Lecture 13: Review and Wrap-up

CS5001 / CS5003:
Intensive Foundations
of Computer Science

PDF of this presentation

Lecture 13: Review and Wrap-up

You have learned a lot of things this semester! The final exam will be *cumulative*, meaning that it will cover everything during the semester. There will be a focus on the material since the midterm.

The exam will be a 3-hour exam in class on **Tuesday, December 10th, from 6pm-9pm** (or **7pm-10pm**, if you want -- you can start at either time).

The exam will be on BlueBook again, and you should already have the program on your computer.

We will review tonight, and on Thursday there is an optional lab where we can also review, or you can work on assignment 8.

Lecture 13: Review and Wrap-up

Here are the topics that we have covered in the course:

Pre-midterm:

- Variables
- Basic types: `int`, `float`, `str`
- Library functions
- Branching: `if` / `elif` / `else`, `not`, `or`, `and`, `==`, `!=`, `<`, `>`, `<=`, `>=`
- Iteration
- Lists, list slicing, list comprehension
- Tuples
- Creating your own functions
- strings and f-strings
- recursion
- dictionaries
- file processing

Lecture 13: Review and Wrap-up

Post-midterm:

- Object oriented programming
  - Creating classes
    - `__init__`, `__str__`, `self`
  - Inheritance
- Stacks
- Queues
- Searching
  - Linear Search
  - Binary Search
- Sorting
  - Insertion Sort, Selection Sort, Merge Sort, Quicksort (and Radix Sort)
- Using Python for AI
- Iterators, Generators, Lambda Functions, and Sets

Lecture 13: Review and Wrap-up

Study Tips

1. Review lecture slides -- make sure you understand all example code
2. Review all labs
3. Review all assignments
4. Review Exam Reference Sheet
5. Review your midterm and make sure you understand what you missed (and would get it right if you did it again!)
6. Do the practice exam: https://course.ccs.neu.edu/cs5001f19-sf/static/final/final-practice.zip
7. Ask questions on Piazza about concepts or programs you don't understand

Lecture 13: Review and Wrap-up

Post-midterm:

- Object oriented programming
  - Creating classes
    - `__init__`, `__str__`, `__eq__`, `self`
  - Inheritance

Be prepared to create a class with an `__init__`, `__str__`, and `__eq__` functions

Be prepared to create an inherited class that calls `r`

Be prepared to use classes provided for you

Lecture 13: Review and Wrap-up

Post-midterm:

- Stacks

Understand the first-in-last-out nature of a stack

Understand what the **`push`**, **`pop`**, **`top`**, and **`empty`** functions do

Be prepared to create a stack using a list

Be prepared to use a stack to solve problems that can benefit from the use of a stack

Lecture 13: Review and Wrap-up

Post-midterm:

- Queues

Understand the first-in-first-out nature of a queue

Be prepared to create a queue from a list

Understand what the **`enqueue`**, **`dequeue`**, **`front`**, and **`empty`** functions do

Be prepared to use a queue in a program to solve a problem

Lecture 13: Review and Wrap-up

Post-midterm:

- Searching
  - Linear Search
  - Binary Search

Understand the difference between a linear search and binary search

Be able to trace a binary search and explain how it works. Make sure you remember that in order to do a binary search, the elements must be sorted

Lecture 13: Review and Wrap-up

Post-midterm:

- Sorting
  - Insertion Sort, Selection Sort, Merge Sort, Quicksort (and Radix Sort)

Be able to explain all of the above sorts (except Radix sort)

Be able to actually code selection sort and insertion sort

Be able to talk about why merge sort is, on average, much faster than insertion sort

Be able to talk about why selection sort is never fast

Lecture 13: Review and Wrap-up

Post-midterm:

- Using Python for AI
- Iterators, Generators, Lambda Functions, and Sets

We won't ask specific questions about AI, nor will we ask about iterators, generators, lambda functions, or sets.

Lecture 13: Review and Wrap-up

Where to go from here

You are prepared for your next class, CS 5004 - Object-Oriented Design

The class is taught in Java, which means that you will need to learn a new programming language. Java is similar to Python in many ways, but it is also different in many ways. You will find it relatively easy to learn Java, but some things will be tricky to remember. Here is an example of the similarities and differences between Python and Java:

Java

```java
class Main {
  public static void main(String[] args) {
    for (int i=0; i < 10; i++) {
      System.out.println("Java " + Integer.toString(i));
    }
  }
}
```

Python

```python
def main():
    for i in range(10):
        print(f"Python {i}")

if __name__ == "__main__":
    main()
```

Some differences:

- Java has curly braces for blocks. Indentation is optional (though recommended)
- `for` loops look different (and require parentheses)
- Variables must have the type (e.g., `int i`)
- Statements end with a semicolon (;) in Java

Lecture 13: Review and Wrap-up

Another example:

```java
import java.util.ArrayList; // import the ArrayList class
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        // create a list
        ArrayList<String> myList = new ArrayList<String>();
        while (true) {
            System.out.print("Please enter a name (blank line to end): ");
            Scanner in = new Scanner(System.in);
            String s = in.nextLine();
            if (s.isEmpty()) {
                break;
            }
            myList.add(s);
        }
        System.out.println("You entered:");
        for (String s : myList) {
            System.out.println(s);
        }
    }
}
```

Yes, Java does look different! See the next slide for the equivalent Python program.

Lecture 13: Review and Wrap-up

Another example:

```
1  def main():
2      my_list = []
3      while True:
4          s = input("Please enter a name (blank line to end): ")
5          if s == '':
6              break
7          my_list.append(s)
8      print("You entered:")
9      for s in my_list:
10         print(s)
11
12 if __name__ == "__main__":
13     main()
```

Python programs tend to be shorter than Java programs. But, there are some similarities:

- There are **`while`** loops and **`for`** loops in both.
- Lists and ArrayLists are similar
- Both have **`break`** statements
- Both use dot notation

Lecture 13: Review and Wrap-up

The bottom line: you will be able to learn Java relatively quickly, despite the differences.

You have learned *programming* skills that translate across languages.

You have learned *problem solving* skills that translate across languages.

You should be proud of how far you've come during the semester, and good luck as you go forward to your future classes in the Align program!