Chapter 1: Introduction to object-oriented programming

Department of Computer Science and Engineering
Kathmandu University

Rajani Chulyadyo, PhD

Programming Paradigm

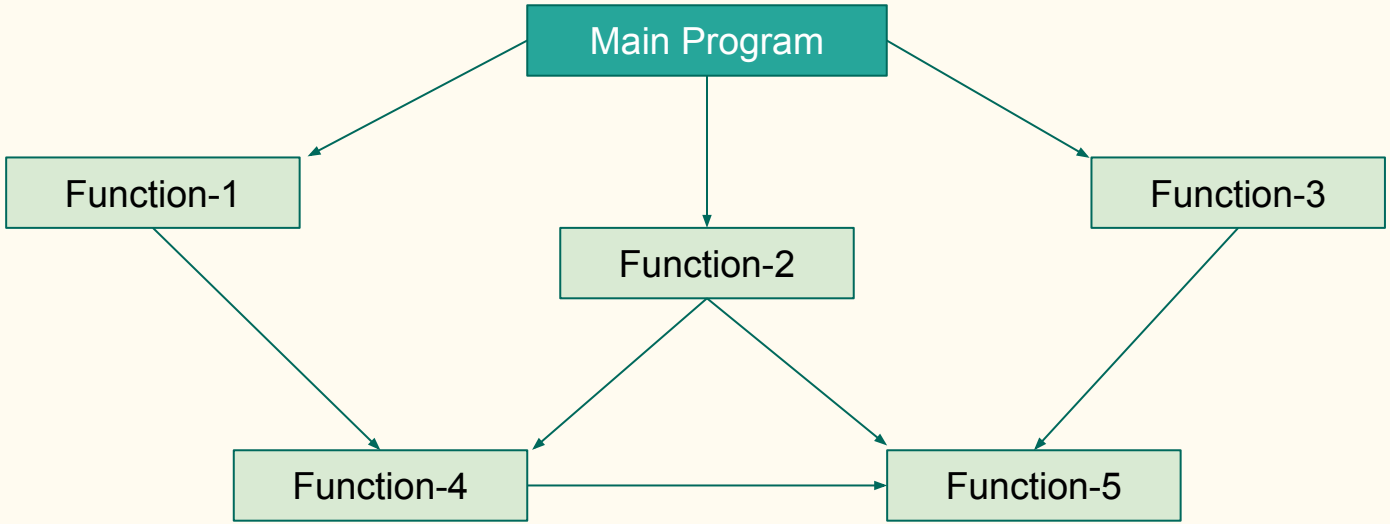
A way to classify programming languages based on their features.

Common programming paradigms include

- Imperative programming
  - The programmer specifies exactly how to do something, not just the desired outcome.
  - Procedural programming (C, C++ etc.) / object-oriented programming (C++, Java etc.)
- Declarative programming
  - The programmer declares what needs to happen, not how it's done.
  - SQL, HTML, Prolog, Common LISP, Clojure etc.

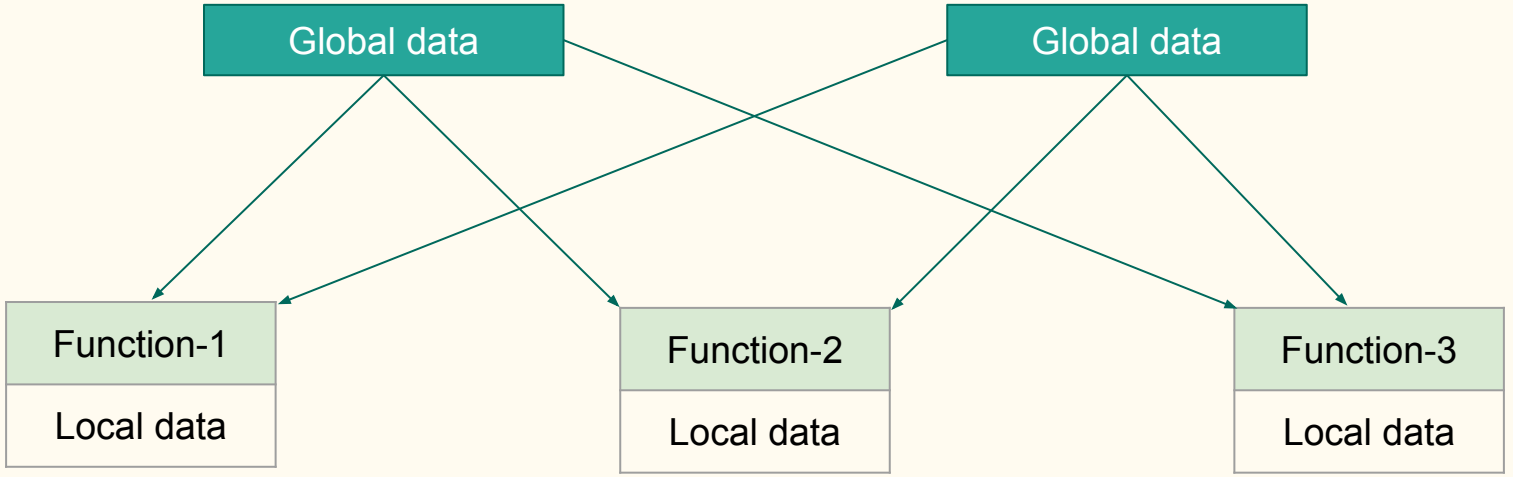
Procedural programming

In “procedural” programming, problems are “decomposed” into smaller units of repeatable activities called *procedures*.



Procedural programming

In “procedural” programming, problems are “decomposed” into smaller units of repeatable activities called *procedures*.



Procedural programming

Disadvantages

- Focus on processes
  - Importance is given to the operation on data rather than the data.
- No information hiding
  - Data is exposed to the whole program.
- Difficult to manage large programs
  - Imagine changing the type of a global variable from `int` to `long` in a large program!!
- Difficult to relate with real-world objects
  - In physical world, we deal with objects such as people, cars etc. Such objects have both attributes (data) and behavior (functions).
  - Neither data nor functions, by themselves, model real-world objects effectively.

Object-Oriented Programming (OOP)

- Relies on the concept of **objects** and **classes.**
- The fundamental idea:
  - Combine into a single unit both data (attributes) and the functions that operate on that data (functionality/behavior).
- In this paradigm, a computer program is built with a collection of reusable components called **objects.**
- Related piece of information and behaviours are organized together into a template called a **class.**
- A class is thus a description of a number of similar objects.
- **An object is an instance of a class.**

Procedural vs Object-oriented programming

Procedural	**OOP**
Focuses on operations	Focuses on objects
Follows top-down approach (General to specific)	Follows bottom-up approach (Specific to general)

Object-Oriented Programming

Suppose you want to build a library management system.

- In procedural programming, you decompose the system by functions or processes, such as add resources, record loans, report fines etc.
- In OOP, you decompose the system by objects or concepts, such as student, book, catalog, library, librarian etc.
  - Here, student may be a class, and John, and Alice, who are students, are objects.
  - Student class may contain data, such as name, address, the number of books borrowed etc., and behaviors, such as borrow a book, return a book (which will update the number of books borrowed information).
  - Similarly other objects (books, catalogs etc.) will also contain data and behaviors.

What kinds of things become objects in object-oriented programs?

Some examples:

- Human entities
  - Employees, Students, Customers etc.
- Physical objects
  - Cars, planes, buses, countries, house, electrical components, food items (pizza, noodles) etc.
- Elements of the computer-user environment
  - Windows, menus, graphic objects (lines, rectangles, circles), mouse, keyboard etc.
- Data-storage constructs
  - Customized arrays, stacks, linked lists, binary trees etc.
- User-defined data types
  - Time, angles, complex numbers etc.
- etc.

Principles of object-oriented programming

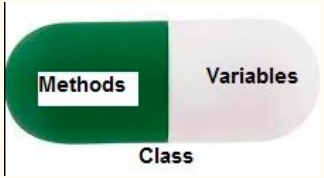
- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Principles of OOP: Encapsulation

Data encapsulation and *data hiding* are among the key concepts in OOP.

Data encapsulation:

- Bundling up of **data** with the **methods** that operate on that data.



source:https://www.geeksforgeeks.org/encapsulation-in-c/

Data hiding:

- Restricting direct access to the data (so that it is safe from accidental alteration).
- Only those functions which are wrapped in the class can access it.
- This simplifies writing, debugging, and maintaining the program.

Principles of OOP: Encapsulation

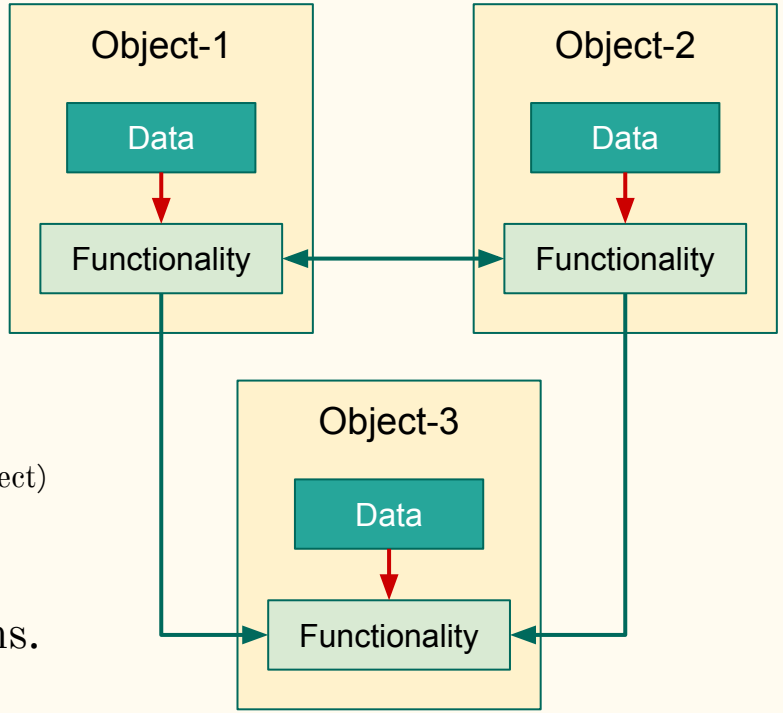
Data hiding:

Data of an object can be accessed only by the function associated with that object.

However, function of an object can access the functions of other objects.

(Calling an object's function is referred to as *sending a message* to the object)

An object cannot modify the data of another object directly but can do so indirectly through the functions.



Principles of OOP: Abstraction

Providing only essential information to the outside world and hiding their background details.

Its main goal is to handle complexity by hiding unnecessary details from the user.

Examples:

- While using mobile phone, we know what happens when we click on home button, back button and many more. But unknown about internal working mechanism.
- Your program can make a call to the sort() function without knowing which algorithm the function actually uses to sort the given values.

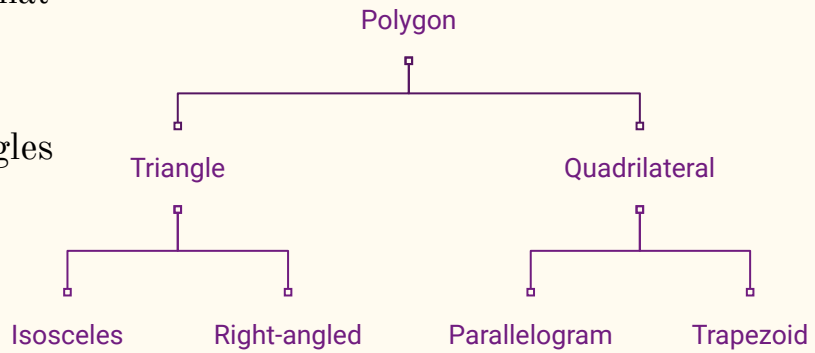
Principles of OOP: Abstraction

Abstraction separates code into **interface** and **implementation**.

Interface should be independent of the implementation so that if the underlying implementation is changed, the interface would remain intact.

Principles of OOP: Inheritance

- Capability of a class to derive **properties** and **characteristics** from another class.
- Helps to add the additional features in an existing class without modifying it.
- Supports the concept of hierarchical classification.
- Example:
  - Suppose we have a class called Polygon.
  - Triangle and Quadrilateral are two classes that inherit some properties of polygon, and add some properties of their own.
  - Similarly, Isosceles, and Right-angled triangles are triangles with special properties.



Principles of OOP: Polymorphism

- Poly = many
- Morph = forms
- Polymorphism: Ability to take more than one form. Processing objects differently based on their data type.
- Examples:
  - + operator will generate sum when two operands are numbers but performs string concatenation and produces third string if the operands are strings.
  - In a car with a gear transmission system, when the engine is accelerated then depending upon which gear is engaged different amount power and movement is delivered to the car.

Benefits of OOP

Some potential benefits are

- Reusability
  - Once a class has been written, created, and debugged, it can be distributed to other programmers for use in their own programs.
- Robustness
  - Most object-oriented languages support exception and error handling
- Extensibility
  - A programmer can take an existing class and, without modifying it, add additional features and capabilities to it (thanks to the inheritance).
- Easier to manage
  - Each object is relatively small, self-contained, and manageable, thus reducing the complexity of the program.