Types of exam questions

- True/False
    - T/F code: Decide if a Boolean expression would evaluate as True or False.
    - T/F concept: Decide if a statement about course concepts is True or False.
- Matching:
    - Match the vocabulary with the definition.
    - Match the key term with an example of that key term.
    - Match the code with its return value or printed output.
- Multiple-choice:
    - Match the code with its return value or printed output.
    - Select the correct or most efficient code.
    - Demonstrate an understanding of important concepts.
- Short-answer:
    - Fill-in-the-blanks in statements about vocabulary, course concepts, or within code.
    - Determine the printed output for a code snippet with hard-coded values.
    - Determine the printed output for a code snippet that takes user input.
    - Determine if a code snippet would raise an exception.
    - Identify the components of an exception Traceback message.
    - Predict the flow in code when debugging in 'Step over' mode.
- Modify given code or pseudocode:
    - Correct mistakes in given code or pseudocode.
    - Given a FOR-loop, write an equivalent WHILE-loop.
    - Given a WHILE-loop, write an equivalent FOR-loop.
    - Convert a non-batch process into a batch one.
    - Revise a workflow (code or pseudocode) for efficiency.
    - Complete a script that has been started.
- Write code:
    - Write a decision-making script.
    - Write a batch processing script.

Study advice

To study for the many of the matching, true/false, multiple-choice, and short answer type questions, reading the book and knowing the kinds of concepts the book and the lectures are teaching is really important. See detailed advice below about "reviewing concepts".

Some of the true/false, matching, multiple-choice, and short answer questions require you to ***predict*** the return values or printed output. This is testing your ability to trace code so that you can interpret found code and know what code you write is going to do. **Practice this skill for the exam**. See the details below about "how to practice tracing code".

To study for the code modification and writing type questions, practice writing code. Write problems from the in-class exercises, slides, book examples, and homework provide. Getting used to writing code comes through practice.

Take the practice exams after studying with the above techniques. To simulate the exam conditions, print the exams, write your answers down, and time yourself. Grade yourself. Post questions as needed on the message board.

Reviewing concepts

Re-read each chapter. Look at the tips and figures, and examine the example code. It should be easier to understand the sample code now than the first time you read it. Closely re-read any concepts you're uncertain about.
Quiz yourself on the key terms at the end of the chapter. Can you define the term? Can you provide several examples of the term or give an example of where it would be used?

Redo the questions in the homework, in-class exercises, and quizzes. Didn't understand a concept? Our feedback? Still not sure how to do it correctly? Follow up with questions on the message board. We will respond with interactive conversations.

How to practice tracing code

Some of the true/false, matching, multiple-choice, and short answer questions require you to ***predict*** the return values or printed output. This is testing your ability to trace code so that you can interpret found code and know what code you write is going to do. At the very basic level, this means you should know what code expressions will return and how the code will flow. This is something you should practice. To practice this skill:
1) Take some example code (from the book, the lectures, the in-class exercises, the quizzes, the homework, or that you make up) and predict the values it will return or print.
2) Write down your prediction on paper (This may seem unimportant--you might want to just hold it in your head, but writing it down will give you practice for exactly what you need to do on the exam)
3) Then try the code in the Interactive Window.
4) Compare your prediction to the result. Look closely at the punctuation, Look closely at the case (capitalization). Do they match? If not, do you understand why? If not, post on the message board.
5) Repeat on different code. Many times.

Learn output tracing of:

- Numerical, string, and list operations
- String and list methods
- Branching: GIven some if/elif/else blocks containing some print commands, write down what is happening based on different values for variables in the conditions

- Compound conditions: Given an if/else block with a compound condition for the if statement, write down what is happening based on different values for variables in the conditions.
- Loops: Given a FOR or WHILE loop containing some print commands (usually iterating over some integer values), write down what is happening, given different iterating values.
- User arguments: how do they print out? How will they affect branching or looping code?
- Broken code: What error will raise and why?

Exam objectives

- As with any graded course component, this helps us to evaluate your current level of understanding.

- More importantly, the exam is designed to cement your learning. To gain the ability to write useful code, you need to take time to practice, study, and reflect on the concepts and you need to know certain basic things by heart.

Once you know the basics, for truly learning how to program, the best way is to do it is to program. Write every workflow you think of. Write every problem from the in-class exercises, slides, book examples, quizzes, and homework provides. Getting used to the language and constructs you are being taught comes through practice. You'll start to see the errors you made in your code and be able to identify them more easily. When you encounter additional coding problems, you will start to recognize the patterns you have seen before.

Your study approach is definitely a matter of preference, but no matter what, if you continue to program and become comfortable around the code, you will succeed.