CS 234

Module 1

September 5, 2018

Introduction

Course goal: Solve problems using complex manipulation of data.

Topic	**First-year CS**	**Beyond first year**
Code	From scratch by one person	From multiple sources integrated together
Design recipe	Plan/code separation	Plan/code and user/provider separation
Data	Built-in ways to structure data	Complex ways of manipulating data
Resources	Introduction to efficient use of time	Analysis of time use

Main ideas

Modularity is the division of a program into independent, reusable pieces.
Data hiding is the protection of data from other parts of the program.

Modularity and data hiding in CS 234

An **abstract data type (ADT)** specifies a set of data items and operations on those items. For each operation, preconditions and postconditions are given.
A **precondition** is a requirement that must be satisfied for an operation to be guaranteed to work.
A **postcondition** is a guarantee of the outcome of an operation being executed.
An ADT does not include details of how items are stored or how the operations are implemented.

Note: In contrast, a **data type** is a data storage format for a programming language, such as a string.

View of ADTs by users and providers

ADTs are like contracts or interfaces between users and providers:

- The user ensures the preconditions are satisfied when an operation is used.
- The provider ensures the postconditions are satisfied in the implementation of the operation.
- The user of an ADT does not need to know details of how the ADT is implemented.
- The provider of an ADT does not need know details of how the ADT is used.

An ADT allows:

- division of labour between user and provider;
- separation of manipulation of data from the rest of the solution;
- code reuse (many problems use the same common ADTs); and
- updates to implementation without requiring the user to make changes.

Stages

Stages for both user and provider:

- Planning
- Coding

Planning is done on paper, using notation known as **pseudocode**, a way of describing an algorithm without specifying a particular machine or programming language.

The provider figures out how to store, access, and modify the data by choosing the data structure and algorithms.

A **data structure** is a way of storing data items, with each operation implemented by an **algorithm**.

- More than one possible algorithm may be possible for a particular operation on a particular data structure.
- A data structure may in turn be defined in terms of other ADTs.

Viewpoints/roles

	User	**Provider**
Plan	Agreement on ADT	
	Pseudocode using ADT	*ADT use unknown*
	Data structure unknown	Pseudocode for data structure
	Algorithms unknown	Pseudocode for operations
Code	Agreement on code interface	
	Code using ADT	*ADT use unknown*
	Data structure unknown	Code for data structure
	Algorithms unknown	Code for operations

Recipes for planning

Solving a problem (user/plan):

1. Determine types of data and operations.
2. For each type, choose/customize an ADT.
3. Develop a pseudocode algorithm using ADT operations.
4. Calculate the algorithm's cost with respect to costs of operations.
5. Using information from the provider, choose the best option.

Choosing among implementations (provider/plan):

1. Create pseudocode of various options for data structures and algorithms to implement the ADT and its operations.
2. Analyze the cost of each operation for each implementation.
3. Provide options for packages of operation costs.

Recipes for coding

User/code:

1. Agree on the code interface.
2. Code a solution to the problem using the ADT.

Provider/code:

1. Agree on the code interface.
2. Code the chosen data structure and algorithms implementing the ADT.

Course goals

Basic steps:

- [User-plan] Given a problem, determine a type of data and operations.
- [User-plan] Given a type of data and operations, choose an ADT.
- [User-plan] Given a problem, write pseudocode using ADT operations.
- [Provider-plan] Given an ADT, write pseudocode implementing operations using different data structures.
- [User/Provider-plan] Given pseudocode, analyze the worst-case running time.

Related topics:

- Common ADTs for different kinds of data
- Common data structures
- Analysis of pseudocode
- Understanding storage in memory
- Handling data features (general, orderable, "digital"); structure

Challenges in choosing a textbook

Similar terms used in the field and in Python

- Python data type array vs. array data structure
- Python data type dictionary versus ADT Dictionary

Options:

- Make up weird names. (Textbook)
- Be careful to make sure what type of entity the name is referring to. (Course notes)

Using the textbook and other sources

Different sources use different names for ADTs, names for operations, operations, and specifications for operations of the same name.

Issues with the textbook:

- Array implementation (see more detailed explanation)
- Iterators in ADTs
- Operator overloading for names of operations
- Blurring of viewpoints:
  - Analyzing Python instead of pseudocode
  - Using Python operations in planning stage
  - Conflating classes and ADTs
- Check textbook for errata (textbook site) and my comments (linked off of resources page) Notice also *psuedo* and *runtime*.

Bottom line: Use lecture material as source for terminology.

Course logistics

Components:

- Lectures (all slides on-line but **without verbal and on-board explanations**)
- Readings (textbook optional, added resources are not)
- Python self-study
- Self-checks (only in class)
- Assignments
- Exams

Tools used:

- Course website: schedule, resources, news; lecture summaries in advance and some updates after
- Piazza: questions
- MarkUs: hand in assignments; check marks

Work to do this week

- Start on Python review (see Python page on website, and schedule).
- Attend a Python review session, if possible, or use the resources on the Python page.
- Read Python 2 versus Python 3 page if you studied Python a while ago.
- Sign up for Piazza.