Syllabus
COMP 152
Data Structures and Algorithms

Spring 2020

1 Logistics

- **Where:**
  - Class: Center for Science and Business (CSB), Room 309
  - Lab: Center for Science and Business (CSB), Room 309
- **When:**
  - Class: MWF 9–9:50am
  - Lab: W 2–4:50pm
- **Instructor:** Logan Mayfield
  - *Office:* Center for Science and Business (CSB), Room 344
  - *Phone:* 309-457-2200
  - *Website:* `http://jlmayfield.github.io/`
  - *Email:* lmayfield *at* monmouthcollege *dot* edu
  - *Office Hours:* or by appointment.
- **Website:** `http://jlmayfield.github.io/teaching/COMP152/`
- **Credits:** 1 course credit

Note: This Syllabus is subject to change based on specific class needs. Significant deviations from the syllabus will be discussed in class.

2 Description, Content, and Learning Goals

A continuation of COMP 151 that explores the essential data structures and algorithms of modern computing, including lists, stacks, queues, heaps, and trees. Students will design, analyze, and build Python programs that implement and utilize these data structures to solve computational problems, including a thorough survey of sorting and search algorithms. These theoretical constructs are complemented by exposure to good software development practices, including data abstraction via abstract data types and object-oriented software design. Strong emphasis is put on analyzing and evaluating how implementation choices made by the programmer impact overall program performance and maintainability.

2.1 Textbook

Goodrich, Michael T. and Tamassia, Robert and Goldwasser, Michael H. Data Structures and Algorithms in Python. Wiley. 2013. ISBN-13: 978-1-118-29027-9.

2.2 Topics

The course aims to cover chapters 1-8, parts of chapters 9-10, chapter 12, and parts of chapter 14 of the text. This includes, but is not limited to:

- Review of Python fundamentals
- Experimental and Asymptotic Algorithm Analysis
- Arrays and Sequence Data Types
- Abstract Data Types
- Binary Search Trees and Tree Traversal Algorithms
- Sorting, Searching, and Selection Algorithms
- Object-Oriented Design Basics and Patterns
- Recursion
- Stacks, Queues, Deques, and Priority Queues
- Linked Data Structures
- Maps and Dictionaries
- Graph Algorithms (as time allows)

2.3 Programming Environment

This course uses Python 3 and the PyCharm-Educational IDE. All software for this course is available free of charge and can be found on the web if students wish to install it on their personal machines. The instructor cannot guarantee support for installing and using other development environments.

2.4 Student Learning Outcomes

This course covers a variety of knowledge areas as categorized by the ACM/IEEE-CS Task Force on Computing Curricula. Note that not all of these areas are introduced in this course; some are touched upon previously and others will be developed further in later courses. At the end of the course, students should achieve the following learning outcomes with the specific level of mastery:

Knowledge Unit	Learning Outcome with Level of Mastery
Basic Analysis	• In the context of specific algorithms, identify the characteristics of data and/or other conditions or assumptions that lead to different behaviors. [Familiarity] • Explain what is meant by "best", "average", and "worst" case behavior of an algorithm. [Familiarity] • In the context of specific algorithms, identify the characteristics of data and/or other conditions or assumptions that lead to different behaviors. • Perform empirical studies to validate hypotheses about runtime stemming from mathematical analysis. Run algorithms on various input sizes and compare performance. [Assessment] • Determine informally the time and space complexity of simple algorithms. [Usage] • Understand the formal definition of big O. [Familiarity] • List and contrast standard complexity classes. [Familiarity] • Give examples that illustrate time-space trade-offs of algorithms. [Familiarity] • Use big O notation formally to give asymptotic upper bounds on time and space complexity of algorithms. [Usage]
Algorithmic Strategies	• Have facility mapping pseudocode to implementation, implementing examples of algorithmic strategies from scratch, and applying them to specific problems. [Familiarity] • Use a divide-and-conquer algorithm to solve an appropriate problem. [Usage]
Fundamental Data Structures and Algorithms	• Implement simple search algorithms and explain the differences in their time complexities. [Usage, Assessment] • Discuss factors other than computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data. [Familiarity] • Be able to implement common quadratic and $O(n \log n)$ sorting algorithms. [Usage] • Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in a particular context. [Usage, Assessment]
Fundamentals	• Explain the concept of modeling and the use of abstraction that allows the use of a machine to solve a problem. [Familiarity]
Basics of Counting	• Perform computations involving modular arithmetic. [Usage]

Continued on next page

Continued from previous page

Knowledge Unit	Learning Outcome with Level of Mastery
Processing	• Analyze simple problem statements to identify relevant information and select appropriate processing to solve the problem. [Assessment] • Identify the issues impacting correctness and efficiency of a computation [Familiarity]
Defensive Programming	• Demonstrate the identification and graceful handling of error conditions. [Usage]
Graphs and Trees	• Demonstrate different traversal methods for trees [and graphs], including pre, post, and in-order traversal of trees. [Usage] • Model a variety of real-world problems in computer science using appropriate forms of [graphs and] trees, such as representing ... a hierarchical file system. [Usage]
Data Modeling	• Describe the main concepts of the OO model such as object identity, type constructors, encapsulation, inheritance, polymorphism, and versioning. [Familiarity]
Object-Oriented Programming	• Compare and contrast the procedural and object-oriented approach. Understand both as defining a matrix of operations and variants. [Assessment] • Use subclassing to design simple class hierarchies that allow code to be reused for distinct subclasses. [Usage]
Algorithms and Design	• Determine whether a recursive or iterative solution is most appropriate for a problem. [Assessment] • Implement a divide-and-conquer algorithm for solving a problem. [Usage] • Apply the technique of decomposition to break a problem into smaller pieces. [Usage] • Implement a coherent abstract data type, with loose coupling between components and behavior. [Usage] • Discuss the importance of algorithms in the problem-solving process. [Familiarity] • Discuss how a problem may be solved by multiple algorithms, each with different properties. [Familiarity] • Implement, test, and debug simple recursive functions. [Usage]
Fundamental Programming Concepts	• Analyze and explain the behavior of simple problems involving the fundamental programming constructs. [Assessment] • Identify the base case and the general case of a recursively-defined problem. [Assessment]
Fundamental Data Structures	• Discuss the appropriate use of built-in data structures. [Familiarity] • Describe common applications for each data structure in the topic list. [Familiarity] • Write programs that use each of the following data structures: arrays, strings, linked lists, stacks, queues, sets, and maps. [Usage] • Compare alternative implementations of data structures with respect to performance. [Assessment] • Choose the appropriate data structures for modeling a given problem. [Assessment]
Development Methods	• Trace the execution of a variety of code segments and write summaries of their computations. [Assessment] • Apply a variety of strategies to the testing and debugging of simple programs. [Usage] • Construct and debug programs using the standard libraries available with the chosen programming language. [Usage] • Apply consistent documentation and program style standards that contribute to the readability and maintainability of software.
Software Construction	• Construct models of the design of a simple software system that are appropriate for the paradigm used to design it. [Usage]
Professional Communication	• Evaluate written technical documentation to detect problems of various kinds. [Assessment]

3 Workload

The course workload is as follows:

Category	Number of Assignments
Exams	6
Projects	2
Labs	10
Homework	8–10

Exams

All exams are weighted equally. There is no midterm or final exam in the sense that the exams are worth more than other exams or that they will necessarily take longer than other exams. Exams will generally focus on material covered since the previous exam but will be in some sense cumulative due to the nature of programming. Unless stated otherwise, assume that exams will be pencil and paper and that computers will not be available during the exam period.

Projects

Two larger scale programming projects will be undertaken during the semester. These projects will be individual efforts and will require much more effort than the programs written in lab or as part of homework. Students can expect to have two weeks from the time of the project assignment to complete the project. One or more lab periods will be dedicated to work on the project. It is highly recommend that all students make ample use of the time given on these projects.

Homework

Homework will consist of exercises from the textbook. Programming problems should be done and tested on a computer. Non-programming questions can be hand written or typed. In both cases, printed copies of the solutions are due in-class on the day the day they appear on the calendar.

Labs

Students will be placed into groups of two or three for each lab. Work will be done using *paired programming*, a programming practice where each member of the group takes turns typing while the other group member helps look for typos, bugs, and otherwise assists in the design of the code. Each group will submit their work at the end of the lab period regardless of the overall completeness of the assignment. The goal is to make good constructive progress on the assignment. Full credit can and will be given on unfinished work so long as it can be executed to complete some portion of the given task, shows evidence of purposeful progress, and the group made full use of the lab period.

3.1 Course Engagement Expectations

The weekly workload for this course will vary by student but on average should be about 13 hours per week. The follow tables provides a rough estimate of the distribution of this time over different course components.

Assignment Type	Time/week
Lectures+Labs	6 hours/week
Homework	1 hours/week
Exam Study Time	1 hours/week
Projects	3 hours/week
Reading	2 hours/week
	13 hours/week

4 Grades

This course uses a standard grading scale where percentage grades translate to letter grades as follows:

Score	Grade
94–100	A
90–93	A-
88–89	B+
82–87	B
80–81	B-
78–79	C+
72–77	C
70–71	C-
68–69	D+
62–67	D
60–61	D-
0–59	F

Students are always welcome to challenge a grade that they feel is unfair or calculated incorrectly. Mistakes made in the student's favor will never be corrected to lower a grade. Mistakes not in the student's favor favor will be corrected. *Basically, after the initial grading, a score can only go up as the result of a challenge.*

4.1 Grade Weights

The final grade is based on a weighted average of particular assignment categories. Students should be able to estimate your current grade based on your scores and these weights, but you may always visit the instructor *outside of class time* to discuss your current standing and check on some or all of the current course grade.

Category	Weight
Exams	48%
Projects	24%
Homework	8%
Labs	10%
Participation	10%

4.2 Participation, Attendance, & Late Assignments

This course will make almost daily use of Socrative for in-class question and answer sessions. Questions will cover portions of the text that were assigned as reading and will range from simple checks to see if the reading was done to more challenging questions that follow from a close examination of the reading. For the most part, the only requirement is to provide an answer to every question and participate in the resultant discussions. On occasion, questions will be evaluated for their correctness and performance on these questions will also factor into the course participation grade. Students who do the reading and start the homework as soon as possible will have very little to worry about.

While there is no strict attendance policy, the course participation grade is based in large part on engagement with socrative. Absent students cannot participate in socrative sessions. Students should avoid unexcused absences, as defined in the college-wide absence policy. Whenever possible, let the instructor know of the absence before it occurs. When unexcused absences do occur, it is the student's responsibility to make up for the lost class time and to seek the permission of the instructor to hand-in or complete assignments that are late due to an unexcused absence.

In general, assignments are due at the specified time and no late assignments will be accepted unless an extension was requested prior to the due date. There are, of course, exceptions to this rule and students needing extra time can always contact the instructor for an extension. Do not just give up and eat a zero for the assignment. Ever. There is no penalty in asking for an extension nor is there a limit on extensions. That being said, there is no guarantee an extension will be given without legitimate need.

This course is designed around the assumption that students *engage in new ideas before they're covered in class meetings*. This means doing assigned reading, taking a stab at homework problems, and as a result coming to class and lab with some understand about a new idea or, just as likely, with a host of questions

about something encountered in the reading and homework. Not attending class, skipping lab, and putting off work to the point that an extension is needed are signs that a student isn't holding up their end of the bargain and is not prepared to participate in class.

4.3 Academic Honesty

From the Monmouth College Academic Honesty Policy:

> "We view academic dishonesty as a threat to the integrity and intellectual mission of our institution. Any breach of the academic honesty policy - either intentionally or unintentionally - will be taken seriously and may result not only in failure in the course, but in suspension or expulsion from the college. It is each students responsibility to read, understand and comply with the general academic honesty policy at Monmouth College, as defined here in the Scots Guide, and to the specific guidelines for each course, as elaborated on the professors syllabus."
>
> "The following areas are examples of violations of the academic honesty policy:
>
> 1. Cheating on tests, labs, etc;
> 2. Plagiarism, i.e., using the words, ideas, writing, or work of another without giving appropriate credit;
> 3. Improper collaboration between students, i.e., not doing ones own work on outside assignments specified as group projects by the instructor;
> 4. Submitting work previously submitted in another course, without previous authorization by the instructor."
>
> "Please note that this list is not intended to be exhaustive."

The complete Monmouth College Academic Honesty Policy can be found on the College web page by clicking on "Student Life" then on "Scots Guide" in the navigation bar to the left, then "Academic Regulations" in the navigation bar at the left. Or you can visit the web page directly by typing in this URL: `https://ou.monmouthcollege.edu/life/residence-life/scots-guide/academic-regulations.aspx`

In this course, any violation of the academic honesty policy will have varying consequences depending on the severity of the infraction as judged by the instructor. Minimally, a violation will result in an"F" or 0 points on the assignment in question. Additionally, the students course grade may be lowered by one letter grade. In severe cases, the student will be assigned a course grade of "F" and dismissed from the class. All cases of academic dishonesty will be reported to the Associate Dean who may decide to recommend further action to the Admissions and Academic Status Committee, including suspension or dismissal. It is assumed that students will educate themselves regarding what is considered to be academic dishonesty, so excuses or claims of ignorance will not mitigate the consequences of any violations.

5 Accessibility

Student Success & Accessibility Services offers FREE resources to assist Monmouth College students with their academic success. Programs include Supplemental Instruction for select classes, Drop-In and appointment tutoring, and individual Academic Coaching. Our office is here to help all students excel academically, since all students can work toward better grades, practice stronger study skills, and manage their time better.

If you have a disability or had academic accommodations in high school or another college, you may be eligible for academic accommodations at Monmouth College under the Americans with Disabilities Act (ADA). Monmouth College is committed to equal educational access. To discuss any of the services offered, please call or meet with Robert Crawley, Interim Director of Student Success & Accessibility Services. SSAS is located in the new ACE space on the first floor of the Hewes Library, opposite Einsteins Bros Bagels. They can be reached at 309-457-2257 or via email at: ssas@monmouthcollege.edu.

5.1 Calendar

This calendar is subject to change based on the circumstances of the course. A detailed, day-by-day calendar of reading requirements and expected exam dates can be found on the course website.

Week	Dates	Notes	Assignments Due	Chapter(s)
1	1/16 — 1/17			1
2	1/20 — 1/24		Lab 1	1
3	1/27 — 1/31		Lab 2. Homework 1 (F).	1,2
4	2/3 — 2/7		Homework 2 (W). Lab 3.	2,3
5	2/10 — 2/14		Exam 1 (M). Lab 4.	3
6	2/17 — 2/21		Homework 3(M).	4
7	2/24 — 2/28		Exam 2 (M). Project 1 (F).	5
8	3/2 — 3/5	SPRING BREAK (F)	Lab 5.	5
	3/9 — 3/13	SPRING BREAK		
9	3/16 — 3/20		Lab 6. Homework 4.	6
10	3/23 — 3/27		Lab 7. Homework 5.	6,7
11	3/30 — 4/3		Exam 3. Lab 8	7,8
12	4/6 — 4/10	EASTER (F)	Exam 4. Homework 6. Lab 9	8
13	4/13 — 4/17	EASTER (M).	Homework 7	9,10
14	4/20 — 4/24	SCHOLAR'S DAY (Tu).	Exam 5. Project 2	9,10
15	4/27 — 5/1		Lab 10	12
16	5/4 — 5/8	READING DAY (Th)	Homework 8. Exam 6 (F,3-6pm)	12