Yes, I put the phrase "Study Guide" in quotes. That is because I don't see this study guide the same way I saw study guides in FCCS or FOP. THAT is because I don't see the competency demo the same way that I saw the CDs in FCCS or FOP.

In those courses I think that most of the CD material was fact based. "Define such and such a term." "Explain this concept/algorithm." "Demonstrate you can program a specific chunk of code." This course is far less FACT based and far more CONCEPT based. It is both its strength and its weakness. That makes the nature of the questions very different.

This isn't a study guide in the traditional sense. It isn't a list of concepts you can review looking for the facts. Instead it will remind you what we did in Unit 1 and give you a couple example questions that I could ask you about this material. For many of these questions you might say "We NEVER talked about that." And you would be right. At least, not directly. But you did spend a lot of time thinking about a couple of basic ideas. This CD sees if you can apply those basic ideas to slightly new questions. I'm not looking for "THE RIGHT" answer. I'm looking for signs of competence. That is, that you can use this material to think/speak about new ideas.

What we studied directly:

Programming Fundamentals Meta-Knowledge

- Identify the Six Programming Fundamentals (sequencing actions, selecting actions, repeating actions, data with actions, creating actions (functions/modularization), responding to actions (events))
- Consider additional elements of programming
- What are the skills of programming
- What issues/elements of programming were particularly difficult for you or do you expect to be difficulty for your students

Questions I could ask that ask you to use/apply this knowledge

1. Pick two of the six programming fundamentals. Explain what they mean in computing. Give an example of the concept in a non-computing/everyday life activity
2. Consider these three fundamentals [example, selecting actions, repeating actions, and data]. Which order would you teach these in and why? What pre-requisite knowledge is gained by placing the first ones earlier than the later ones?
3. Consider this fundamental [example, functions]. Do you think this fundamental would be easier to teach in a block-based language (such as Scratch) or a text-based language (such as Python)? Why?
4. Pick one of the "skills of programming" you and/or your team discussed in this unit. Identify and briefly explain that skill. Discuss an activity you could conduct with your students to attempt to teach/develop that skill.
5. Pick one of the difficulty issues that you and/or your team discussed in this unit. Identify and briefly explain that difficulty. Discuss an activity, lesson, suggestions, etc. that you could conduct with your students to attempt to teach them to overcome this difficulty.