming paradigms, series of computational steps are divided modules which means that the code is grouped in functions and the code is serially executed step by step so basically, it combines the serial code to instruct a computer with each step to perform a certain task. This paradigm helps in the modularity of code and modularization is usually done by the functional implementation. This programming paradigm helps in an easy organization related items without difficulty and so each file acts as a container

- **Procedural Programming** is a programming language that follows a step-by-step approach to break down a task into a collection of variables and routines (or subroutines) through a sequence of instructions. In procedural oriented programming, each step is executed in a systematic manner so that the computer can understand what to do.
- The programming model of the procedural oriented programming is derived from structural programming. The concept followed in the procedural oriented programming is called the "procedure". These procedures consist several computational steps that are carried out during the execution of a program. Examples of procedural oriented programming language include – **C**, **Pascal**, ALGOL, **COBOL**, BASIC, etc



```
# Function to calculate the area of a rectangle
def calculate_rectangle_area(length, width):
    return length * width

# Function to calculate the area of a circle
def calculate_circle_area(radius):
    return 3.14 * radius ** 2

# Function to display the results
def display_results(rectangle_area, circle_area):
    print(f"Rectangle Area: {rectangle_area}")
    print(f"Circle Area: {circle_area}")

# Main program
def main():
    # Input values
    length = float(input("Enter the length of the rectangle: "))
    width = float(input("Enter the width of the rectangle: "))
    radius = float(input("Enter the radius of the circle: "))

    # Calculate areas
    rectangle_area = calculate_rectangle_area(length, width)
    circle_area = calculate_circle_area(radius)

    # Display results
    display_results(rectangle_area, circle_area)

# Run the program
if __name__ == "__main__":
    main()
```



- A procedure is a section of code that performs a specific task.
- A great example of procedures would be the well-known “for loop”.
- The for loop's main purpose is to cause side effects and it does not return a value.
- Sometimes referred to as **top-down** problem-solving.
- We start with a large problem and break it down into smaller problems.
- In procedural programming data and logic are two separate entities
- Example : C program

NOTE: Procedures aren't functions.
The difference between them is that functions return a value, and procedures do not.

```
public static void main(String[] args) {
    int j = 10;
    int i =10;
    addTwoNumber(j, i);
    subtractTwoNumber(j, i);

}

private static void addTwoNumber(int j, int i) {
    int k= i+j;
    System.out.println(k);
}

private static void subtractTwoNumber(int j, int i) {
    int k= i-j;
    System.out.println(k);
}
```

Procedural programming paradigms

Advantages

- General-purpose programming
- Code reusability
- Portable source code

Disadvantages

- Data protection
- Not suitable for real-world objects
- Harder to write

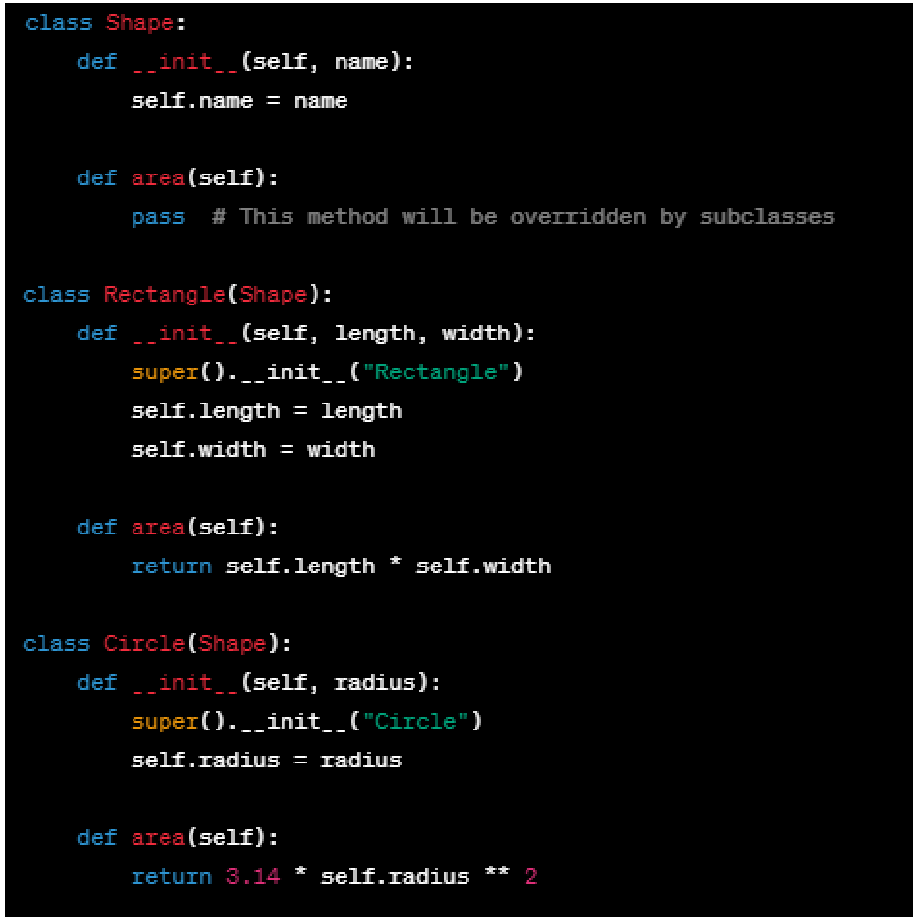
Object Oriented programming paradigms

In the object-oriented programming paradigm, objects are the key element of paradigms. Objects can simply be defined as the instance of a class that contains both data members and the method functions.

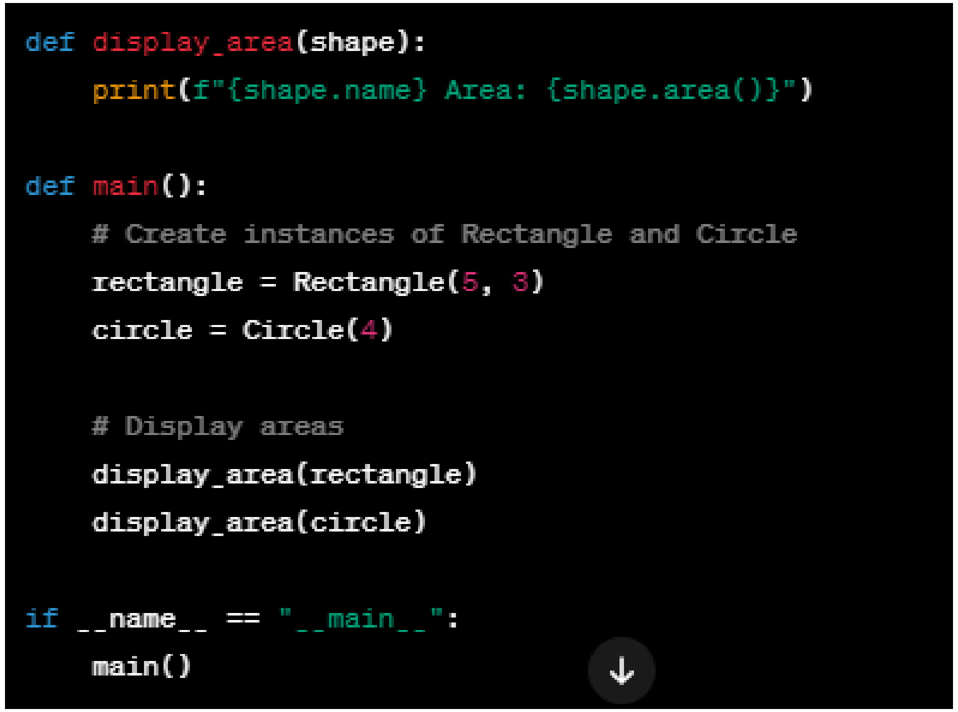
Moreover, the object-oriented style relates data members and methods functions that support encapsulation and with the help of the concept of an inheritance, the code can be easily reusable but the major disadvantage of object-oriented programming paradigm is that if the code is not written properly then the program becomes a monster.

Object-oriented Programming is a programming language that uses classes and objects to create models based on the real world environment. These objects contain data in the form of attributes and program codes in the form of methods or functions. In OOP, the computer programs are designed by using the concept of objects that can interact with the real world entities.

We have several types of object oriented programming languages, but the most popular is one among all is class-based language. In the class-based OOP languages, the objects are the instances of the classes that determine their types. Examples of some object oriented programming languages are – **Java**, **C++**, **C#**, **Python**, **PHP**, **Swift**, etc.

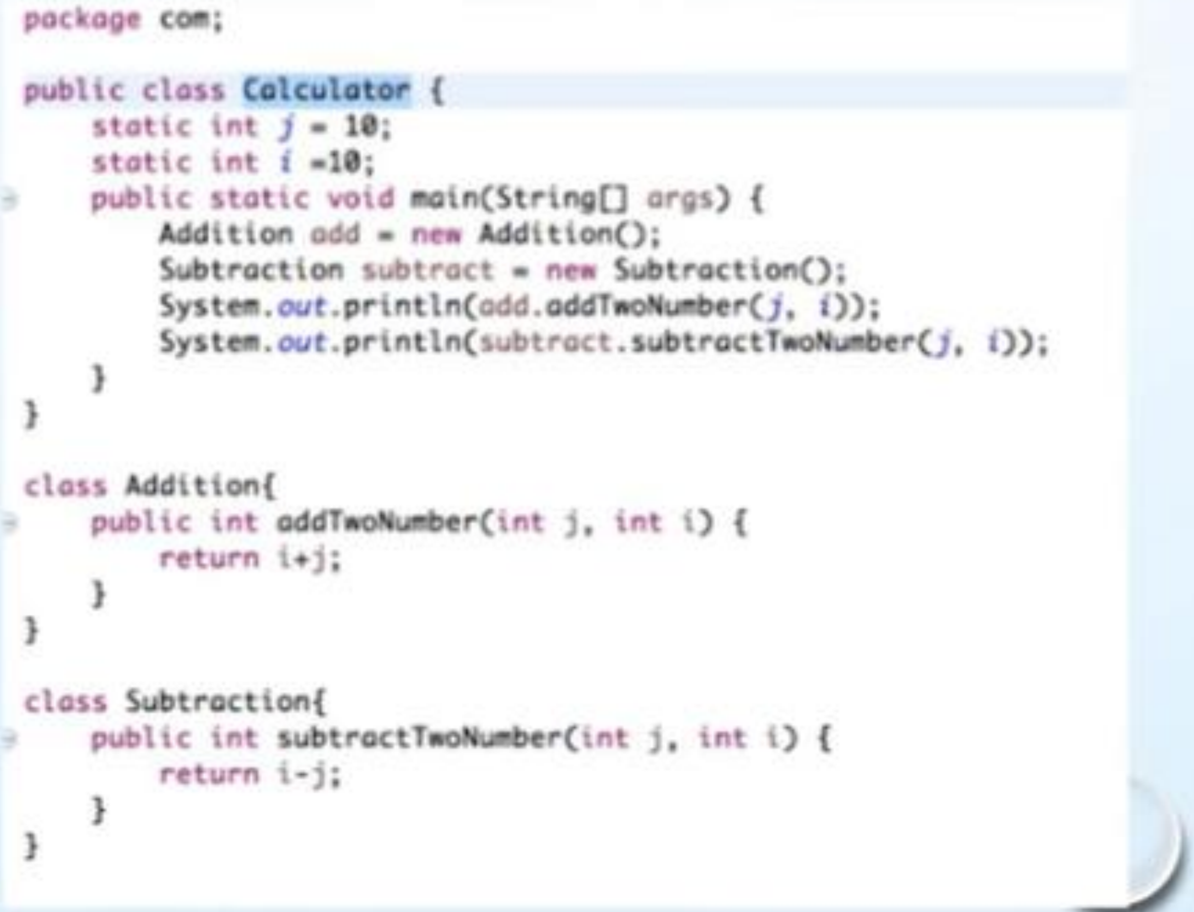


```python
class Shape:
    def __init__(self, name):
        self.name = name

    def area(self):
        pass  # This method will be overridden by subclasses

class Rectangle(Shape):
    def __init__(self, length, width):
        super().__init__("Rectangle")
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

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

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



```python
def display_area(shape):
    print(f"{shape.name} Area: {shape.area()}")

def main():
    # Create instances of Rectangle and Circle
    rectangle = Rectangle(5, 3)
    circle = Circle(4)

    # Display areas
    display_area(rectangle)
    display_area(circle)

if __name__ == "__main__":
    main()
```

- Objects are equals to real-life entities.
- Objects will have Properties and behaviors.
- Classes are blueprints for the objects.
- Classes define the object Properties and behaviors.
- Example : Java



```java
package com;

public class Calculator {
    static int j = 10;
    static int i =10;
    public static void main(String[] args) {
        Addition add = new Addition();
        Subtraction subtract = new Subtraction();
        System.out.println(add.addTwoNumber(j, i));
        System.out.println(subtract.subtractTwoNumber(j, i));
    }
}

class Addition{
    public int addTwoNumber(int j, int i) {
        return i+j;
    }
}

class Subtraction{
    public int subtractTwoNumber(int j, int i) {
        return i-j;
    }
}
```

Object Oriented programming paradigms

Advantages

- Relation with Real world entities
- Code reusability
- Abstraction or data hiding

Disadvantages

- Data protection
- Not suitable for all types of problems
- Slow Speed

Differences between Procedural and Object Oriented Programming

Object Oriented Programming

- Object-oriented Programming is a programming language that uses classes and objects to create models based on the real world environment. In OOPs, it makes it easy to maintain and modify existing code as new objects are created inheriting characteristics from existing ones.
- In OOPs concept of objects and classes is introduced and hence the program is divided into small chunks called objects which are instances of classes.
- C++, C#, Java, Python, etc. are the examples of OOP languages.

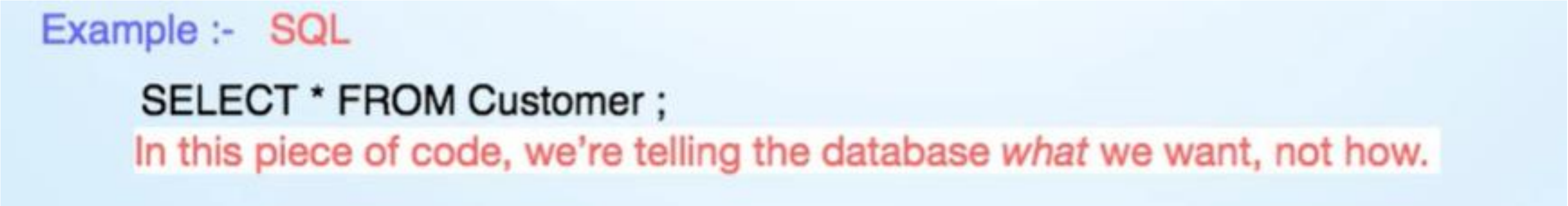
Procedural Programming

- Procedural Programming is a programming language that follows a step-by-step approach to break down a task into a collection of variables and routines (or subroutines) through a sequence of instructions. Each step is carried out in order in a systematic manner so that a computer can understand what to do.
- In procedural programming, the main program is divided into small parts based on the functions and is treated as separate program for individual smaller program.
- C, BASIC, COBOL, Pascal, etc. are the examples POP languages.

Declarative Programming

Programmer defines what need to be accomplished by the program

1. Tell the computer what to do rather than how to do
2. Program describe the desired results
3. It has two categories
   - Logic
   - Functional

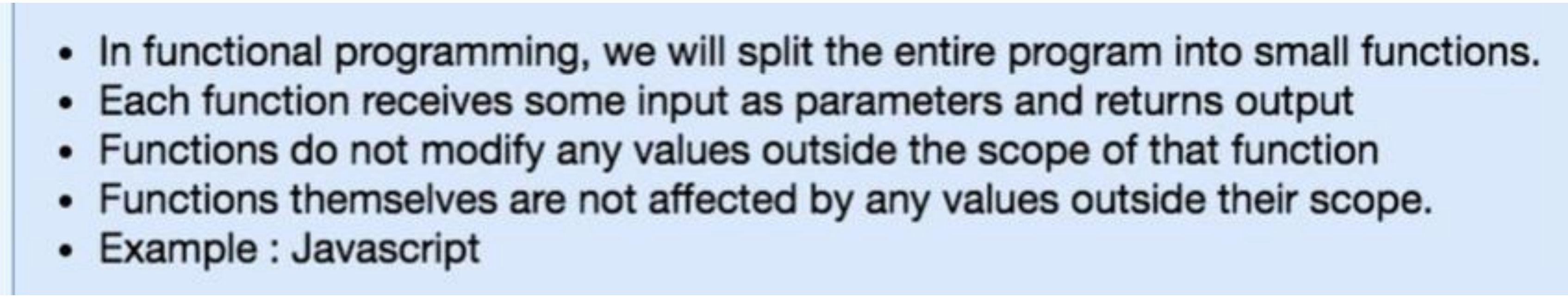


Example :- SQL

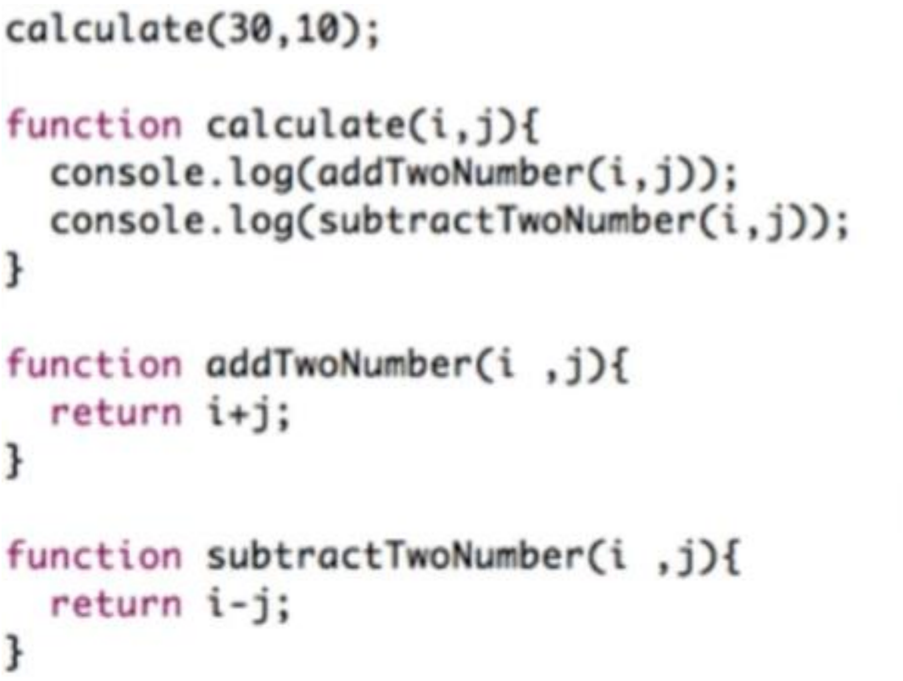
```
SELECT * FROM Customer ;
```

In this piece of code, we're telling the database *what* we want, not how.

Functional Programming



- In functional programming, we will split the entire program into small functions.
- Each function receives some input as parameters and returns output
- Functions do not modify any values outside the scope of that function
- Functions themselves are not affected by any values outside their scope.
- Example : Javascript



```
calculate(30,10);

function calculate(i,j){
  console.log(addTwoNumber(i,j));
  console.log(subtractTwoNumber(i,j));
}

function addTwoNumber(i ,j){
  return i+j;
}

function subtractTwoNumber(i ,j){
  return i-j;
}
```

Functional programming paradigms

Functional programming paradigms is a paradigm in which everything is bind in pure mathematical functions style. It is known as declarative paradigms because it uses declarations overstatements.

It uses the mathematical function and treats every statement as functional expression as an expression is executed to produce a value. Lambda functions or Recursion are basic approaches used for its implementation.

The paradigms mainly focus on “what to solve” rather than “how to solve”. The ability to treat functions as values and pass them as an argument make the code more readable and understandable

Functional programming paradigms

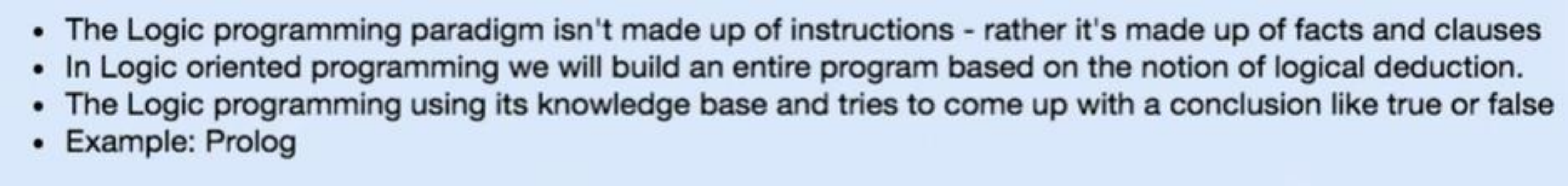
Advantages

- Simple to understand
- Making debugging and testing easier
- Enhances the comprehension and readability of the code

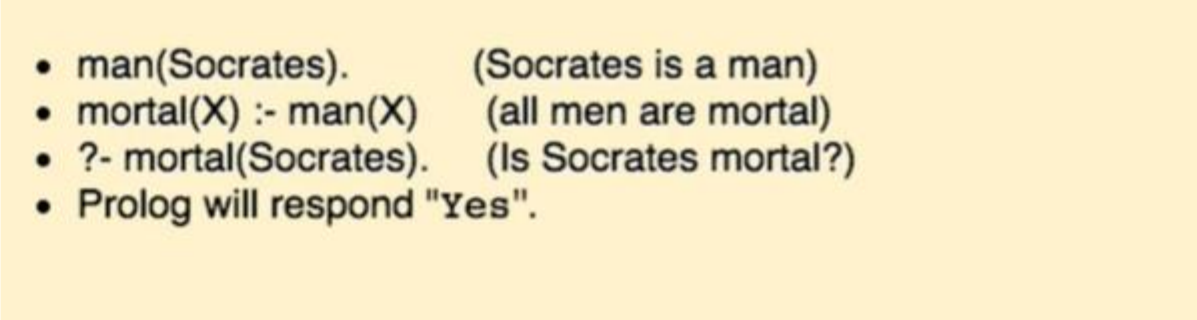
Disadvantages

- Low performance
- Writing programs is a daunting task
- Low readability of the code

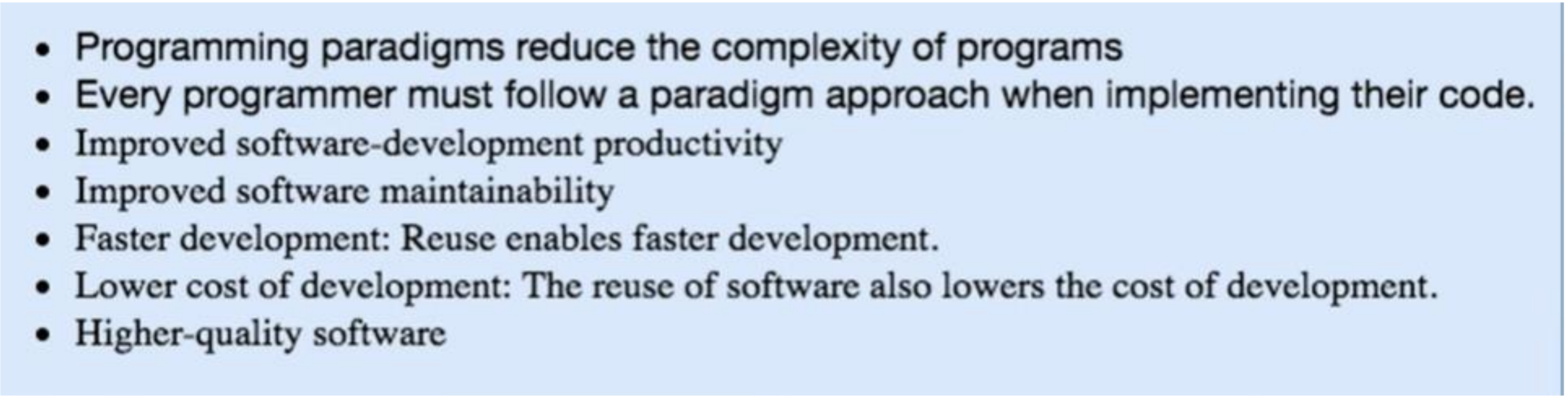
Logic Programming



- The Logic programming paradigm isn't made up of instructions - rather it's made up of facts and clauses
- In Logic oriented programming we will build an entire program based on the notion of logical deduction.
- The Logic programming using its knowledge base and tries to come up with a conclusion like true or false
- Example: Prolog



- man(Socrates).   (Socrates is a man)
- mortal(X) :- man(X)   (all men are mortal)
- ?- mortal(Socrates).   (Is Socrates mortal?)
- Prolog will respond "Yes".



- Programming paradigms reduce the complexity of programs
- Every programmer must follow a paradigm approach when implementing their code.
- Improved software-development productivity
- Improved software maintainability
- Faster development: Reuse enables faster development.
- Lower cost of development: The reuse of software also lowers the cost of development.
- Higher-quality software