Guidelines for Authors

Here are some general guidelines to help familiarize you with O'Reilly's house style and get you writing chapters quickly.

Planning

Creating a detailed outline before you start writing may seem like busywork, but it can be extremely helpful. Knowing exactly what you plan to write about will keep chapters from meandering and help prevent scope creep. You probably wrote an outline for your proposal, but it may be helpful to flesh it out with more details before you start writing.

At the very least, an outline should indicate the topic of each chapter and the main sections that each chapter will contain. If you want to indicate subsections (and even sub-subsections), great. The more details you include, the easier it will be to actually write the chapter.

Also, if you'd like, you can organize the book into parts (e.g. Part 1 covers topic X and contains Chapters 1-4, Part 2 covers topic Y and contains Chapters 6-7), though this isn't required.

Writing Style

Make your writing as conversational as possible. Imagine you're speaking to someone one on one, not giving a formal lecture to a large group. Refer to the reader as "you" and to yourself as "I".

Define your terms. Whenever you introduce a new term, if you aren't confident that 99% of your readers will know what the term means, define it.

Present information clearly and succinctly. Your readers are busy people, so you're doing them a huge favor if you teach them efficiently.

Use active voice. Unlike passive voice, active voice makes it clear exactly who is performing the action. Identifying passive voice is simple: If you can insert the words "by zombies" after the verb and the sentence still makes sense, you've got passive voice; in that case, revise the sentence to make it active. For example:

> **Passive voice (bad):** "The file is then imported [*by zombies*]."
> **Active voice (good):** "Android Studio then imports the file."
>
> **Passive voice (bad):** "Planning has been painstakingly carried out [*by zombies*], the stakeholders for the intelligence program have been identified [*by zombies*], goals have been set [*by zombies*], and requirements identified [*by zombies*]."
> **Active voice (good):** "You've painstakingly carried out planning, identified the stakeholders for the intelligence program, set goals, and identified requirements."

Chapter Structure

If you ever learned about 5-paragraph essays in school, most of that structure applies to writing chapters, too (though you're *not* limited to 5 just paragraphs per chapter!).

Here's the short version of how to use that structure:

> Tell readers what you're going to say, then say it, and then summarize what you said.

And here's the longer version:

> Every chapter should begin with a paragraph or two that summarizes what the chapter is about and why that information is important to the overall topic of your book.
>
> The subsequent sections (and subsections) should walk readers through the information you're presenting. Keep readers oriented by including signposts like "As you learned in Chapter 3" and "I'll discuss this topic in more detail later in this chapter." Also, make sure to transition between sections. For example, say you have a section about topic X followed by one about topic Y. End the section about topic X by saying something like, "Now that you understand X, you're ready to dig into topic Y," and start the section about topic Y by explaining how topic Y relates to topic X. This daisy-chaining helps readers understand how concepts are connected and why you're covering them in this particular order.
>
> At the end of each chapter, summarize what you discussed in that chapter, and mention what the following chapter is going to cover.

Chapter Elements

Here's a quick overview of various elements that are found in most chapters.

Headings

Headings should be in sequential order: level 1, then level 2, then level 3, but not level 1 and then level 3. It's also fine to have only level 1 headings in a chapter.

Figures

Figures should always be mentioned in the text that precedes them, so that readers know what you're about to show them. Something as simple as "(see Figure #-#)" is fine, or you can elaborate a bit: "As Figure #-# shows, the XYZ widget…".

Also, ***figures need captions.*** Short captions are fine (e.g. "The XYZ widget"), though you're welcome to write longer, more informative captions. The following section includes examples of figures with captions.

Code

If you want to show readers a few lines of code (up to ~a dozen), you can simply intersperse it with the body text, as shown in Figure 1-1.

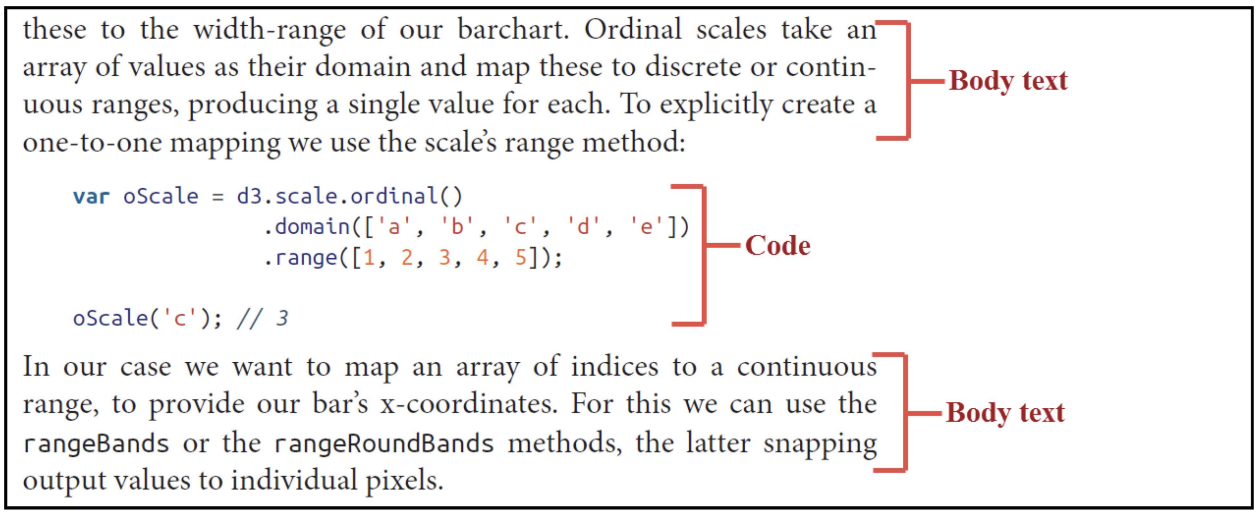


Figure 1-1: Code interspersed with body text.

If you want to include longer and/or more involved code, format it as a formal code example. Doing so means the code will have a number (e.g. Example 3-1) and a title, and you'll have the ability to include cross-references to it (hyperlinks that readers can click to hop to the example). Being able to cross-reference a code example is helpful if you plan to refer back to the code at several points in the chapter or book. Formal code examples can also include code callouts, which are labels you can use to point out specific lines of the code. Figure 1-2 shows what all this looks like.

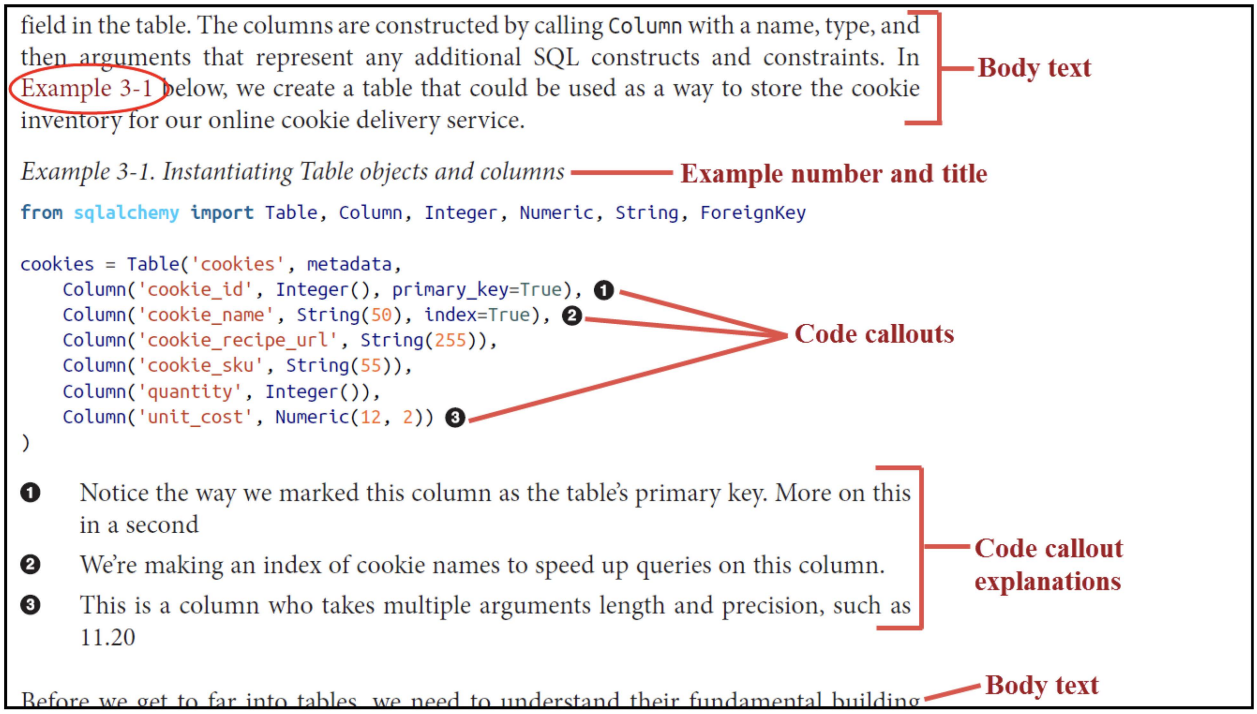


Figure 1-2: A formal code example with callouts. Be sure to refer to the example in the body text that precedes it, as shown here (circled).

Note: If you want to make code from your book available for readers to download, let your editor know. She'll work with you to make that happen. (If the book is available via O'Reilly's Early Release program, sample code needs to be available when the Early Release goes live.)

Tables

Just like formal code examples, tables should be numbered, have titles, and be mentioned in the body text that precedes them. Figure 1-3 shows an example.

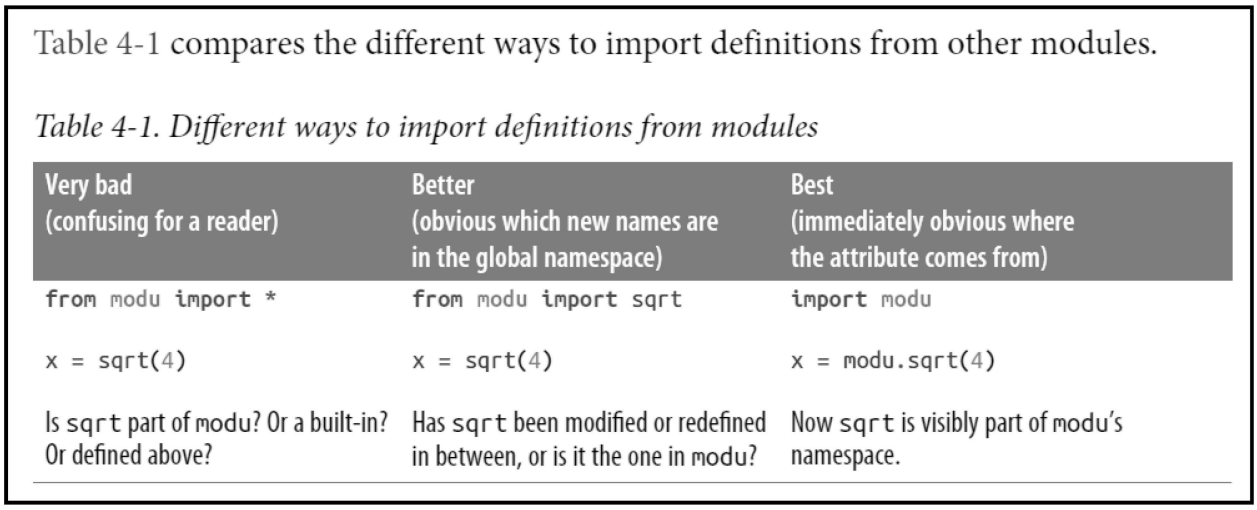


Table 4-1 compares the different ways to import definitions from other modules.

Table 4-1. Different ways to import definitions from modules

Very bad (confusing for a reader)	Better (obvious which new names are in the global namespace)	Best (immediately obvious where the attribute comes from)
`from modu import *`	`from modu import sqrt`	`import modu`
`x = sqrt(4)`	`x = sqrt(4)`	`x = modu.sqrt(4)`
Is `sqrt` part of `modu`? Or a built-in? Or defined above?	Has `sqrt` been modified or redefined in between, or is it the one in `modu`?	Now `sqrt` is visibly part of `modu`'s namespace.

Figure 1-3: An example of a table (with body text that introduces it).

Tips, notes, and warnings

These items are described in your book's preface and are generally used the same way in all O'Reilly books. Usually, they're just a couple of sentences long. If they're a long paragraph or more, you may actually have a sidebar instead (see below).

- **Tip:** A couple of sentences that convey a helpful bit of information, a quick way to do things better.
- **Note:** A couple of sentences of supplemental information. It describes something you want readers to keep in mind as they work, so you use a note to set it apart and make sure they see it.
- **Warning:** Similar to a note or tip, but specifically focused on a way to help readers avoid making a mistake or getting into trouble.

Each of these elements gets its own animal icon (see Figure 1-4).

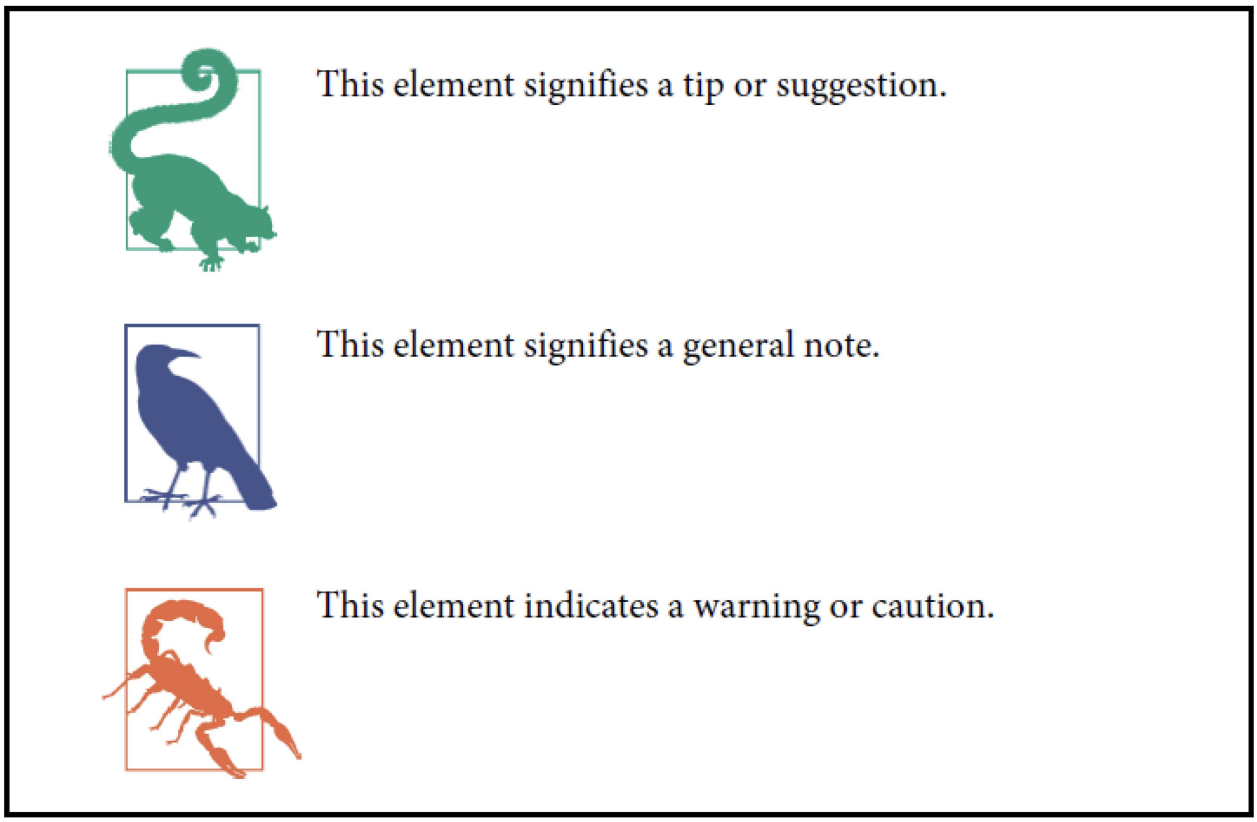
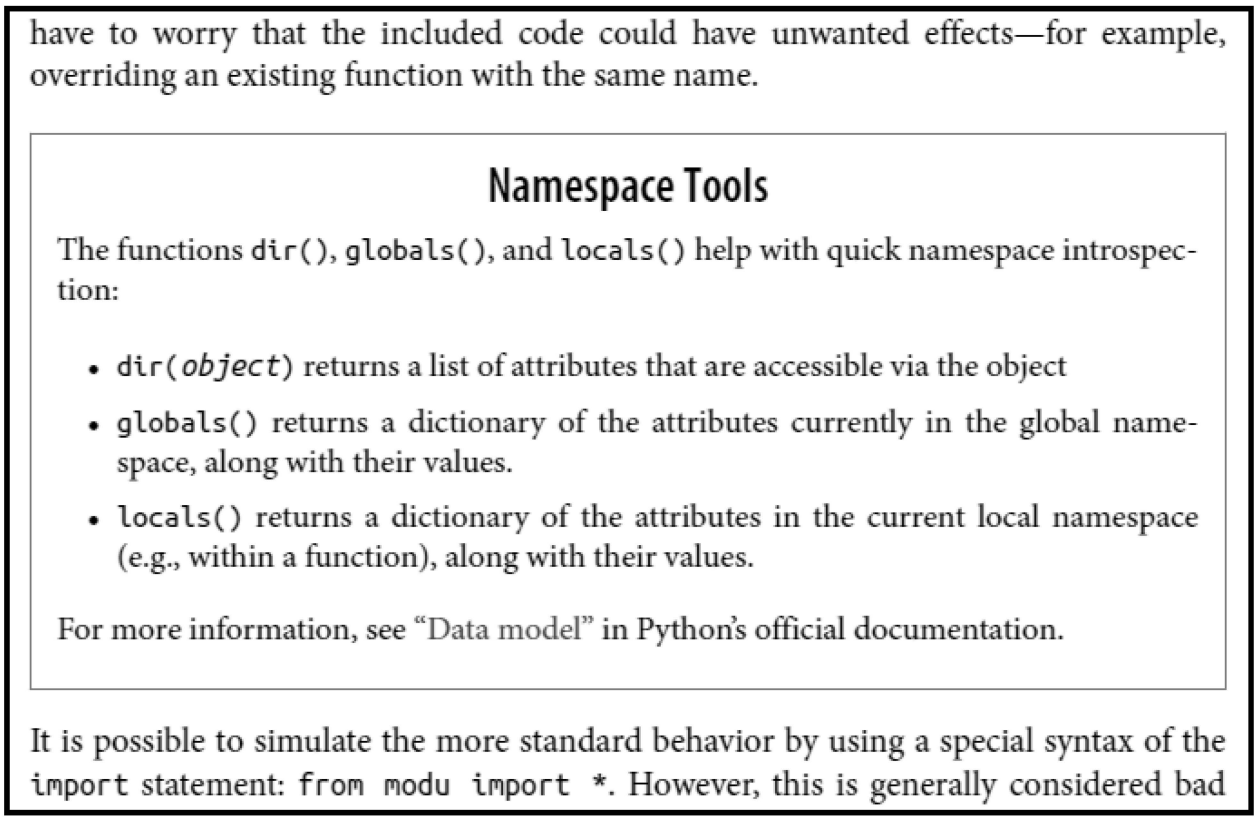


Figure 1-4: Icons for tips, notes, and warnings.

Sidebars

A sidebar is a great way to handle information that's not central to the main flow of the text. It may be historical, basic, or background information for readers who have some catching up to do, or advanced information that will only apply to your most experienced readers.

All sidebars have a title and are set off from the rest of the text by a bounding box (see Figure 1-5). A sidebar is usually between 1 and 3 paragraphs long. If it's longer than that, consider making it into a section of the chapter, perhaps with a Level 3 heading. If it's shorter than a paragraph, then it should probably be a note or tip instead (see above).


have to worry that the included code could have unwanted effects—for example, overriding an existing function with the same name.

Namespace Tools

The functions `dir()`, `globals()`, and `locals()` help with quick namespace introspection:

- `dir(object)` returns a list of attributes that are accessible via the object
- `globals()` returns a dictionary of the attributes currently in the global namespace, along with their values.
- `locals()` returns a dictionary of the attributes in the current local namespace (e.g., within a function), along with their values.

For more information, see "Data model" in Python's official documentation.

It is possible to simulate the more standard behavior by using a special syntax of the `import` statement: `from modu import *`. However, this is generally considered bad

Figure 1-5: Sidebars have boxes around them, to distinguish them from the surrounding body text.

After You're Done Writing

Once you've drafted your whole manuscript, incorporated feedback from tech reviewers, and worked with your editor, the book will go to production. A production editor will be assigned to you and will work with you on the following tasks:

- Formatting the text (adding page breaks, placing figures, etc.)
- Redrawing illustrations to match O'Reilly style
- Copyediting to catch typos, grammatical errors, etc.
- Having a professional indexer write the book's index

The production phase typically takes ~2 months. After that, the production editor will send your book to the printer and you'll officially be a published O'Reilly author!