COMP 372 — Artificial Intelligence — Fall 2023

Instructor:	Phillip Kirlin
Meetings:	Tue/Thu, 11–12:15pm, Briggs 108
Course website:	`https://pkirlin.github.io/ai-f23/`
Email:	`kirlinp@rhodes.edu` (please include "AI" somewhere in the subject)
Office hours:	See webpage for scheduled office hours. I am also available by appointment.

Course Overview: This course presents an overview of the fundamental techniques used in artificial intelligence today. Topics will include a subset of the following: agents, intelligent search, constraint satisfaction, game playing, utility theory, decision making under uncertainty, reinforcement learning, probabilistic reasoning, machine learning, and neural networks. Other topics may be presented with student interest and as time permits.

Text: Russell and Norvig, *Artificial Intelligence: A Modern Approach*, 4th edition, Pearson, 2021. (You may also use the older 3rd edition.)

Prerequisites: COMP 241 (Data Structures and Algorithms) is required. COMP 172 (Discrete Structures) is highly suggested, but not required. In particular, knowledge of the data structures taught in 241 (especially sets, maps, priority queues, and graphs) will be assumed, as will the mathematical concepts (such as elementary probability theory) taught in 172.

Course Work:

	Tentative weight	Tentative date
Written homework	20%	
Programming projects	35%	
Midterm	20%	Tuesday, October 24, in class
Comprehensive final exam	25%	Tuesday, December 12, 1pm

Final letter grades of A–, B–, C–, and D– are guaranteed with final course grades of 90%, 80%, 70%, and 60%, respectively. If your final course grade falls near a letter grade boundary, I may take into account class participation, attendance, and/or improvement during the semester.

Written assignments are due at the beginning of class on the assigned date. Programming projects are due on Canvas by 11:59pm on the assigned date. Homework assignments should be written neatly. Poorly written work will not be graded. All pages of assignments should be stapled together.

Tentative Course Schedule

- Uninformed search (DFS, BFS, uniform-cost search)
- Heuristic search (A* algorithm)
- Adversarial search (Minimax algorithm, alpha-beta pruning)
- Probabilistic reasoning
- Bayesian networks
- Naive Bayes classifiers
- Markov chains and hidden Markov models

- Reinforcement learning (value iteration, Q-learning)
- Neural networks

Late Work and Makeup Assignments: In general, late work will not be accepted without arranging an extension in advance with the instructor, and will often come with a late penalty. Please make every effort to submit assignments on time.

If you have a valid reason for a makeup exam, inform your instructor as soon as you know. A valid reason is a medical emergency, a death in the family, religious observation, a college-sponsored off-campus activity, and, quite frankly, very little else. Generally, assignment extensions will only be granted for *unplanned* circumstances (e.g., the first two reasons above).

Programming Assignments:

- All programs assigned in this course must be written in Python or Java. For some assignments, you will be able to choose the language you prefer to use, while others might require one language or the other.
- If you would prefer to use a different language for an assignment, this may be possible; contact the instructor.
- When turning in Java projects, submit only the Java source code files (`.java`); do not submit any files generated by the IDE (e.g., `.class`).
- Back up your code somewhere as you're working on your assignments. Computer crashes or internet downtime are not valid excuses for missing a deadline.
- Programming grades will be based on correctness of the program output, efficiency and appropriateness of the algorithms used in the code, and style and documentation of the source code.

Office Hours: In addition to regular office hours, I am also available immediately after class for short questions. You never need an appointment to see me during regular office hours; you can just come by. Outside of regular office hours, feel free to stop by my office, and if I have time, I'll try to help you. If I don't have time at that moment, we'll set up an appointment for a different time. Don't be shy about coming by my office or sending me email if you can't make my regular office hours. I always set aside time each week for "unscheduled" office hours.

Attendance: Attendance is expected for each class. If your attendance deteriorates, you will be referred to the dean and asked to drop the course. Attendance, participation, and apparent overall improvement trend may be considered in assigning a final grade. Attendance will be checked each class lecture period. After five unexcused absences, each additional absence will reduce the final grade for the course by one letter grade.

Class Conduct:

- I encourage everyone to participate in class. Raise your hand if you have a question or comment. Please don't be shy about this; if you are confused about something, it is likely that someone else is confused as well. Teaching and learning is a partnership between the instructor and the students, and asking questions not only helps you understand the material, it also helps me know what I'm doing right or wrong. Teaching and learning

is a partnership between the instructor and the students, and asking questions not only helps you understand the material, it also helps me know what I'm doing right or wrong.
- Please turn on your video camera when on Zoom. This is not required, but it is highly preferred.
- If you cannot make it to class for whatever reason, make sure that you know what happened during the lecture that you missed. It is your responsibility, and nobody else's, to do so. The best way to do this is to ask a classmate. Remote classes will often have recordings you can watch, but in-person classes might not.

Collaboration: Students should talk to each other about the subject matter of this class and help each other. It is fine to discuss the readings, lectures, and problems and ask questions about them. I encourage such questions in class as well as elsewhere. However, there is a line past which you must not go, e.g., copying a solution from a fellow student, book, website, etc. will cause you to fail the course, or worse. If a significant part of one of your solutions is due to someone else, or something you've read, then you must acknowledge your source. Failure to do so is a serious academic violation. Of course, even after you acknowledge your source you must still understand your solution and write it in your own words. Copying a solution from the web, a book, or classmate will result in failure even if you acknowledge your source, unless you put it in quotation marks and say something like, "Here is Amy's solution, but I don't understand it enough to absorb it and write it in my own words." However, this won't get you much — if any — credit.

Rules for Completing Assignments Independently

- Unless otherwise specified, programming assignments handed in for this course are to be done *independently.*
- Talking to people (faculty, other students in the course, others with programming experience) is one of the best ways to learn. I am always willing to answer your questions or provide hints if you are stuck. But when you ask other people for help, sometimes it is difficult to know what constitutes legitimate assistance and what does not. In general, follow these rules:
  - **Rule 1: Do not look at anyone else's code for the same project, or a different project that solves a similar or identical problem.**
    Details: "Anyone else" here refers to other members of the class, people who have taken the class before, people at other schools enrolled in similar classes, or any code you find online (including generated by AI tools such as ChatGPT) or in print. "Similar or identical problem" here should allow you to look at code that uses techniques applied in different situations that you can then adapt to your project. However, if you find yourself copying-and-pasting code or directly transforming code line by line to fit into your program, then that is considered plagiarism.
    Exception: You may help someone else debug their program, or seek assistance in debugging yours. However, this requires the person writing the code being debugged to have made a good-faith attempt to write the program in the first place, and the goal of the debugging must be to fix one specific problem with the code, not re-write something from scratch.
  - **Rule 2: Do not write code or pseudocode with anyone else.**

Details: You must make a good faith effort to develop and implement your ideas independently before seeking assistance. Feel free to discuss the project *in general* with anyone else before you begin and as you're developing your program, but when you get to the level of writing code or pseudocode, you should be working independently.

The underlying idea is that you are entitled to seek assistance in ways which will genuinely help you to learn the material (as opposed to just getting the assignment done). Programming assignments are graded as a benefit to you; they are your chance to show what you have learned under circumstances less stressful than an exam. In return, I ask only that your work fairly reflect your understanding and your effort in the course.

Additional Information: To streamline this syllabus, I have moved a number of policies common to all my classes to a separate document called "Additional Course Policies." Those policies should be interpreted as a part of this syllabus.