Notes on Reading

**Data Structures and Algorithms**

**Using Python**

**By Jizhen Zhao**

**March, 2016**

**The Book**

|  |  |
| --- | --- |
| Title | Data Structures and Algorithms Using Python |
| Author | [Rance D. Necaise](http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470618299,descCd-authorInfo.html) |
| Publisher | John Wiley & Sons |
| Pub. Date | December 21, 2010 |
| Print ISBN | 978-0-470-61829-5 |
| Web ISBN | 0-470618-29-9 |
| Pages in  Print Edition | 538 |

[Chapter 1. Abstract Data Types 4](#_Toc444729614)

[1.1. Introduction 4](#_Toc444729615)

[1.2. The Date Abstract Data Type 5](#_Toc444729616)

[1.3. Bags 5](#_Toc444729617)

### Chapter 1. Abstract Data Types

1. The foundation of computer science is based on the study of algorithms.
2. An ***algorithm*** is a sequence of clear and precise step-by-step instructions for solving a problem in a finite amount of time.
3. Algorithms are implemented by translating the step-by-step instructions into a ***computer program*** that can be executed by a computer.
4. This translation process is called ***computer programming*** or simply ***programming***.
5. Computer programs are constructed using a ***programming language*** appropriate to the problem.
6. While programming is an important part of computer science, computer science is not the study of programming.
7. Nor is it about learning a particular programming language.
8. Instead, programming and programming languages are tools used by computer scientists to solve problems.

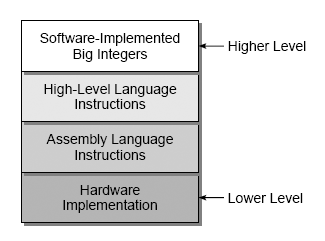
#### 1.1. Introduction

**Data Types:**

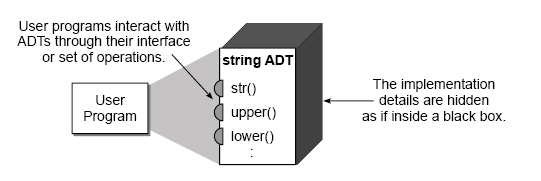
1. Data items are represented within a computer as a sequence of **binary** digits.
2. They can be differentiated as different types:
   1. **Type**: often refer a collection of values
   2. **Data Type**: often refer to a type and its associated operations
3. Programming languages commonly provide data types by itself, known as **primitives**, which can fall into two categories:
   1. **Simple data types**: e.g. integer, and other numeric types
   2. **Complex data types**: e.g. string, tuple, list dictionary in Python
4. Most programming languages also allow **user-defined types** to be created.

**Abstractions:**

1. Used to manage complex problems and complex data types.
2. 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** is the use of a function or method knowing what it does but ignoring **how** it's accomplished.
   2. **Data abstraction** is the separation of the properties of a data type (its values and operations) from the **implementation** of that data type.
3. Levels of abstractions:



1. 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.
   1. Defined independent of their implementations, through an **interface** (a set of operations) – **information hiding**.



* 1. The categories of the operations:
     1. Constructors
     2. Accessors
     3. Mutators
     4. Iterators

1. A **data structure**, which is the physical representation of how data is organized and manipulated, can be used to implement a complex **ADT**.
   1. A **collection** is a group of values with no implied organization or relationship between the individual values.
   2. A **container** is any data structure or abstract data type that stores and organizes a collection.
   3. A **sequence** is a container in which the elements are arranged in linear order from front to back, with each element accessible by position.
   4. A **sorted sequence** is one in which the position of the elements is based on a prescribed relationship between each element and its successor.

#### 1.2. The Date Abstract Data Type

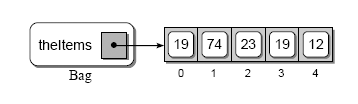
Skip.

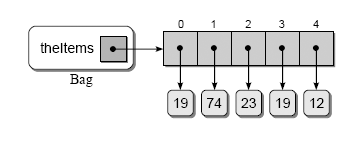
#### 1.3. Bags

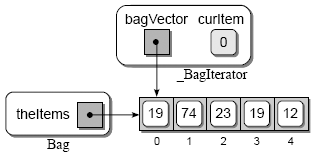
1. A **bag** is a container that stores a collection in which **duplicate** values are allowed. The items, each of which is individually stored, have no particular order but they must be comparable.

* Bag():
* length ():
* contains( item ):
* add( item ):
* remove( item ):
* iterator ():

1. Implement using Python list:







# Chapter 10 Recursion

**def** printRev( n ):

**if** n > 0 :

**print**( n )

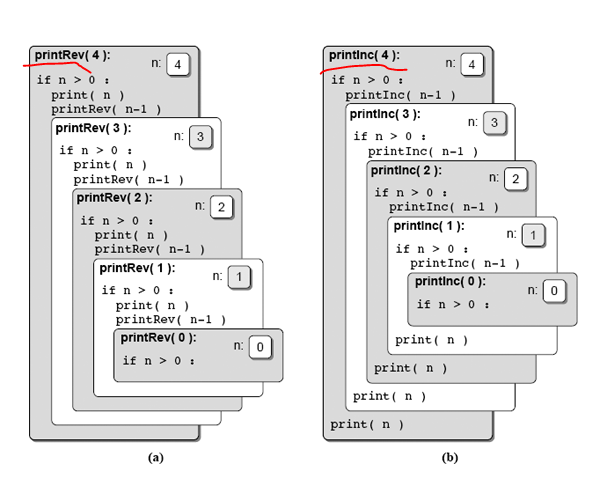
printRev( n-1 )

**def** printInc( n ):

**if** n > θ :

printInc( n− 1 )

**print**( n )



***Fibonacci sequence***

