BOB JONES UNIVERSITY
EST. 1927

College of Arts & Sciences

CpS 110
Introduction to Object Oriented Programming
Spring 2025

Instructor:	Jordan Jueckstock
Office:	AL 76 *(3rd floor, left hall)*
Office Hours:	MWF 3pm, Tuesday 1pm *(others by appointment)*
Email:	jpjuecks@bju.edu

Course Description:

Introduction to computer science through use of a high-level programming language as vehicle for understanding capabilities and limitations of computers. An emphasis on the object-oriented paradigm, including basic class design principles. Development of problem-solving skills through programming exercises.

Course Context:

This course fulfills the following objectives of the Computer Science department:

- CS1: Design and implement efficient solutions to problems in various domains.
- CS2: Demonstrate understanding of fundamental concepts in computer science.
- CS3a: Communicate technical information effectively.

Course Reading:

No textbook purchases are required for this class. Readings will be assigned from freely accessible electronic resources:

- *How to Think Like a Computer Scientist* (online interactive textbook)
- *Automate the Boring Stuff* (automatetheboringstuff.com/2e)

Course Goals:

The goals this semester are:

- To strengthen problem-solving abilities
- To develop debugging skills (i.e., what to do when your solution doesn't work at first)
- To develop an appreciation for the power of abstraction as a tool for managing complexity
- By learning to create small systems, to better appreciate the genius of the Creator of the Universe

Schedule:

The anticipated schedule of lecture topics, class milestones, and due dates is maintained on the CpS 110 website at https://protect.bju.edu/cps/courses/cps110/schedule/.

Assignments:

Quizzes cover topics that have been recently introduced in class to check student comprehension of the presented materials. Quizzes are taken *before* class on the day due and reviewed in class.
Lab assignments are small to medium program exercises to give you practical experience with the algorithms and theory discussed in class.
Programs are larger-scale *individual* assignments that require original thinking to solve a nontrivial problem using newly acquired concepts and skills.

Lab tests are given in conjunction with traditional written tests. Lab tests verify your understanding and mastery of practical programming skills by requiring individual completion of small, well-defined programming challenges within a 50 minute time window.
Tests cover all theoretical and practical topics covered in class. In general, tests are 50 questions and take approximately 50 minutes. Students should expect a variety of questions covering theory, practical application and programming questions, with emphasis on being able to read, understand, and predict the behavior of small portions of Python code.

Grading			
#	*Category*	*Pts.*	*Total*
13**	Quizzes	10	100
10*	Labs	10	90
6	Programs	60	360
3*	Lab Tests	100	200
2	Tests	75	150
1	Final Exam	100	100
	Total		**1000**

* Lowest grade dropped
** Lowest *three* grades dropped

Scale	
A	89.5+
B	79.5 – 89
C	69.5 – 79
D	59.5 – 69
F	0 - 59

Program Grading:
Learning how to program is important, but so is learning to write programs which are readable by others. Therefore, programs are graded as follows:

- 60% Correctness: Does the program produce correct results? Does it run according to specification. *Attention to the written specifications is vitally important!*
- 20% Style: Is the code written according to the style guidelines and the instructor's requirements? *Consistency and attention to detail are important!*
- 15% Reports: Is the program accompanied by a written report documenting the student's experiences crafting it and enumerating any unresolved issues? *Clear communication about expectations and results is important!*
- 5% Presentation: Was the program submitted according to instructions?

Deadlines / Late Work:
See the standard department late policy: https://cs.bju.edu/academics/policies/late-work-policy/. Note that a "free late" is an *earned privilege*, generally reserved for students who (a) have already established a track record of timely, quality submissions, (b) are experiencing an unanticipated/unavoidable scheduling hardship, and (c) proactively talk to the instructor about the situation before the posted due date.

The instructor reserves the right to change assignment due dates as deemed necessary. Assignments are due, electronically, by 11:59 pm of the date posted in the course schedule unless otherwise noted.

Due to grading constraints during finals week, the instructor reserves the right to shorten the late period for end of semester projects.

Accommodations:
Students needing accommodations due to a learning disability (visual, auditory, etc.) should provide an accommodation form obtained from the Academic Resource Center as soon as possible. Accommodations cannot be given without a form provided by the Academic Resource Center.

Getting Help:
Students struggling with an assignment or concepts in the class are strongly encouraged to ask the instructor for assistance either:

- in class *(usually best; someone else in class often has the same problem!)*
- before / after class
- during office hours (walk-in or by appointment)
- via email
- via Teams discussions

In order to maximize your opportunity to receive help and receive the best possible grade on an assignment / in the course:

- Start assignments early. This will give you more opportunities to realize you don't fully understand a concept and ask for assistance.
- Don't wait until the night before an assignment is due to ask questions. The instructor will answer last-minute questions on a best-effort, first-come-first-serve basis, but the odds that you are the only one asking questions *and* that a single answer is all that stands between you and successful completion of the assignment are perilously low.
- Request feedback. I cannot tell you what grade I would give to your solution for an assignment, but I can offer comments for how your solution can be improved.

Academic Honesty and Integrity Policy:
This policy is posted online at https://cs.bju.edu/academics/policies/academic-integrity-policy/. The policy text is included here verbatim for your convenience.

Cheating on assignments and tests is forbidden. All work is to be done individually unless group work is explicitly permitted. No collaboration is allowed on tests. For regular individual assignments, we expect that the submitted work represents the student's own intellectual effort, defined as follows:

1. The program was written primarily by the student. This means that the code (aside from starting code provided by the instructor, or code automatically generated by approved software tools) must have been crafted, not copied, by the student.
2. External resources used, whether electronic or from another human, must be documented as follows:
    a. Code snippets copied from online resources must be documented by a comment just above the copied snippet giving the URL of the page containing the source.
    b. Explanatory help or advice regarding the design or implementation of the solution received from people other than the instructor must be documented in a report accompanying the assignment submission. This report must detail:
        i. Source of information (e.g., name/email of the person who helped)
        ii. Relevance (i.e., how this resource helped and/or what it provided)
    c. Note that students **must not consult a solution to the assignment as a resource** in crafting their own solution, **nor share their own solution** with another student. Doing so constitutes cheating.
3. The student must be able to explain, on demand, the entirety of the program (or their assigned portion of the program, for group projects) on both the syntactic and semantic level.

Not all kinds of programming assignments require the same demonstration of personal intellectual effort. In the absence of any specific instructions, students should assume that at a minimum:

- For individual *lab* assignments, requirements **1** and **3** apply.
- For individual *programming* assignments, **all three** requirements apply.

Failure to comply with any relevant integrity requirement constitutes cheating. Such incidents will be reported to the academic integrity committee. To avoid trouble:

- **Do not look at another student's program code when seeking assistance**. On the other hand, if another student is seeking help from you, never use your own program code as an example. The only acceptable reason another student may look at *your* code is to help *you* find a problem in *your* program.
- **Do not write program code while another student (or lab assistant) is sitting with you**. You may work out designs in pseudocode on paper with another student, but you must write program code by yourself.
- **When seeking assistance from another person on a program assignment, always ask for and write down his/her name** so you can fulfill the documentation requirements (plus, you should be getting to know all your fellow students anyway).

Generative AI Policy:
The assignments in CpS 110 are designed to instill fluent programming intuition and methodology into students by requiring them to read and understand very simple programming problems and to design, implement, and debug working solutions through their own creativity and problem solving efforts.

The learning process, not the resulting program, is the goal!

Using a generative AI tool (e.g., ChatGPT, GitHub Copilot) to generate all or part of a solution can produce a program (which may even work), but it cannot produce the full experience of thinking through and solving the problem for oneself. Therefore, use of any such code-generating tool for any part of any CpS 110 assignment without prior approval by the instructor constitutes cheating. Evidence of such tool usage will be treated and reported accordingly.

Copyright Policy: