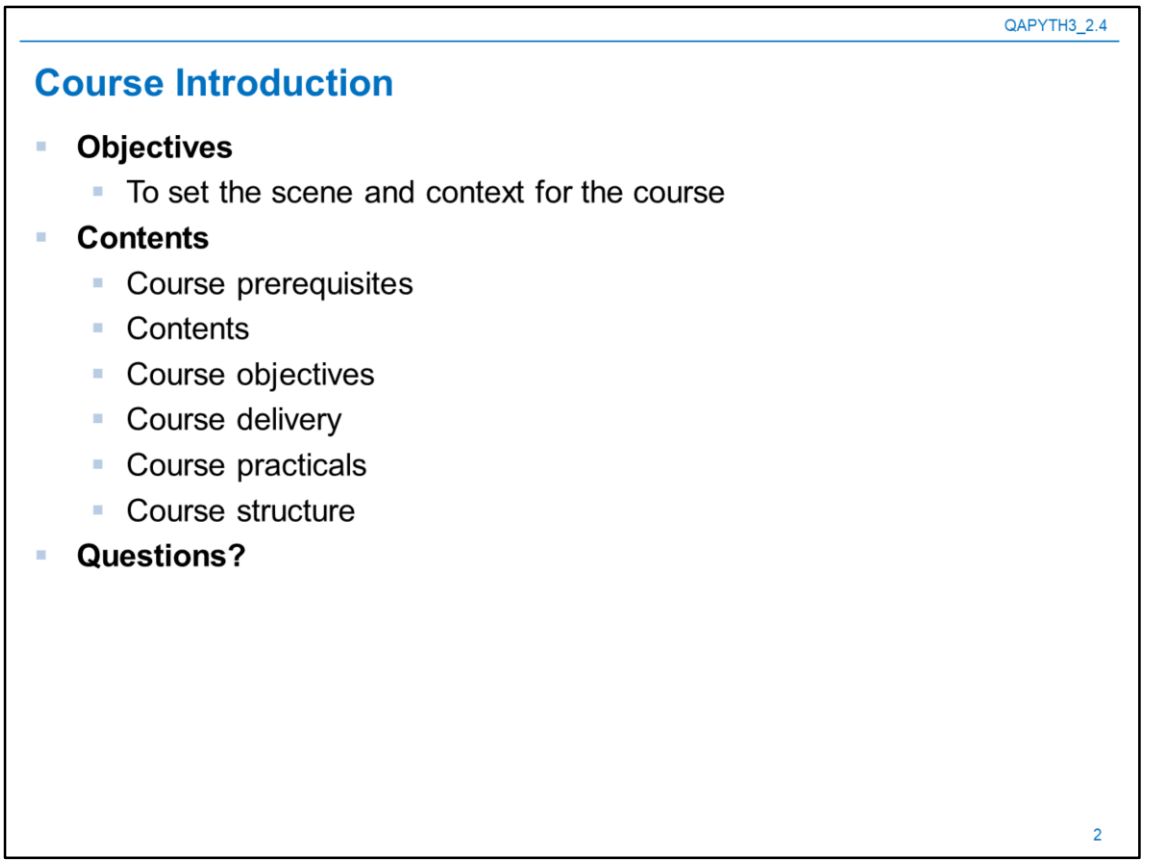
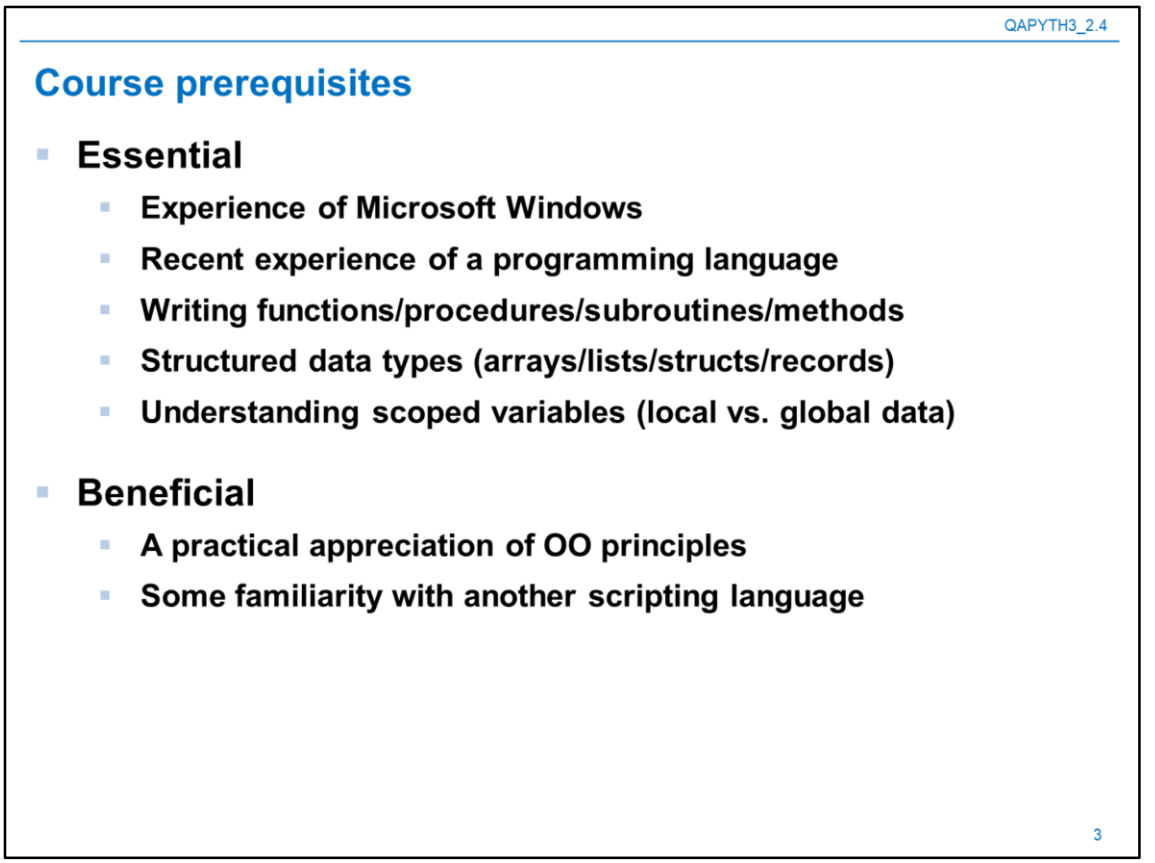
Python 3 Programming

Course Introduction



The objectives for this chapter are to introduce the course structure, to provide an overall picture of the course content and format.



If you feel that you are on the wrong course, tell us now! We will try to help you find the right one.

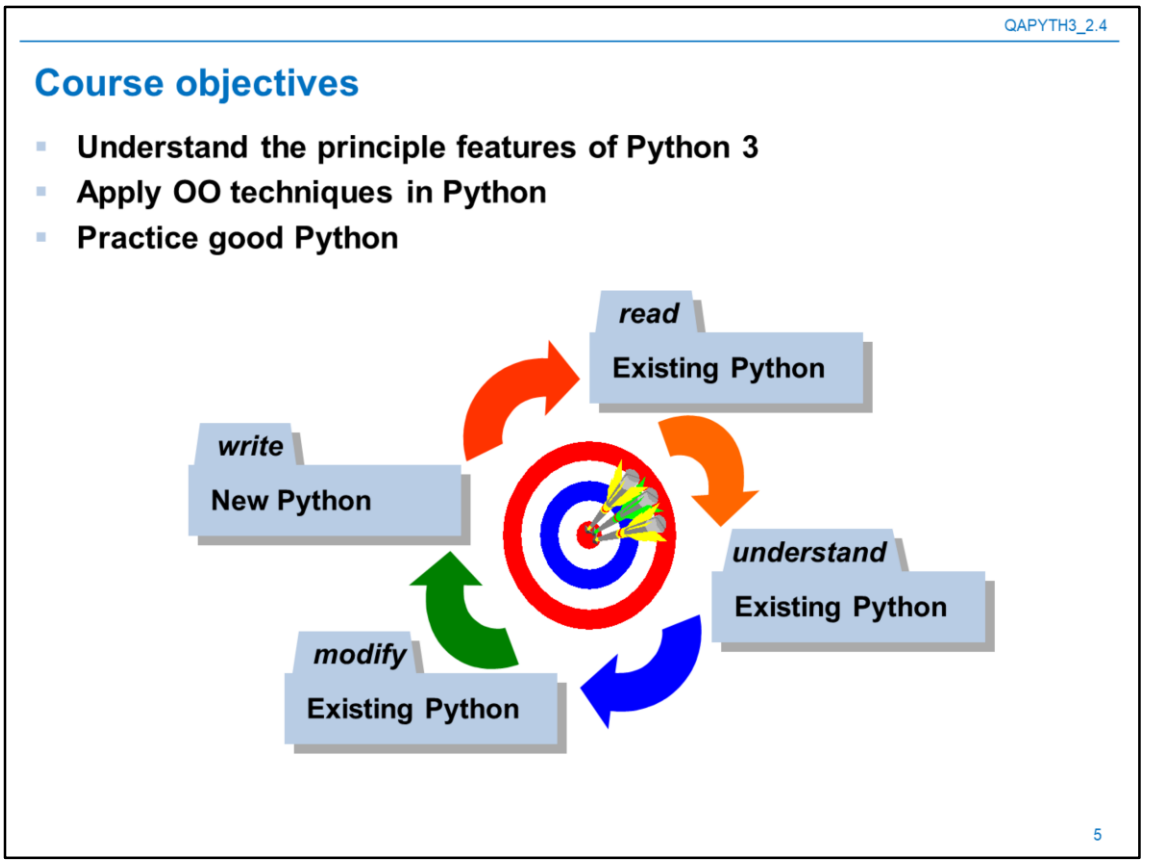
We do make some assumptions about you, as stated in the course outline and prerequisites in the course catalogue. And this slide!

QAPYTH3_2.4

Python 3 Programming - Contents

1. Introduction to Python 3
2. Fundamental variables
3. Flow control
4. String handling
5. Collections
6. Regular expressions
7. Data storage and file handling
8. Functions
9. Advanced collections
10. Modules and packages
11. Classes and OOP
12. Error handling and exceptions
13. Multitasking
14. The Python standard library
15. The way ahead

4



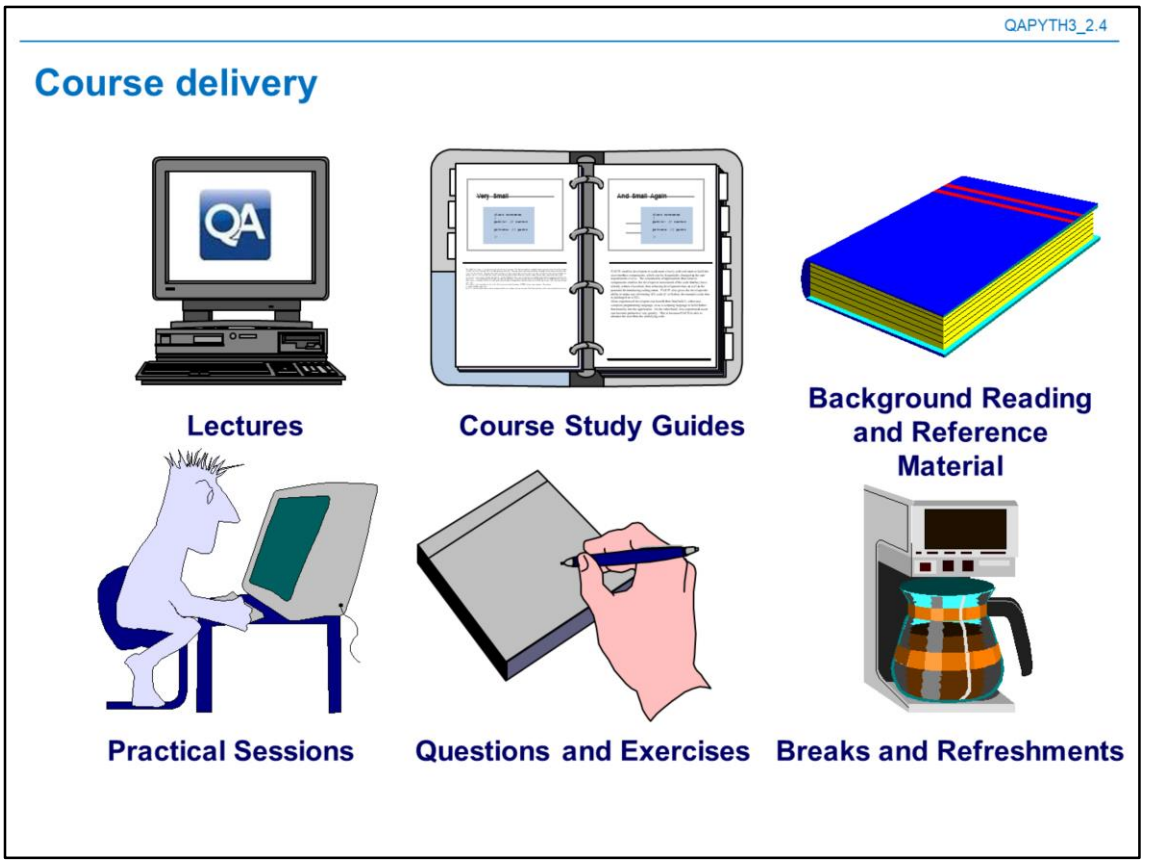
The course has been designed with this criteria in mind. It is assumed that you are an experienced and proficient programmer without any or with a modest amount of Python. This course is not an introduction to programming or an overview of Python.

For many features in the language, you can consider four levels of knowledge and experience:

> The entry level is to be able to read code that uses certain language features and syntax.
>
> The next level is to be able to use code that has already been written, for instance prewritten libraries.
>
> Then comes the ability to modify existing code, to change (or correct!) the way it behaves.
>
> The final level is to write new functions, classes and programs from scratch.

Object-oriented techniques are considered to be the most powerful way of getting the most from the language, and these will be introduced and discussed throughout the course.

Newcomers to such a large language are often faced with daunting task of having to program new systems without a complete knowledge of all the language's features and its common pitfalls. Providing development guidelines assists developers in using the language effectively from an early stage.



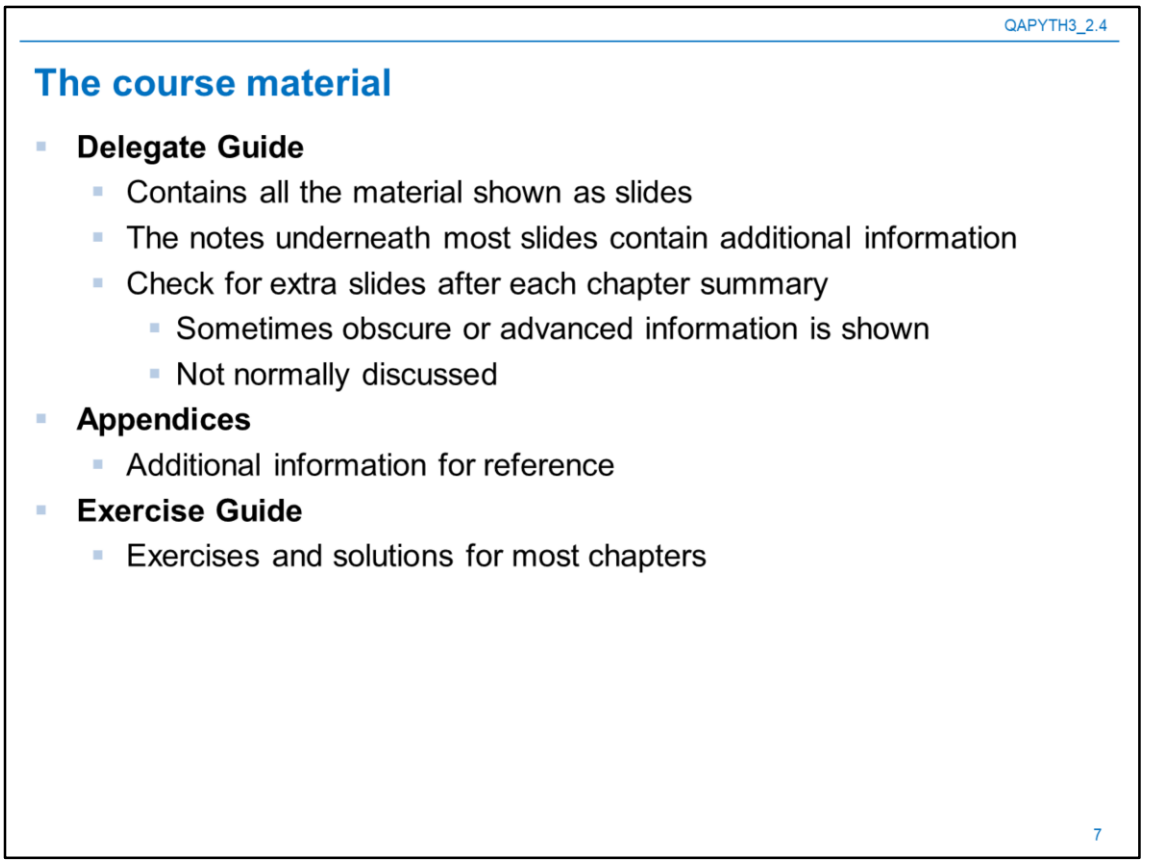
The course time will be divided between lecture periods and practical exercise sessions. The idea behind this is "listen and learn, see and understand, do and remember."

The lecture material includes many examples. The appendices provide reference and background reading material.

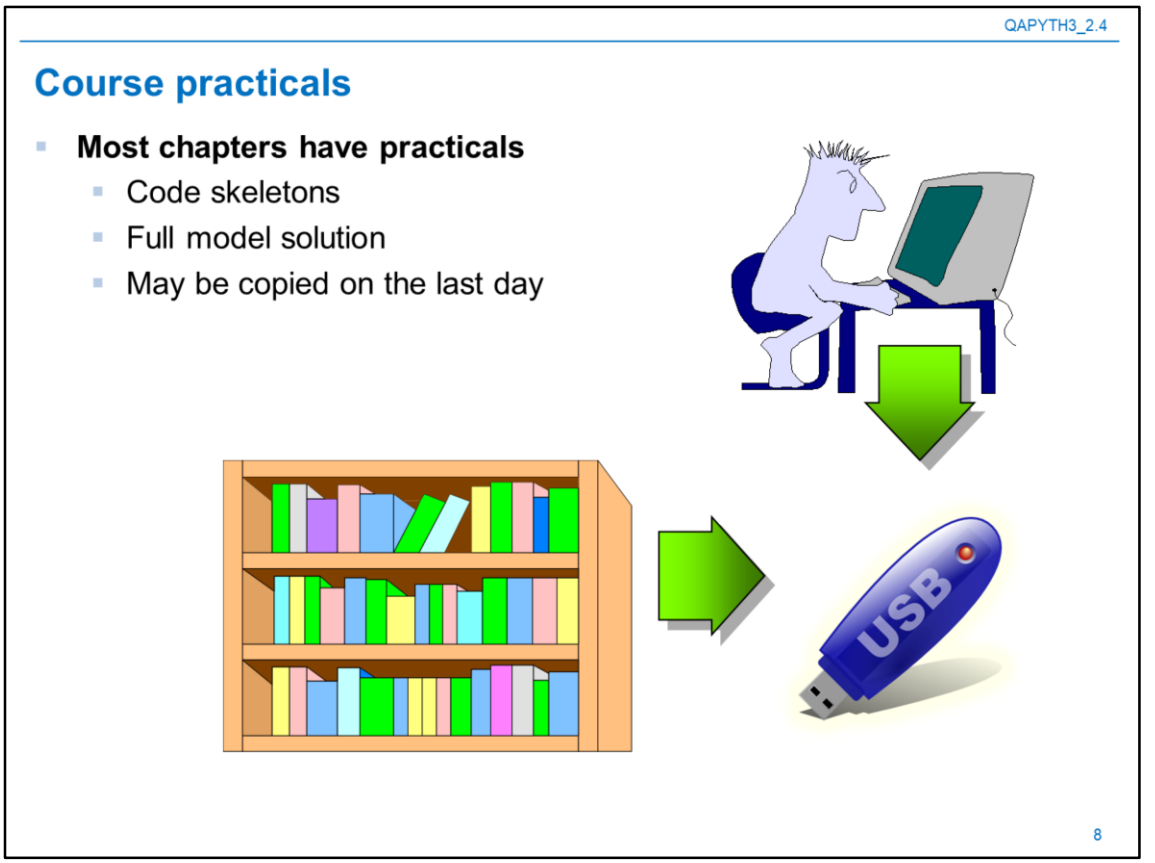
The course notebooks contain all the slides that will be shown, so you do not need to copy them. In addition, there are extra textual comments (like these) below the slides, which should amplify the slide or provide further information.

The best courses are not those in which the instructor spends all his or her time pontificating at the front of the class. Things get more interesting if there is dialogue, so please feel free to make comments or ask questions. At the same time, the instructor has to think of the whole group, so if you have many queries, he or she may ask to deal with them off-line.

Work with other people during practical exercise sessions. The person next to you may have the answer, or you may know the remedy for the problem that your neighbor is having.

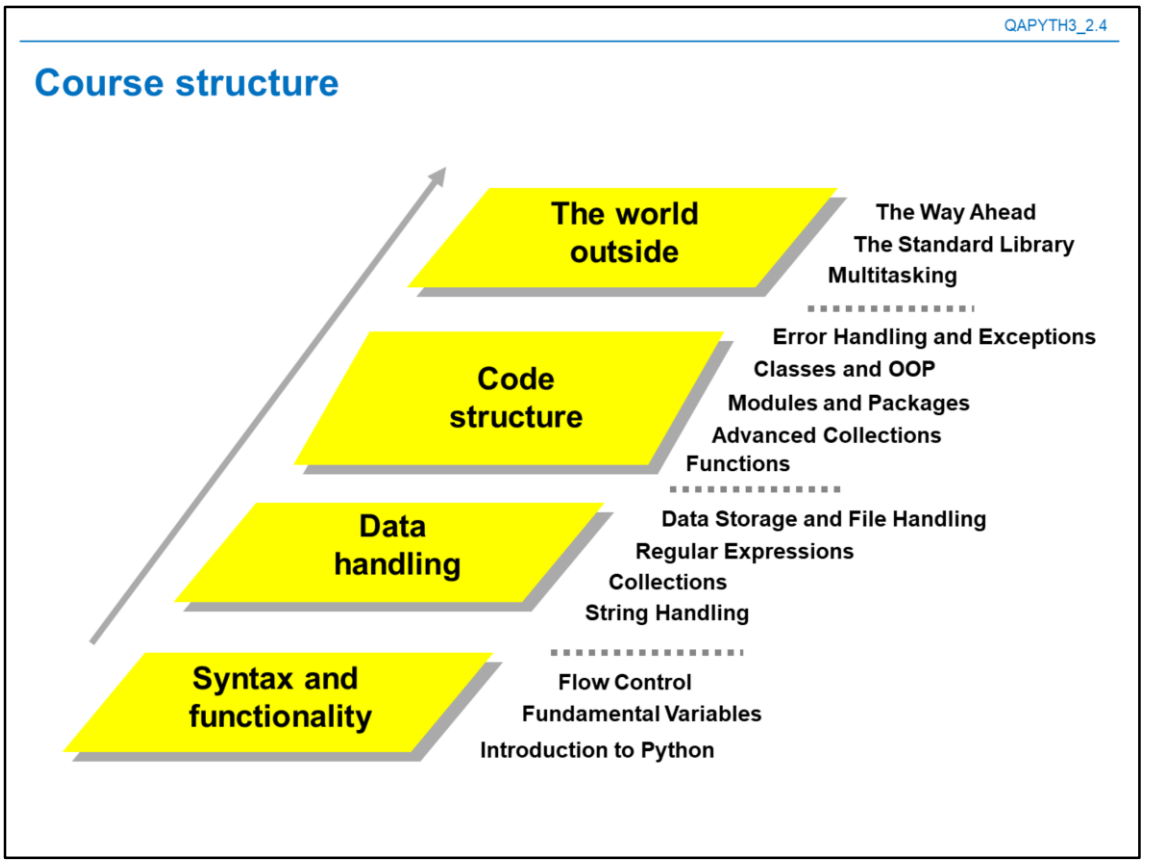


Most slides in the Delegate Guide have notes like this, although this chapter is an exception. The notes often contain additional information which you can read later. Obviously, you have spotted these already!



The practical questions are printed in their own section. The practical questions for each chapter are arranged so that they get progressively harder. Opening questions are based on the core material of the chapter, with later optional questions being based on more advanced material or detail. Many of the practicals in a session build on each other.

At the end of the course, you can take away a copy of the all the practicals, the worked solutions and the original code skeletons.



The course has been designed as a set of progressive sections, each of which builds on the section before.

Please feel free to ask any questions at any time! If in doubt, ask.

The importance of asking questions cannot be overstated. Do not be afraid to ask questions. Later chapters build on earlier chapters, so it is especially important to ask questions early on. There is no such thing as a stupid question.

Chinese proverb: "He who asks is a fool for five minutes, but he who does not ask remains a fool forever."

"To learn we must be willing to make mistakes." (Gerald Weinberg; The Psychology of Computer Programming)