Lesson Plans
Mohammad Taha Khan

Course: CS 100 Introduction to Computer Science and Programming (3 credit hours)
Class Timings: T-R 2:3:15 pm. Recitations on Friday (multiple sections)
Instructor: Mohammad Taha Khan

Note: This is a sample one week lesson plan developed by Mohammad Taha Khan based the syllabus of MIT’s OCW course 6.00 Introduction to Computer Science and Programming.

Lesson 5 (Tuesday, Week 3)

Introduction
One of the great things about programming in Python is the diversity of specialized data containers and the variety of their built-in methods. The goal of lecture 5 is to provide a comprehensive overview of the widely used data containers in Python with a focus on the versatility and specialized utility of each one. At this point in the course, students are familiar with variables, iterators and functions. Learning data containers in the next step in advancing their knowledge on how to assign, variables to containers, and how to use for and while iterators to perform traversal operations on them. The lecture will also encourage students to map their understanding of previously learned Big(O) complexities (in lesson 3) to these operations and also develop an understanding of mutability in Python and how to move these data containers around in functions.

Content Focus
The content of this lecture is primarily about data containers in Python and their associated concepts. This includes lists, tuples, sets, dictionaries. Along with an introduction to the containers, students will also learn about the methods that are available for each data container. For lists, this will be various list comprehensions, indexing slicing, appending and removing elements. The next part of the class content will emphasize on tuples, sets and dictionaries, how each of them is unique, and what are they best used for. This discussion will be followed by the respective methods of tuples, sets and dictionaries. Once students have conceptualized data containers, the content will shift over to how to perform iterations on these containers and integrating basic methods like searching and modifying, through iteration operations. The final part of the class will discuss mutability in Python, an introduction to passing by reference, and how this affects moving data containers around in Python functions.

Learning Goals
By the end of the lesson students should be able to:

- Recall previous leaned concepts on expressions, objects, complexity, and integrate them with Python data containers to form program statements.
- Discuss and contrast the use cases of lists, tuples and dictionaries.
- Analyze code to differentiate between mutable and immutable elements.
- Implement functions with correct parameter passing of Python data containers.

Class Outline (Total Time: 75 minutes)

1. Introduction, administrivia, and announcements. (5 minutes)
   a. Provide an introduction to the topics being covered in the lecture.
   b. Make announcement for homework 3 being released.

2. Lists and their respective comprehensions (10 minutes)
   a. Introduction to Python lists and list properties.
   b. How to create lists, and populate them using the append() method.
   c. Discuss on more advanced list methods: index(), insert(), pop(), remove() and sort().
   d. Explain list slicing and indexing.
   e. Answer students’ questions related to this module.
Lesson Plans
Mohammad Taha Khan

3. **Tuples, sets and dictionaries (20 minutes)**
   a. Define tuples, and introduce adding pairwise elements to tuples.
   b. Discuss the commonalities and difference between lists.
   c. Introduce sets, creation, and explain how they are store unique elements.
   d. Relate the set operations such as: `union()`, `intersection()`, `difference()` to classical mathematical sets.
   e. Explain O(1) fast searching in sets due to hashed based indexing.
   f. Introduce the concept of unique key-value pairs, which forms the fundamentals of dictionaries.
   g. Explain dictionary creation and the details of `keys()`, `values()` methods to isolate the keys and values.
   h. Summarize other common methods: `get()`, `pop()`, `items()`, `max()`, `min()` and `zip()`.
   i. Answer students’ questions related to this module.

4. **In-class clicker quiz on lists and list comprehensions (10 Minutes)**
   a. Conduct a clicker quiz on that covers the following list-based concepts:
      - Q1 – Syntax of creating a list and appending elements.
      - Q2 – sorting elements in a list.
      - Q3 – Observe a code snippet on list operation and judge if it will throw an exception/error.
   b. Discuss the solutions of the quiz and answer students questions.

5. **Iteration on data containers (10 minutes)**
   a. Explain the use of `for` and `while` loops to iterate over lists, tuples and sets dictionaries.
   b. Elaborate on the concept of auto-indexing while performing iteration.
   c. Emphasize how serialization of dictionaries and how iteration on dictionaries happens over keys.
   d. Demonstrate with code example on how to perform method operations, and use conditionals along with an iterator.
   e. Answer student questions related to this module.

6. **Mutability and Pass by Referencing (10 minutes)**
   a. Define mutability.
   b. Explain which data containers are mutable (lists) and which ones are immutable (sets).
   c. Explain what is passing by reference, and how Python does not copy elements when they are passed.
   d. Show faulty code that improperly uses pass by referencing, causing incorrect output.
   e. Provide an explanatory solution to how to address this built-in Python feature, and by efficient programming or alternatively using the copy constructor.
   f. Answer student questions related to this module.

7. **Programming Activity (10 Minutes)**
   a. Break student into discussion groups.
   b. Provide a programming activity with the following required deliverables:
      - Read a list of names and phone numbers from a CSV file.
      - Store the names as keys and phone numbers as values in a dictionary.
      - Iterate over the dictionary to search for phone numbers with Chicago area code.
      - Write a function that dictionary keys into a sorted list.
   c. Discuss the solutions of the programming activity and answer students’ questions.

**Required Class Materials**
1. Laptop with PowerPoint, and required adapters
2. Projector setup
3. iClicker device to collect responses
4. Students required to bring iClicker devices
Lesson Plans
Mohammad Taha Khan

5. Whiteboard and markers
6. Class slide-deck handouts
7. Classroom design which allows students to feasibly converse in groups without major disruption

Preparation Materials
- Pages 49 - 69 of Introduction to Computation and Programming Using Python by John V. Guttag.
- Supplemental online materials:
  - https://docs.python.org/3.4/tutorial/datastructures.html
  - http://sthurlow.com/python/lesson06/

Lesson 6 (Thursday, Week 3)

Introduction
Understanding recursion is one of the fundamentals of learning efficient programming. Recursion provides an alternative way of thinking, and programming than just writing sequential code. Explaining the fundamentals of recursion early on allows students to incorporate it as a common programming technique. As students are already familiar with iteration and functions, this is a good point in the course to bring the discussion of recursion. The sole focus of this lecture is to teach students recursion by incorporating mathematical concepts, providing them with examples and conducting in-class programming activities to allow them to apply the newly learned concept.

Content Focus
The content focus of this lecture is mainly on recursion. Recursion is a process when a function calls itself again and again while performing a certain operation. This is a new and intriguing concept for students and requires a dedicated lecture. After students have been given a comprehensive introduction and know the types or recursion, the lecture will follow detailed examples of recursion, which incorporate interesting and common mathematical problems they are already familiar with. Finally, students will be given a discussion-based exercise in which they find the Fibonacci sequence up to the nth term, where n is the main recursive function parameter.

Learning Goals
By the end of the lesson students should be able to:
- Demonstrate an understanding of recursion by using it within numerical applications.
- Differentiate between recursion methods interpret their complexity and memory usage.

Class Outline (Total Time: 75 minutes)

1. Introduction, administrivia, and announcements. (5 minutes)
   a. Provide an introduction to the topics being covered in the lecture.
   b. Make an announcement to start on homework 3 if they have not started.
   c. Inform them of the topics that will be covered in the recitation.

2. Introduction to recursion (15 minutes)
   a. Explain the mathematics ore recursion and what it is in the context of computer programming.
   b. Elaborate on the concept of base case and additional cases.
   c. Detail how recursion is different from sequential programming.
   d. Demonstrate the strong linkage between function and recursion.
   e. Answer students’ questions related to this module.

3. Types of recursion (20 minutes)
   a. Discuss the following kinds of recursion:
      - Direct and indirect recursion.
      - Single and multiple recursion.
      - Linear, binary, and tail recursion.
Lesson Plans
Mohammad Taha Khan

b. Discussion of the unique applications and examples of each kind of recursion method.
c. Explain recursion and stack overflow.
d. Answer students’ questions related to this module.

4. Guided examples of recursion (15 minutes)
a. Walk students through the following guided examples of recursion.
b. Answer student questions related to this module.

5. Programming activity (15 minutes)
a. Break student into discussion groups.
b. Provide a programming activity with the following required deliverables:
   ▪ The next number in the Fibonacci sequence is determined by adding two previous numbers.
   ▪ Given number 0,1, write a function that calculates the Fibonacci sequence up to n via recursion.
   ▪ Modify the function such that it sums the last and third to last instead of two consecutive numbers.
c. Discuss the solutions of the programming activity and answer students’ questions.

6. Weekly survey and feedback (5 minutes)
a. Request students to fill out a short survey and provide feedback on the topics learn.

Required Class Materials
1. Laptop with PowerPoint, and required adapters
2. Projector setup
3. Whiteboard and markers
4. Class slide-deck handouts
5. Classroom design which allows students to feasibly converse in groups without major disruption

Preparation Materials
- Supplemental online materials:
  - https://realpython.com/python-thinking-recursively/
  - https://www.geeksforgeeks.org/recursion/
  - https://pythonspot.com/recursion/