COMP 152: Object-Oriented Data

Structures and Algorithms, Fall 2019

*This syllabus is subject to change based on specific class needs, especially the schedule. Significant*

*deviations will be discussed in class.*

 Logistics

 Content

 Description

 Sources

 Topics

 Programming Environment

 Assessment

 Assignments and Workload

 Grading

 Policies

 Schedule

 Monmouth College Services

 Student Learning Outcomes

Logistics

 Class meetings:

 Class: MWF 9:00 am - 9:50 AM in CSB 309

 Lab: W 2:00 pm - 4:50 PM in CSB 309

 Instructor: Robert Utterback

 Office: CSB 342

 Phone: 309.457.2202

 Website: https://robertutterback.github.io

 Email: rutterback@monmouthcollege.edu

 Office hours: After 10/21: M 10-12. T 9-11. Th 9-10. F 10-11. By appointment. Before 10/21: M 2-4. T

9-11. Th 9-10. F 2-3. By appointment (see the schedule on my webpage).

 Website: https://robertutterback.github.io/courses/comp152/f19/

 Credits: 1 course credit

 Prerequisites: A C or better in COMP151 or COMP161.

 Feedback: At any time during the course you can use this form to provide anonymous suggestions,

criticism, praise, etc.

Content

Description

/

Data structures continues the study of abstraction and programming through a focused study of data structures, algorithms, and abstract data types. The primary focus of this course is the design and development of algorithms and programs using data abstraction and information hiding via abstract data types *ADTs*. Data abstraction is absolutely fundamental to good programming practice and the management of large scale problems and data sets. This course is designed to round out a student’s understanding of basic computer science concepts while strengthening the program design and development skills through continued use of an object-oriented design methodology.

It is imperative that programmers and program designers be able to determine which of the many solutions available is the best for their specific task. Throughout the course, students will focus on to making solid quantitative and qualitative judgments about program efficiency and overall program design choices. Gone are the days when “it works” is a good enough assessment for the quality of a program.

Students will explore the ideas and concepts brought up in class and homework assignments during the weekly lab session. In addition to hands on exercises, lab sessions will be used to explore current, relevant research in computer science.

Sources

The course textbook will be:

 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.

Other sources will be posted on this webpage as needed.

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

- Object-Oriented Design Basics and Patterns

|  |  |
| --- | --- |
| - Experimental and Asymptotic Algorithm Analysis | - Recursion |

- Arrays and Sequence Data Types

- Abstract Data Types

- Binary Search Trees and Tree Traversal

Algorithms

- Sorting, Searching, and Selection Algorithms

Programming Environment

- Stacks, Queues, Deques, and Priority Queues

- Linked Data Structures

- Maps and Dictionaries

- Graph Algorithms (as time allows)

This course uses Python 3 and repl.it, an online IDE service. Students will receive, complete, and submit assignments all via the website. Students can also complete assignments on their local machines (I recommend the Anaconda software package) or on the departmental server, but all assignment submissions must be done through repl.it unless otherwise specified.

|  |  |
| --- | --- |
| Assessment | / |

Assignments and Workload

The weekly workload for this course will vary by student and by week but should be about 14 hours per week on average. The following table provides a rough estimate of the distribution of time over different course components for a 16 week semester, as well as detailing the type, amount, and relative value of all assignments.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Category** | **Amount** | **Final Grade** |  | **Total** | **Time/Week** |
| **Weight** | **Time** | **(Hours)** |
| Lectures | 41 | 10% (Participation) | - |  | 2.25 |
| Labs | 8–10 | 15% | - |  | 2.75 |
| Homework | 8–10 | 15% | 48 |  | 3 |
| Exam Study | - | - | 16 |  | 1 |
| Exams | 3 | 30% | - |  | - |
| Projects | 2 | 30% | 48 |  | 3 |
| Reading+Unstructured | - | - |  |  | 2 |
| Study |

14

Homework assignments will always either precede a lab to prepare for it or follow a lab to complete it. Each exam will focus primarily, but not necessarily exclusively, on the material covered since the previous exam.

Grading

Lab and homework assignments are graded on a simple 3 point scale, marked with (in decreasing order) a check-plus, check, or check minus. Your final grade for these two assignment categories is then based on the respective averages.

Your participation grade is based on a variety of activities. During class I will often make use of the Socrative app, so you’ll need to install this on your phones. Participating in Socrative questions and with in-class group activities is required for a decent participation grade; an A includes asking questions either in class or in office hours.

Your final grade is based on a weighted average of particular assignment categories. You can estimate your current grade based on your scores and these weights. You may always visit the instructor outside of class to discuss your current standing. Assignments and final grades use a standard grading scale shown below and will not be curved except in rare cases when deemed necessary by the instructor.

This courses uses a standard grading scale. Assignments and final grades will not be curved except in rare cases when its deemed necessary by the instructor. Percentage grades translate to letter grades as follows:

|  |  |
| --- | --- |
| **Score** | **Grade** |
| 94–100 | A |
| 90–93 | A- |
| 88–89 | B+ |
| 82–87 | B |

/

|  |  |
| --- | --- |
| **Score** | **Grade** |
| 80–81 | B- |
| 78–79 | C+ |
| 72–77 | C |
| 70–71 | C- |
| 68–69 | D+ |
| 62–67 | D |
| 60–61 | D- |
| 0–59 | F |

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

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

Policies

 **Late assignments**: In general, late assignments will *not* be accepted. Exceptions may be made only for situations beyond your control. If you feel your reason is justified, schedule a meeting with the instructor to plead your case.

 **Academic dishonesty**: Monmouth College’s official policy on academic dishonesty can be found here. **You** are responsible for reading and complying with that policy.

*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 student’s 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*

 **Collaboration**: We encourage you to make use of the resources available to you – it is fine to seek help from a friend, tutor, instructor, internet, etc. However, *copying of answers and any act worthy of the label of* *“cheating” is never permissible*! It is understandable that when you work with a partner or a group that the resultant product is often extremely similar. This is acceptable but be prepared to be asked to defend your collaborations to the instructor. *You should always be able to reproduce an answer on your own, and if you*  *cannot you likely* ***do not really know the material***.

One way to collaborate effectively is to avoid taking careful notes during a collaboration session. Discuss the material and sketch out possible solutions on a whiteboard. When you have finished, take a break and

|  |  |
| --- | --- |
| then write up your solutions without any help from notes or pictures from the study session. This not only | / |

helps avoid violations of academic dishonesty, it also improves your retention of the material!

When assignments are meant to be done in groups, you will be directed to turn in one set of solutions per group. Otherwise, each student must turn in an assignment representing their own work.

 **Electronic devices**: Do not use your phone in class. Keep it on silent or leave it at home. Any computer or tablet usage should be related to the course. Other usage is rude and distracting to others.

 **General expectations**: In short, I expect you to be respectful of others and take responsibility for your own learning. You are here to learn, so work hard and be professional.

 Just attending class is not sufficient to truly learn the material. Read the text, use the resources available at Monmouth College, and go beyond the material.

 If you miss class, you are responsible for everything covered on that day. College is, in some sense, your job. Take pride in creating quality work. Staple your assignments, label problems, and present your answers neatly and orderly.

 Your job is to convince me that you have learned the material – show your work! Even if you do not know a particular answer, guide me through your thought process.

Schedule

The following **tentative** calendar should give you a feel for how work is distributed throughout the semester. Assignments and events are listed in the week they are due or when they occur. *This calendar is subject to change based on the circumstances of the course*.

**Note**: All readings should be done **before** the class period in which they are listed below.

|  |  |  |
| --- | --- | --- |
| **Date** | **Topic** | **Assignments and Readings** |
| Wed 08/21 | Logistics, Overview, Programming |  |
| Environment |
| Wed 08/21 – | Lab 1: Python Basics with Turtle |  |
| Lab |
| Fri 08/23 | Python Review I: References, Parameter | 1.1-1.5 |
| Passing, Data Types |
| Mon 08/26 | Exceptions, Generators, and more | 1.6-1.9, Python Tutorial 3.1.2- |
| 3.1.3, 5.1.0, 5.3 |
| Wed 08/28 | Review Exercises | 1.10-1.11, Python tutorial 5.4-5.5 |
| Wed 08/28 – | Lab 2: Using Sets and Dictionaries |  |
| Lab |
| Fri 08/30 | Modules and Classes | 2.1-2.3 |
| Mon 09/02 | Class Hierarchies and UML | 2.4 |
| Wed 09/04 | Inheritance and Polymorphism in Python | Source code example |
| Wed 09/04 – | Lab 3: Object-Oriented Turtle |  |
| Lab |

/

|  |  |  |
| --- | --- | --- |
| **Date** | **Topic** | **Assignments and Readings** |
| Fri 09/06 | Common Python Bugs | 2.5-2.6 |
| Mon 09/09 | Experimental Performance Analysis | 3.1 |
| Wed 09/11 | Big O and Asymptotic Analysis | 3.2 |
| Wed 09/11 – | Lab 4: Prefix-Sums Design and Analysis |  |
| Lab |
| Fri 09/13 | Asymptotic Analysis Exercises | 3.3, Asymptotic Notation (all |
| sections) |
| Mon 09/16 | Justification and Proof | 3.4 |
| Wed 09/18 | Sequences and Dynamic Arrays | 5.1-5.2, Practice Exam 1 |
| Wed 09/18 – | Lab 5: Exploring Dynamic Array |  |
| Lab |
| Fri 09/20 | Dynamic Array Analysis | 5.3 |
| Mon 09/23 | Analyzing Python Sequence Operations | 5.4 |
| Wed 09/25 | Exam I Review | Practice Exam 1 Solutions |
| Wed 09/25 – | Project 1 | Project 1 out |
| Lab |
| Fri 09/27 | Exam I |  |
| Mon 09/30 | Stacks | 6.1 |
| Wed 10/02 | Exam Solutions, Queue | 6.2, Exam 1 Solutions |
| Wed 10/02 – | Free lab for Project 1 |  |
| Lab |
| Fri 10/04 | Deques, Exercises | 6.3 |
| Mon 10/07 | Linked Stack | 7.1 |
| (Wed 10/09) | (No class – 1st Half Semester Course |  |
| Exams) |
| (Wed 10/09 – | (No class – 1st Half Semester Course |  |
| Lab) | Exams) |
| (Fri 10/11) | (Fall Break) |  |
| Mon 10/14 | Linked Queue, Doubly-Linked List | 7.3 |
| Wed 10/16 | Recursion | 4.1, 4.3 |
| Wed 10/16 – | Lab 6: Linked Lists | Project 1 Due |
| Lab |
| Fri 10/18 | Recurrences I | 4.2 |
| Mon 10/21 | More Recursion | 4.4 |
| Wed 10/23 | (No class) |  |

/

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Topic** | **Assignments and Readings** |  |
| (Wed 10/23 – | (No class – Mentoring Day) |  |  |
| Lab) |
| Fri 10/25 | (No class) |  |  |
| Mon 10/28 | Project 1 Solution | 4.5 |  |
| Wed 10/30 | Recursion Exercises | 4.6 |  |
| Wed 10/30 – | Lab 7: Recursion Practice |  |  |
| Lab |
| Fri 11/01 | Selection Sort | 12.1 |  |
| Mon 11/04 | Exam Review |  |  |
| Wed 11/06 | Exam II |  |  |
| Wed 11/06 – | Lab 8: Sorting |  |  |
| Lab |
| Fri 11/08 | Insertion Sort | 5.5.2 |  |
| Mon 11/11 | Merge Sort | 12.2 |  |
| Wed 11/13 | Quick Sort | 12.3, 12.5 |  |
| Wed 11/13 – | Lab 9: Sorting II |  |  |
| Lab |
| Fri 11/15 | Binary Trees | 8.1-8.3.1 |  |
| Mon 11/18 | Depth-First Traversals | 8.4 |  |
| Wed 11/20 | Breadth-First Traversal |  |  |
| Wed 11/20 – | Project 2 | Project 2 out |  |
| Lab |
| Fri 11/22 | Priority Queues and Heaps | 9.1-9.3, skim 9.4 |  |
| Mon 11/25 | Heapsort; Implementing Maps | 10.1 |  |
| (Wed 11/27) | (Thanksgiving Break) |  |  |
| (Wed 11/27 – | (Thanksgiving Break) |  |  |
| Lab) |
| Fri 11/29 | (Thanksgiving Break) |  |  |
| Mon 12/02 | Hash Tables; Search Trees | 10.2, 11.1-11.2 |  |
| Wed 12/04 | Final Exam Review |  |  |
| Wed 12/04 – | Free lab for Project 2 |  |  |
| Lab |
| **Sat 12/07** | **Final Exam** |  |  |
| **8:00 AM** |
| Tue 12/10 |  | Project 2 due | / |
| Noon |

Monmouth College Services

 Student Success and Accessibility Services offers FREE resources to assist Monmouth College students

with their academic success. Programs include supplemental instruction for difficult classes, drop-in and

appointment tutoring, and individual academic coaching. The office is here to help students excel

academically, since everyone can work toward better grades, practice stronger study skills, and mange

their time better.

 Accessibility Services: 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 and

Accessibility Services. SSAS is located in the new ACE space on the first floor of Hewes Library, opposite

Einstein Bros. Bagels. They can be reached at 309-457-2252 or via email at ssas@monmouthcollege.edu.

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]

/

|  |  |
| --- | --- |
| **Knowledge Unit** | **Learning Outcome with Level of Mastery** |
| 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] |
| 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]

/

|  |  |
| --- | --- |
| **Knowledge Unit** | **Learning Outcome with Level of Mastery** |

- 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] |

/