**Chapter 1. Abstract Data Type**

**Algorithm**: a sequence of clear and precise step-by-step instructions for solving a problem in a finite amount of time.

**Computer Programming**: the process of translating the algorithm into a computer program that can be executed by a computer.

**Type**: a collection of values

**Data type**: a given type along with a collection of operations for manipulating values of the given type.

**Primitives**: data types that are provided as part of the language itself.

1. **Simple data types**: consist of values that are in the most basic form and cannot be decomposed into smaller parts. (Integer and real types)
2. **Complex data types**: constructed of multiple components consisting of simple types or other complex types. (Strings, lists, and dictionaries)
3. User-defined types

Abstractions

An **abstraction** is a mechanism for separating the properties of an object and restricting the focus to those relevant in the current context.

1. **Procedural abstraction**: the use of a function or method knowing what it does but ignoring how it’s accomplished. (mathematical square root)
2. **Data abstraction**: the separation of the properties of a data type (its values and operations) from the implementation of that data type.

Abstract Data Types

An **abstract data type** (or **ADT**) is a programmer-defined data type that specifies a set of data values and a collection of well-defined operations that can be performed on those values.

-Separates the definition from the implementation

-Requires interface

-Information hiding

-User programs interact with instances of the ADT by invoking one of the several operations defined by its interface.

1) **Constructors**: create and initialize new instances of the ADT

2) **Accessors**: Return data contained in an instance without modifying it

3) **Mutators**: Modify the contents of an ADT instance

4) **Iterators**: process individual data components sequentially

**Data Structure**: the physical representation of how data is organized and manipulated

* Can be characterized by how they store and organize the individual data elements and what operations are available for accessing and manipulating the data.
* Eg. Arrays, linked lists, stacks, queues, and trees