Codebook: The Structure and Format of Tutorials

Background

This codebook was written after using an initial version of the codebook to analyze 100 tutorials. It was updated occasionally as the rest of the tutorials were analyzed to clarify ambiguities.

Examples have been included as screenshots and links to web pages on the Internet Archive. These formats were chosen to preserve examples even as tutorials are edited or taken down.

Coding Guidelines

Labeling Scheme: *Yes / No / (Unclear)*

To speed up analysis, nearly all variables are *Yes / No / Unclear* labels, with these values:

- **Yes**: This label clearly applies
- **No**: This label clearly does not apply
- **Unclear**: This label might apply, but it's not clear from the tutorial alone

In the vast majority of cases, you should choose *Yes* or *No*, as whether the label applies will be obvious and unambiguous (e.g., whether there are comments in the code).

However, sometimes it may be *Unclear* whether a label applies, because you would need to understand the author's intentions to know for sure. For example, if a tutorial contains a block of code that could be run in IPython shell or placed in a file, you should say it's *Unclear* whether the tutorial contains shell commands. *Unclear* values will make later analysis tricky, so only use them when it's truly a tossup.

Skipping Broken Links

Try to open the web page from among the archived, local files. If this fails, open the original URL in a browser. If this link is broken, skip it. If the link works, find the most recent version of the page on Wayback Machine and analyze that. If the Wayback Machine hasn't archived the page, skip it.

How to Read the Tutorial

Analyze the Full Document

Even if the URL pointed to a specific target on the page (e.g., '#specific-section'), analyze the full document, excluding navigation, ads, and other boilerplate text.

Analyze Code for All Environments

If a tutorial has different instructions for different development environments (e.g., different platforms like OSX vs. Linux vs. Windows, or different tools like Gradle vs. Maven), analyze all unique code blocks across all configurations. If a step is the same for multiple environments, only analyze it once.

Only Analyze the First Language in Parallel-Language Tutorials

There's a class of tutorials we'll call *parallel language tutorials*, where the exact same functionality is shown for multiple languages. The reader is expected to only look at the subset of code blocks in the language they want to use. I have seen this pattern in some Google documentation, where Android programming instructions are provided in both Kotlin and Java. To speed up analysis, only count and review code blocks in the first language of blocks written for multiple language.

Infer What the Author Intended

Some tutorials are old, and have broken links for images, or use broken templating engines. Give tutorials the benefit of the doubt and analyze them as if they were in their most functional state.

Codes

Below is the complete list of codes with examples. Lowest-level headings each correspond to a single variable. There is one row for each variable in the tutorial analysis spreadsheet. Codes have been grouped under high-level headers (e.g., "Is This a Tutorial?", "Code Organization").

I. Is This a Programming Tutorial?

Is The Link Functional?

If the link is broken (e.g., it 404s, or doesn't open the page suggested by the URL), skip it.

Does the Document Contain Two Or More Code Blocks?

A *code block* is one or more lines of code, separated from the rest of the text (i.e. not in the paragraph text). See Number of Code Blocks for more clarifications of what a code block is, with examples.

Examples

This is a code block (from Java Volatile Keyword):

```
public class SharedObject {

    public int counter = 0;

}
```

The next example (from the same web page) is *also a code block*, even though there's a weird apostrophe at the start: it's probably not supposed to be there, so we give the tutorial author the benefit of the doubt. It's also a code block even though the code is pretty vague. From context, it's clearly not pseudocode, but a simple piece of Java code used to make a point.

```
'
int a = 1;
a++;

int b = 2;
b++;
```

No. the Google Cloud Messaging: Overview page has no code blocks.

Does the Document Contain a Sequence of Dependent Code Blocks?

Do at least two code blocks need to be run or reused together to achieve a described outcome? See the examples below for dependence—this includes blocks that build on each other, blocks that reuse code from other blocks, blocks that show blocks from the same file, and blocks that need to be executed one after the other to have their intended effect.

Examples

Yes. These code block are a sequence of dependent blocks. The second block relates to the first block in that it builds on the first block. Anytime you see a step-by-step sequence of code blocks where they show how a file builds up over time, that's a dependent pair of blocks.

```
public class SharedObject {

    public int counter = 0;

}
```

```
public class SharedObject {

    public volatile int counter = 0;

}
```

From Java Volatile Keyword

Yes. In this case, the second block is dependent on the first, as it shows a snippet of first block's code:

```
public class MyClass {
    private int years;
    private int months
    private volatile int days;


    public void update(int years, int months, int days){
        this.years  = years;
        this.months = months;
        this.days   = days;
    }
}
```

```
public void update(int years, int months, int days){
    this.days   = days;
    this.months = months;
    this.years  = years;
}
```

From Java Volatile Keyword

Yes. The second block is dependent on the first, as the second block will not work unless the code in the first block (which adds dependencies to the build configuration) is added:

```
dependencies {
    implementation 'com.android.support:recyclerview-v7:28.0.0'
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<!-- A RecyclerView with some commonly used attributes -->
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

From “Create a List with RecyclerView” (no link available)

Yes. In the following two blocks of shell code *nbrs* is defined in the first block and used in the second block. The second block couldn’t run without the first one.

```
>>> from sklearn.neighbors import NearestNeighbors
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> nbrs = NearestNeighbors(n_neighbors=2, algorithm='ball_tree').fit(X)
>>> distances, indices = nbrs.kneighbors(X)
>>> indices
array([[0, 1],
       [1, 0],
       [2, 1],
       [3, 4],
       [4, 3],
       [5, 4]]...)
>>> distances
array([[0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356]])
```

```
>>> nbrs.kneighbors_graph(X).toarray()
array([[1., 1., 0., 0., 0., 0.],
       [1., 1., 0., 0., 0., 0.],
       [0., 1., 1., 0., 0., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 0., 1., 1.]])
```

From Nearest Neighbors

Yes (Tricky). These code blocks are split up in a strange way (dates are defined as "x" at the top, and used everywhere else as "x" in the "Example" columns of the reference table). That said, the choice of the shared name *x* for initializing and calling methods on the dates implies that a user might be expected to chain together these initializers and accessors.

new Date ( )

This creates a new `Date` object with the value of the current date and time on the browser PC. (We'll use this technique later on to display the current date on the page!) For example:

```
<Code />
x = new Date ( );
```

Method	Description	Example
`getFullYear ()`	Retrieves the year component of the date as a 4-digit number.	`y = x.getFullYear ()`

From Working with Dates

No (Tricky). It's possible these two snippets need to be used together, but it's not clear from the context, mostly from the differing variable names—is 'pane' a JPanel too, and does it have the 'add' method?

You can set a panel's layout manager using the `JPanel` constructor. For example:

```
JPanel panel = new JPanel(new BorderLayout());
```

require you to add the component explicitly; for example, `GroupLayout`. For example, `BorderLayout` requires that you specify the area to which the component should be added (using one of the constants defined in `BorderLayout`) using code like this:

```
pane.add(aComponent, BorderLayout.PAGE_START);
```

From Using Layout Managers

Does It Contain Descriptions of Those Blocks?

Is there at least one sentence of text near to a code block describing the purpose or contents of the code? There must be some description besides the tutorial introduction.

Examples

Yes. The message right above the following code block refers to several tokens in the code, and also alludes to the purpose of the code block (to show a "volatile declaration")

Here is how the `volatile` declaration of the `counter` variable looks:

```
public class SharedObject {

    public volatile int counter = 0;

}
```

From Java Volatile Keyword

Is It About Programming?

Determine the tutorials intended outcome with the guidelines in the "Intended Outcome" section. If it does not match any of the intended outcomes, we will mark this tutorial as "Not Relevant."

II. Should We Skip This Tutorial?

Is This a Duplicate Tutorial?

Most duplicates will have been removed programmatically. However, sometimes, multiple URLs refer to the same document. If this tutorial is a duplicate, do not annotate it a second time.

Is the Tutorial Formatting Unreadable?

If the content of the page are unreadable to the point of making coding inaccurate (which might happen if the tutorial is old and, for example, the links to the styles have broken), skip it.

Examples

From Slide Elements in Different Directions

Is this a Single-Page Tutorial?

To focus analysis on a coherent set of documents, we analyze only single-page tutorials.

Many tutorials appear in collections or series of tutorials. How do we decide if a tutorial in a collection is "single page"? We make an admittedly subjective choice about whether it's likely the content on this page was meant to sometimes be read on its own. If this is page 1 in a getting started tutorial, probably not. If this is a self-contained topic in a long language reference manual, then probably.

If this page is a set of links to other pages without any code, skip it.

Examples
Skip. The How To Create a Wordpress Theme page is a set of links to other pages:

WordPress Theme Tutorial Table of Contents

Ready for a WordPress Theme Tutorial that will show you how to create a powerful WordPress Theme from scratch? Read it from the beginning and code yourself up something awesome.

1. WordPress Theme Tutorial Introduction
2. Theme Development Tools
3. Creating a Theme HTML Structure
4. Template and Directory Structure
5. The Header Template
6. The Index Template
7. The Single Post, Post Attachment, & 404 Templates
8. The Comments Template
9. The Search Template & The Page Template
10. The Archive, Author, Category & Tags Template
11. The Sidebar Template
12. Reset-Rebuild Theme CSS & Define Your Layouts

Skip. "Getting Started on Heroku with Node.js" (no link available) is a series of tutorials designed to be read, by default, across multiple pages.

Skip. Writing your first Django app, part 1 is the first page in a six-page series for building and administering a Django application.

Skip. Customize Table View Cells for UITableView builds on code from a tutorial that comes before it, and it might be tricky to borrow code from it without having the previous tutorial's code to build on.

Don't skip. The author of UICollectionView Tutorial: Getting Started (no link available) links to a tutorial with "enhancements" to the code from this tutorial, though because these are just enhancements and this tutorial comes to a concrete conclusion, this tutorial is probably meant to be read on its own.

Don't skip. Each of the pages of Threading in C# covers a distinct and relatively self-contained topic about threading, using code examples that aren't related to each other.

Don't skip. The C++ Language guides describe themselves as "tutorials" rather than a single tutorial. Code snippets presented in each section are self-contained. It's a language reference and frankly, who reads all the entire reference in a single sitting.

Don't skip. "Animating Interfaces with Core Animation: Part 4" (no link available) covering distinct concepts and presents a distinct project from the other parts of the tutorial.

Is This Tutorial a Reference Manual?

If it's an entire reference manual, skip it. It might contain a tutorial, but it also includes a lot more.

Examples

Yes. the reference manual for c3p0. You can tell that this is a reference manual because it is written by the author of the software, and because it contains everything they want you to know about the software (including installation, configuration, performance, etc.).

c3p0 - JDBC3 Connection and Statement Pooling

version 0.9.5.3

by Steve Waldman <swaldman@mchange.com>

- API docs for c3p0 are here.
- Looking for the definition of configuration properties?
- Looking for advice in using c3p0 with hibernate?

- Download the latest version from c3p0's site on SourceForge
- Follow or fork c3p0 on GitHub.
- Follow @c3p0_jdbc on Twitter.
- This may not be the most recent version of c3p0. See the current CHANGELOG.

Note: Coordinates of this version on the Maven central repository: `[groupId: com.mchange, artifactId: c3p0, version: 0.9.5.3]`

Contents

Yes. Groovy Language Documentation comprises all of the documentation for the Groovy language.

III. Intended Outcome

Tutorials can be used in many ways—to learn new concepts, to build systems, to borrow snippets, and more. Tutorials are typically written with an explicit purpose, or outcomes a reader will experience if they follow the tutorial as directed. After reviewing one-hundred tutorials, we discovered, that these tutorial fell into the following categories. The goal is to determine one most likely intended outcome per tutorial. To do this, look for objectives in the title and the first few paragraphs of the tutorial.

Learn Concepts

The tutorial aims to teach an algorithm or computational idea, independent of a language. Code is provided to simulate, visualize, or concretize ideas, or to provide a reference implementation. If a reader uses the tutorial as intended, they will spend much of their time reading and understanding the text.

Examples

- Understanding Hash Functions and Keeping Passwords Safe: the ideas in this tutorial are mostly presented without consideration of language or tool. The main purpose seems to be to provide an intuition behind secure and efficient password hashing. There are many PHP snippets, though most could be ported easily to another language. PHP isn't mentioned in the title or introduction.
- Optimization: Stochastic Gradient Descent: a chapter in an online course textbook. Visualizations and text describe concepts behind gradient descent. The little bit of Python code feels like concrete pseudocode instead of code the reader will benefit from reusing in their projects.
- A Beginner's Guide to HTTP and REST: Introduces HTTP, REST, common verbs, types of methods, content representations. There is an example application designed to "expose the low-level functionality" of a server, from which a few excerpts are shown.
- Fix Your Timestep!: describes five different schemes for computing time steps. Code isn't concrete and instead reads like pseudocode for describing an algorithm. REaders are unlikely to reuse this code to build a system, improve their own system, or an example system.

Implement Behavior

The tutorial shows how to implement a specific behavior or set of behaviors by writing code. By following the tutorial, the reader will have written functional code that accomplishes some purpose (e.g., analyzes data, implements part of a user interface, serves content over HTTP, etc.). This includes tutorials where most of the code is intended to be run in interactive terminals (e.g., IPython, SQL) and tutorials where most of the code is supposed to be pasted into files and compiled / interpreted.

If the tutorial's stated purpose is "how to do task T with tool U," then the purpose is usually to implement a behavior. If it's to "learn how to use language / tool L," then it's learning the language or tool.

Cookbooks of loosely-related cookbooks may fall into this category, if the main purpose the cookbook is to collect miscellaneous useful snippets and show them all in one place. Check first to see if they should be labeled "Learn Language / Library / Tool."

Examples

- How to Create and Infinite Scrolling Web Gallery: the title clearly indicates this is about implementing a behavior—a scrolling web gallery.
- Build an Android App Using Firebase and the App Engine Flexible Environment: as the title indicates, this tutorial describes building an Android app using some specific components.
- How to Add an API to your Web Service: the purpose of the tutorial is to "give an easy to understand starting point for anyone on a quest to API'ify their web service," implying that the goals is to show how to write code for an API.
- Getting Started with Core Data (No link available) (Tricky): Although the title of the tutorial suggests that this might be about learning a framework, the introduction clarifies that "you'll

write your very first Core Data app," and describes a short and concrete list objectives of what one will be able to do with Core Data by the end of the tutorial.
- Image manipulation and processing using Numpy and Scipy: The organization of this tutorial suggests it's a collection of miscellaneous useful snippets for implementing a bunch of different behaviors. It doesn't seem like the purpose is to teach someone about the libraries, given that there's no overarching narratives or concepts covered in the tutorial.
- DAO tutorial - the data layer: implementation of a system with a data layer. This isn't "Manage Environment," because this is about writing code to implement a complex system from scratch, rather than setting up the environment.
- Spring 3 MVC hello world example: implements non-trivial Spring web server.

Manage Environment

The tutorial describes how to set up or manage a development environment—installing dependencies, using version control, editing configuration files, and tweaking code to run in another environment. Sometimes, this is a subclass of the *Implement Behavior* outcome, where the implemented behavior is a configured system and project that will let the developer create, deploy, or monitor the project. Includes tutorials where existing code is slightly tweaked to port it into a new environment.

Examples

- Migrate to Android Studio: Describes tweaks to a project that let a developer run the project in a different integrated development environment.
- Git Branching - Basic Branching and Merging: The purpose is to show a "simple example" of a version control workflow "that you might use in the real world."
- How to get "create-react-app" to work with your API: Helps reader set up a new and mostly empty React project with 'create-react-app'
- Introduction to Java Development: Command line instructions for setting up a Java program that uses the OpenCV library. There is Java application code, but that's not the focus; the tutorial gives the sense that the Java code is there to test that the setup worked.

Fix a Problem

The tutorial offers instructions on how to detect, comprehend, or fix bugs or errors in existing code.

Examples

- Resolving EACCES permissions errors when installing packages globally: Describes an npm error, and gives instructions on how to fix it.
- Android application (performance and more) analysis tools - Tutorial: Guide describing how to use development tools to for diagnosing Android performance problems.
- Java Debugging with Eclipse - Tutorial: Instructions how to inspect program behavior using Eclipse debugging tools.

Improve a System

The tutorial provides tips, with accompanying code and explanations, that can help the reader improve an existing code project (e.g., improve speed, code quality, clean up style, etc.). In these tutorials, a lot of the code is irrelevant as it's the rewrites to the code that matter.

Examples

- Operation and OperationQueue Tutorial in Swift: In this tutorial, the author gives you a starter project, and describes how to improve its performance using a specific library ("you'll rework the app to add concurrent operations and provide a more responsive interface to the user!")
- Beginning ARC in iOS 5 Tutorial Part 1 (link not available): Guides the reader through rewriting variable references in a project to take advantage of a new reference counting backend. It's basically a guided refactoring.
- CSS Media Queries & Using Available Space: Most of this tutorial describes how to restyle an existing application when it's viewed in different sized windows.
- Bundling and Minification: Practical guidelines for rewriting parts of server code to minimize amount of content downloaded with a web page and thus speed up loading.

Learn Language / Library / Tool

The tutorial will usually describe "getting started with X." These tutorials will often contain lots of conceptual text about the language, library, or tool. Many of the code blocks will have only loosely related goals. Reference documentation may be interspersed with the code.

Examples

- Java Volatile Keyword: introduces a keyword in the Java language
- Providing Constructors for Your Classes: this is a self-contained page of a many-page tutorial dedicate to introducing the Java language.
- Building web apps in WebView: Includes conceptual knowledge about WebView, and the body of the text focuses on many orthogonal use cases of WebView.
- (Tricky) Android SQLite Database Tutorial: while the tutorial shows you how to implement an app, it's describes its own purpose in the first two paragraphs as to "*learn basics of SQLite database* with a realtime example of Notes App."
- (Tricky) Action Mailer Basics: What makes this tricky is that the tutorial intro does describe a concrete behavior that will be implemented ("send and receive mail within a Rails application"), though when reading the intro as a whole it's clear that the point is to introduce Action Mailer, how it works, how to test it, and several use cases.
- (Tricky) Working with Transitions: judging by the tutorial's claim that "it is helpful to understand the system's underlying design" and the focus on how transitions work and pitfalls to using them, we label this as about learning about D3 transitions, rather than implementing animations
- (Tricky) Adding HTML Content to JavaFX Applications: the title refers to a task, and most of the code in the tutorial builds up one large application. However, the purpose of the tutorial is to "introduce the JavaFX embedded browser," and the tutorial includes a brief introduction to the features of the system and the embedded browser API, making this feel more like a general introduction to the API than a description of how to build something quickly

- (Tricky) How to Use Progress Bars: these Java tutorials about Swing components are usually there to provide an introduction to everything that's worth knowing about a component. Beyond just implementing progress bars, there's information about how to select which API to use, and reference documentation.

IV. Code Content

Number of Code Blocks

Our goal with counting code blocks is to understand how many pieces of code an author has to keep track of when writing and maintaining a tutorial. So code blocks can generally be anything, from a single literal value to the contents of an entire file.

Count up the number of blocks of code in the tutorial. A block is only a code block if:

- It appears on a line of its own, or in a table cell of its own, outside of paragraph text
- It is not console output
- It is text, not an image
- It is (mostly) syntactically valid
- It is not a URL. While a URL with query parameters is kind of like a function call, the distinction between a code-like and not-code-like URL is hard, so we just avoid the decision
- It is not an HTTP request (see rationale of last bullet point)
- It is not a fill-in-the-blank exercise (like what W3Schools offers)
- This source code is not pseudocode. If there is pseudocode in a block of code, only count the block if at least half of the lines in block are not pseudocode.

That said, please ignore a lot of formatted reference material that could be mistaken as code (e.g., grammars, method signatures, type declarations, variable declarations, lists of options, list of available commands, lists of regular expression primitives). Follow these rules of thumb:

- method signatures, type declarations, variable declarations, options, operators, and dependencies: include only if copied from other code blocks in the tutorial, or sample project code
- regular expressions and XPath: include if they appear on their own outside a table / list; if they are in a list, only if they are not primitive and the user is encouraged to try them.
- commands: include if they appear on their own outside a table / list; if they are in a list, only include them if the user is encouraged to try them.

Also, ignore pseudocode. The best way to check for pseudocode is to determine if the code could compile, give or take a few small edits. Code that is complete except for abstract placeholders for user-supplied values (e.g., "your-string-here") is not pseudocode.

A few other special cases to keep in mind are:

- Code is still code even if it's hidden on the page by default.
- A block can include code from multiple files, as long as all code shares a border and background.

- Code is still code even if it was produced by a tool or by another piece of code from the tutorial.

Examples

This label has many examples, as it's difficult to come up with inclusive yet precise notion of a code block. Compare to these examples when deciding whether text is a code block.

1x code block. It's text, on a line of its own, and it's not pseudocode.

One might be tempted to have an `ExecutionContext` that runs computations within the current thread:

```
val currentThreadExecutionContext = ExecutionContext.fromExecutor(
  new Executor {
    // Do not do this!
    def execute(runnable: Runnable) { runnable.run() }
})
```

From Futures and Promises | Scala Documentation

2x code blocks. Even though the reader isn't supposed to write the second block of code, it is still syntactically-valid code that the tutorial author potential had to test and maintain.

The for-comprehension in this example is translated to:

```
f.failed.foreach(exc => println(exc))
```

(... and the other example…)

Has the same effect as

```
Future { blockingStuff() }
```

From Futures and Promises | Scala Documentation

1x code block. It's still a code block even if the code was generated by a tool.

WordPress' *.htaccess* file looks like this:

```
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
```

Copy

From Introduction to URL Rewriting.

2x code blocks. "$python setup.py build" and "$sudo python setup.py install" are code blocks, as they show code as text on their own lines (even though the code is not formatted as monospace).

- Go inside **pywebsocket-x.x.x/src/** directory.
- $python setup.py build
- $sudo python setup.py install
- Then read document by –

From HTML5 WebSockets

No *code blocks*. This is a method signature and it's not clear it came from the author's code.

In the application with the custom URL scheme, the app delegate must implement the method with the signature below:

```
- (BOOL)application:(UIApplication *)application
  openURL:(NSURL *)url
  sourceApplication:(NSString *)sourceApplication
  annotation:(id)annotation
```

From The Complete Tutorial on iOS/iPhone Custom URL Schemes

No *code blocks*. The code does not appear on a line of its own.

`Bicycle yourBike = new Bicycle();` invokes the no-argument constructor to create a new `Bicycle` object called `yourBike`.

From Providing Constructors for Your Classes

No *code blocks*. Don't count URLs or HTTP requests as code blocks.

OData defines parameters that can be used to modify an OData query. The client sends these parameters in the query string of the request URI. For example, to sort the results, a client uses the $orderby parameter:

```
http://localhost/Products?$orderby=Name
```

From Supporting OData Query Options in ASP.NET Web API 2

1x code block. Even though two files are shown in this one code block, they share the same border and background, so count them as one code block.

```
# app/mailers/application_mailer.rb
class ApplicationMailer < ActionMailer::Base
  default from: "from@example.com"
  layout 'mailer'
end

# app/mailers/user_mailer.rb
class UserMailer < ApplicationMailer
end
```

From Action Mailer Basics

0x code blocks. This is a list of operators, and wasn't copied from other tutorial code.

expression	can be read as
`*x`	pointed to by `x`
`&x`	address of `x`
`x.y`	member `y` of object `x`
`x->y`	member `y` of object pointed to by `x`
`(*x).y`	member `y` of object pointed to by `x` (equivalent to the previous one)
`x[0]`	first object pointed to by `x`
`x[1]`	second object pointed to by `x`
`x[n]`	`(n+1)`th object pointed to by `x`

From Classes (I)

1x code block. This is *not* pseudocode; it's a command with two abstract placeholders.

```
$ concurrently "command1" "command2"
```

From How to get "create-react-app" to work with your API

0x code block. This list of options was copied from library code, not sample project code.

Options

The following options are available:

```
// the options
$.PFold.defaults = {
  // perspective value
  perspective : 1200,

  // each folding step's speed
  speed : 450,

  // each folding step's easing
  easing : 'linear',
```

From PFold: Paper-Like Unfolding Effect

4x code blocks. Rows 3 and 4 might look like they aren't code because of the obvious typos. We give the author the benefit of the doubt, assume they were meant to be syntactically correct, and count them.

Method	Description
`s.matches("regex")`	Evaluates if `"regex"` matches `s`. Returns only `true` if the WHOLE string can be matched.
`s.split("regex")`	Creates an array with substrings of `s` divided at occurrence of `"regex"`. `"regex"` is not included in the result.
`s.replaceFirst("regex"), "replacement"`	Replaces first occurance of `"regex"` with `"replacement"`.
`s.replaceAll("regex"), "replacement"`	Replaces all occurances of `"regex"` with `"replacement"`.

From Regular expressions in Java - Tutorial

3x code blocks (not 6x). The first image includes 3 code blocks ("ls -l", "ls -a", "ls -R"), while the second one doesn't (for commands "mkdir new-directory-name", "cd directory-name", "cd"). The reader is only encouraged to run the commands from the first image.

Like practically all commands in Linux, you can add options to the "`ls`" command to alter its output or influence its behaviour. An option is preceded by a dash (eg, "`ls -a`"). Try out the following variations of the ls command, to see different forms of output:

ls -l
Produces a "long format" directory listing. For each file or directory, it also shows the owner, group, size, date modified and permissions

ls -a
Lists *all* the files in the directory, including hidden ones. In Linux, files that start with a period (.) are usually not shown.

ls -R
Lists the contents of each subdirectory, their subdirectories etc (recursive).

Directory Commands

Here are the most common commands to work with directories.

mkdir new-directory-name
Creates a new directory, "new-directory-name"

cd directory-name
Goes to the specified directory, making it the "current directory"

cd
When you don't give a directory name, goes to your "home" directory.

From Beginner's Bash

7x code blocks. 7 method usages with concrete arguments.

Call	Files Added or Exception Raised
Include("~/Scripts/Common/*.js")	*AddAltToImg.js, ToggleDiv.js, ToggleImg.js*
Include("~/Scripts/Common/T*.js")	Invalid pattern exception. The wildcard character is only allowed on the prefix or suffix.
Include("~/Scripts/Common/*og.*")	Invalid pattern exception. Only one wildcard character is allowed.
Include("~/Scripts/Common/T*")	*ToggleDiv.js, ToggleImg.js*
Include("~/Scripts/Common/*")	Invalid pattern exception. A pure wildcard segment is not valid.
IncludeDirectory("~/Scripts/Common", "T*")	*ToggleDiv.js, ToggleImg.js*
IncludeDirectory("~/Scripts/Common", "T*", true)	*ToggleDiv.js, ToggleImg.js, ToggleLinks.js*

From Bundling and Minification

2x code blocks. Even though the SQL code is hidden by default, it will be visible after the reader clicks the “SQL” button. The Python and SQL are two code blocks instead of one, because they do not share the same background and border.

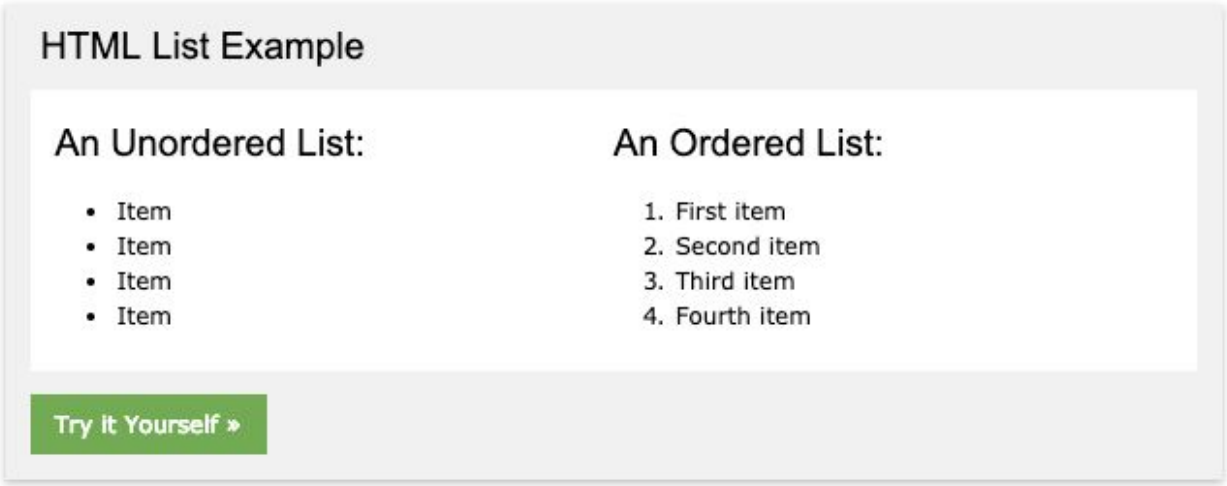
```
>>> session.query(BlogPost).\                                    SQL
...             filter(BlogPost.author==wendy).\
...             filter(BlogPost.keywords.any(keyword='firstpost')).\
...             all()
```

```
SELECT posts.id AS posts_id,
        posts.user_id AS posts_user_id,
        posts.headline AS posts_headline,
        posts.body AS posts_body
FROM posts
WHERE ? = posts.user_id AND (EXISTS (SELECT 1
    FROM post_keywords, keywords
    WHERE posts.id = post_keywords.post_id
        AND keywords.id = post_keywords.keyword_id
        AND keywords.keyword = ?))
(2, 'firstpost')
```

```
[BlogPost("Wendy's Blog Post", 'This is a test', <User(name='wendy', fullname='Wen
```

From Object Relational Tutorial

0x code blocks. There's code on the page at the "Try it Yourself" link, but it's not on the tutorial page.



From HTML Lists

0x code blocks. The following is pseudocode and won't compile without major changes.

```
class class_name {
  access_specifier_1:
    member1;
  access_specifier_2:
    member2;
  ...
} object_names;
```

From Classes (I)

0x code blocks. This is a grammar—a listing of all of the available ways code *could* be written. It resembles code, but should be considered a grammar given the "Syntax" heading right above.

Syntax

```
import defaultExport from "module-name";
import * as name from "module-name";
import { export } from "module-name";
import { export as alias } from "module-name";
import { export1 , export2 } from "module-name";
import { foo , bar } from "module-name/path/to/specific/un-
exported/file";
import { export1 , export2 as alias2 , [...] } from "module-name";
import defaultExport, { export [ , [...] ] } from "module-name";
import defaultExport, * as name from "module-name";
import "module-name";
var promise = import("module-name"); // This is a stage 3 proposal.
```

From import - JavaScript

Linux / cmd.exe / Terminal Commands

The tutorial includes commands for the system shell (e.g., bash, zsh, OSX, Terminal Windows cmd.exe or PowerShell) that the reader may need to execute when following the tutorial. This includes commands to download code, install dependencies, manage code versions, and compile and run code.

Shell commands nat appear as formatted text either in paragraph text, or as block elements. To speed up analysis, we will ignore any shell commands in paragraph text.

If it's likely that system shell commands are supposed to be executed in the shell as a reader follows the tutorial, label *Yes*. If the shell commands are supposed to be written in a source code file, label *No*. If both are equally likely, label *Unclear*.

Examples

2. On the command line, in your home directory, create a directory for global installations:

```
mkdir ~/.npm-global
```

From Resolving EACCES permissions errors when installing packages globally

Other Language Shell Commands

The tutorial includes commands for shells for other languages. There are many languages that provide interactive shells, including SQL, Python, Scala, PHP, and some very small languages and protocols implemented in these tutorials. If it looks like code can be executed by submitting statements to a shell, check to see if a shell exists for that language.

We consider a shell to be a program that takes code as input and interprets / executes that code.

See the instructions for "Linux / cmd.exe / Terminal Commands" for where to look for shell commands, and what labels to assign in what circumstances.

Examples

Yes. This command is meant to be run in a Python or IPython shell. This is clear from the shell prompt on the side (">>>") and the output that directly follows the execution of a command.

```
>>> from sklearn.neighbors import NearestNeighbors
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> nbrs = NearestNeighbors(n_neighbors=2, algorithm='ball_tree').fit(X)
>>> distances, indices = nbrs.kneighbors(X)
>>> indices
array([[0, 1],
       [1, 0],
       [2, 1],
       [3, 4],
       [4, 3],
       [5, 4]]...)
>>> distances
array([[0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356]])
```

From Nearest Neighbors

Unclear. It's unclear whether the two d3 commands below are supposed to be run in an interactive shell (e.g., the JavaScript Debug console) or whether they're meant to be put in a file. There's no context in the text or the code that makes one a better choice than the other.

D3's selection.transition method makes it easy to animate transitions when changing the DOM. For example, to change the text color to red instantaneously, you can select the body element and set the color style:

```
d3.select("body").style("color", "red");
```

To instead animate the change over time, derive a transition:

```
d3.select("body").transition().style("color", "red");
```

From Working with Transitions

File Contents

The tutorial includes at least one code block that, in order to be useful, is supposed to be written to a file. This will be the default for compiled code (e.g., C, Java, etc.), and will depend on context for code for which interactive interpreters exist (e.g., JavaScript, Python). If it is equally likely that the tutorial author intends for the code to be run in a shell as it is for it to be put in a file, label *Unclear*.

Examples

Yes. The following code example must be placed in a file to be useful.

```
public class SharedObject {

    public int counter = 0;

}
```

From Java Volatile Keyword

No. This code is probably not meant to be put in a file (e.g., an Apache config file). In this regular expression tutorial, it's just meant to be read, and maybe run in the regular expression interpreter.

Here in their glory are the Apache regular expression statements we used (maybe you can understand them now, 'cos we cannot)

```
BrowserMatchNoCase [Mm]ozilla/[4-6] isJS
BrowserMatchNoCase MSIE isIE
BrowserMatchNoCase [Gg]ecko isW3C
BrowserMatchNoCase MSIE.((5\.[5-9])|([6-9]|1[0-9])) isW3C
BrowserMatchNoCase (W3C_|[Vv]alidator) isW3C
BrowserMatchNoCase (iphone|[mM]obile) isMob
```

From Regular Expressions - User Guide

V. Code Structure

The codes in this group only apply to code blocks that are supposed to be written to a file, and not to commands that are supposed to be executed in a shell (Linux, cmx, Terminal, Python, SQL, etc.). Analyze the following variables only if the answer to "File Contents" was Yes *or* Unclear.

Snippets

A snippet is, simply, a block of tutorial code that is not meant to stand on its own, but instead should be part of a larger file. You can tell a snippet if the it's supposed to be wrapped in more code to run it. For example, a Java method without the class that contains it is a snippet. You can also tell a snippet if it's missing "include"s, imports, script tags needed to define the used symbols.

If a code block shows a subset of code from a larger code block, it's probably a snippet. That said, in a step-by-step tutorial, if a code block is just an early but complete version of a file containing all the code added to the file so far, it's *not* a snippet.

If a code block shows a line that should get added to a file that *may be* empty and may not be (e.g., a variable definition to add to a .bashrc file), then consider it a snippet.

A tutorial without snippets contains only complete, standalone file listings.

Examples

Yes. The following Java snippet cannot run on its own without being wrapped in a class and method

```
int a = 1;
int b = 2;

a++;
b++;
```

From Java Volatile Keyword

Yes. The following code block...

```
public void update(int years, int months, int days){
    this.days   = days;
    this.months = months;
    this.years  = years;
}
```

is not meant to stand on its own. It is a subset of the code block that appears before it:

```
public class MyClass {
    private int years;
    private int months
    private volatile int days;


    public void update(int years, int months, int days){
        this.years  = years;
        this.months = months;
        this.days   = days;
    }
}
```

From [Java Volatile Keyword]

Yes (Tricky). This code block looks complete, until you realize that it's missing import statements:

KOTLIN JAVA

```
class MyAdapter(private val myDataset: Array<String>) :
        RecyclerView.Adapter<MyAdapter.MyViewHolder>() {

    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, ar
    // you provide access to all the views for a data item in a vi
    // Each data item is just a string in this case that is shown
    class MyViewHolder(val textView: TextView) : RecyclerView.View


    // Create new views (invoked by the layout manager)
    override fun onCreateViewHolder(parent: ViewGroup,
                                    viewType: Int): MyAdapter.MyVi
        // create a new view
        val textView = LayoutInflater.from(parent.context)
                .inflate(R.layout.my_text_view, parent, false) as
        // set the view's size, margins, paddings and layout param
        ...
        return MyViewHolder(textView)
    }

    // Replace the contents of a view (invoked by the layout manag
    override fun onBindViewHolder(holder: MyViewHolder, position:
        // - get element from your dataset at this position
        // - replace the contents of the view with that element
        holder.textView.text = myDataset[position]
    }

    // Return the size of your dataset (invoked by the layout mana
    override fun getItemCount() = myDataset.size
}
```

From “Create a List with RecyclerView” (no link available)

Yes (tricky). Complete, except for the cut (“// ..”) which suggests hidden lines of code.

```
build.gradle

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    // ..

    implementation 'com.android.support:recyclerview-v7:26.1.0'
}
```

From Android SQLite Database Tutorial

Multiple Files

When a reader uses the code from this tutorial, code should be placed in multiple files. If it’s equally likely that code could be placed in a one file as multiple files, label *Unclear*.

Examples

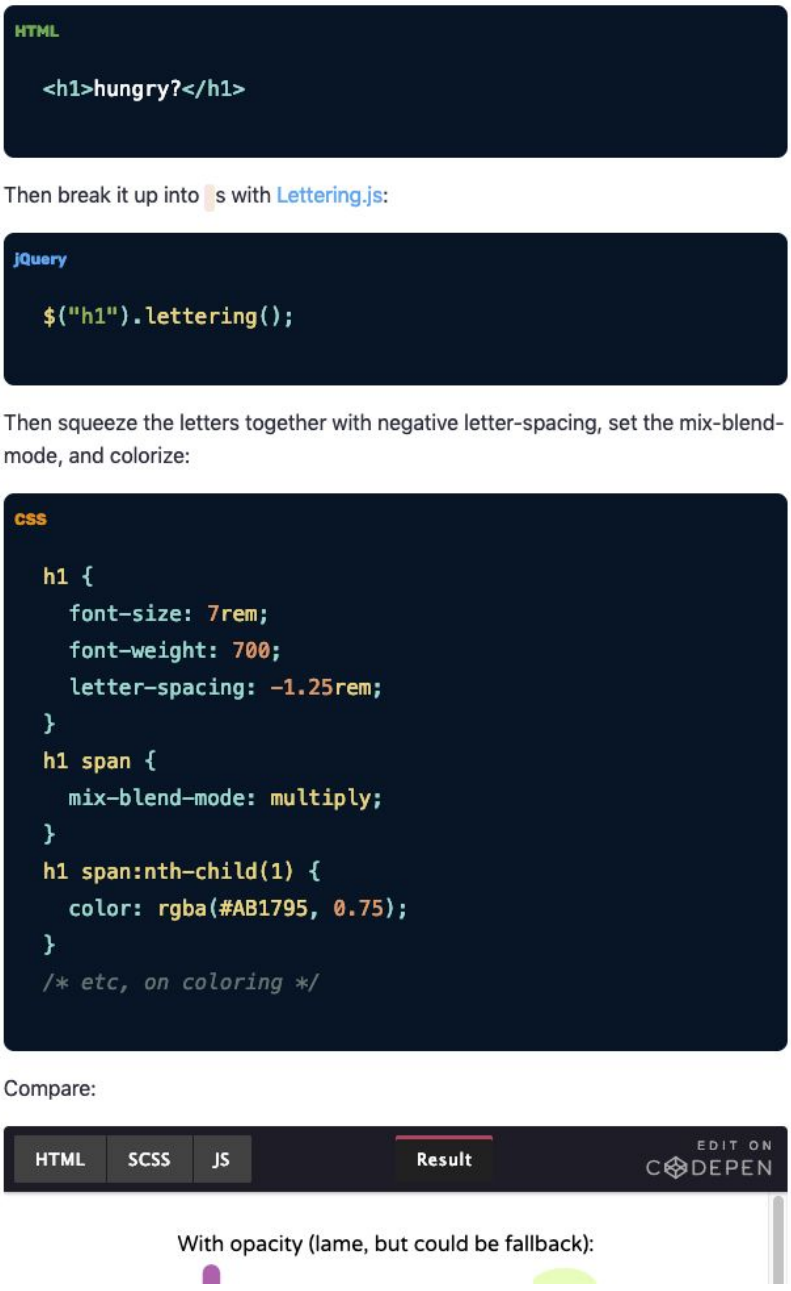
Yes. The following two code blocks from the same tutorial are destined for separate files—if a user reuses this code, they cannot place them in the same file.

```
public class SharedObject {

    public volatile int counter = 0;

}
```

```
public class MyClass {
    private int years;
    private int months
    private volatile int days;


    public void update(int years, int months, int days){
        this.years  = years;
        this.months = months;
        this.days   = days;
    }
}
```

From Java Volatile Keyword

Yes. The CodePen built on code from the code blocks makes it clear the code is intended to be written in three separate files (HTML, SCSS, and JS).



From “Basics of CSS Blending” (no link available)

Unclear (Tricky). These two blocks do not appear to be from the same file, or intended to be put into the same file. On the one hand, the class name implies that both would be put in the same file (“Whatever.java”). On the other hand, the name and the differences between the snippets suggest the author doesn’t care what file the code will go in.

```
class Whatever {
    public static varType myVar = initializeClassVariable();

    private static varType initializeClassVariable() {

        // initialization code goes here
    }
}
```

```
class Whatever {
    private varType myVar = initializeInstanceVariable();

    protected final varType initializeInstanceVariable() {

        // initialization code goes here
    }
}
```

From Initializing Fields

No (Tricky). The tricky part in these code blocks is that they are concrete code that reads like pseudocode, and it's not clear if the reader is actually supposed to reuse the code. If there aren't enough context clues to determine whether the author thought the code is supposed to go in different files, if we think reader would rewrite the same file when reading the tutorial, assume it's one and not multiple files.

From Fix Your Timestep!

Repeated Code

A subset of code is shown twice in the tutorial. In both cases, the subset is from the same place in the same file. This subset of code can be small (e.g., an identifier, short comment, or expression) or large (e.g., a list of statements, a function, a class). Doesn't apply to whitespace.

Examples

Yes. Between these two code blocks, everything from @GetMapping to "}" is repeated. It's from the same place (the "greeting" method) in the same file ("GreetingController.java").

```
    @GetMapping("/greeting")
    public Greeting greeting(@RequestParam(required=false, defaultValue="World") String
        System.out.println("==== in greeting ====");
        return new Greeting(counter.incrementAndGet(), String.format(template, name));
    }
```

```
    @CrossOrigin(origins = "http://localhost:9000")
    @GetMapping("/greeting")
    public Greeting greeting(@RequestParam(required=false, defaultValue="World") String
        System.out.println("==== in greeting ====");
        return new Greeting(counter.incrementAndGet(), String.format(template, name));
    }
```

From [Enabling Cross Origin Requests for a RESTful Web Service]

No (Tricky). These two blocks probably weren't meant to both be included—the second one is a repeat, and likely a typo from when the magazine print tutorial was transcribed to this online version.

```
parser1.y (Rules Section)

nt_program
:
        nt_statement_list nt_quit_statement
;

nt_quit_statement
:
        T_QUIT T_SEMICOLON
;

nt_statement_list
:
        /* empty */
|
        nt_statement
|
        nt_statement_list nt_statement
;

        //code omitted

nt_number
:
        T_NUMBER_LITERAL
|
        T_NUMBER_IDENTIFIER
|
        nt_number T_MULTIPLY nt_number
|
        nt_number T_ADD nt_number
|
        T_OPEN_PAREN nt_number T_CLOSE_PAREN
|
        T_NUMBER_IDENTIFIER T_OPEN_BRACE
                        nt_expression_list T_CLOSE_BRACE
;

        //code omitted
```

A rule describes to the parser when it can do a reduction. The result of the reduction (a non-terminal) is given first followed by a colon (:), then there are a sequence of patterns separated by vertical bars (|) and finally a semicolon (;). After each pattern, there is an optional block of code. Some of the blocks in parser1.y contain comments about the Miniset grammar.

```
parser1.y (Rule Section)

nt_program
:
        nt_statement_list nt_quit_statement
;

nt_quit_statement
:
        T_QUIT T_SEMICOLON
;

nt_statement_list
:
        /* empty */
|
        nt_statement
|
        nt_statement_list nt_statement
;
        //code omitted
nt_number
:
        T_NUMBER_LITERAL
|
        T_NUMBER_IDENTIFIER
```

From Using Flex and Bison

No (Tricky). Even though parser1.y is cloned into parser2.y and therefore the nt_statement_list declaration is “repeated,” we ignore this instance of likely repeated code. For the sake of a clean and unambiguous definition of repeated code, repeated and rewritten code is supposed to come from the same place in the same file, so repeated code in forks of a file don’t count

```
parser1.y (Rules Section)

nt_program
:
        nt_statement_list nt_quit_statement
;

nt_quit_statement
:
        T_QUIT T_SEMICOLON
;

nt_statement_list
:
        /* empty */
|
        nt_statement
|
        nt_statement_list nt_statement
;
```

```
parser2.y

nt_statement_list
:
        /* empty */
|
        nt_statement_list nt_statement
;
```

From Using Flex and Bison

Rewritten Code

Code shown in one block is rewritten in another block. In other words, you see different versions of code intended to be written in the same place of the same file.

If it's unclear whether this code was meant to run in the shell or a file, mark it as "No." However, if it's clear it's supposed to go in a file, but it's unclear if it's supposed to go in the *exact* same file as a previous block, count it as a rewrite if it's described as an update to an earlier block.

Some common examples of rewrites include:

- deletions (removing a line shown in an earlier block)
- updates (changing part of a line shown in an earlier block, including comments, and including code that takes the place of a placeholder)
- statements in a method have been reordered
- wrapping old code in a new structure (e.g., moving code into a method or an if-block)
- small, additive changes to a statement (e.g., adding comments to a line, or adding an argument to a function call)
- any of the above changes caused by running code or a tool

Rewrites do not include:

- rewrites that are probably just typos
- changes that solely update whitespace
- cuts (i.e. just 'hiding' code from earlier steps)
- equivalents (i.e. snippets with the same behavior as a subset of earlier code) should only be considered a rewrite if there's context in the code around the equivalent, or indications in the text, that the file from the earlier code snippet is being rewritten

Examples

Yes. Between these code blocks, a statement is changed. Even though this is an addition without deleting code, we count it anyway as it's a fine-grained update that changes a statement,

```
public class SharedObject {

    public int counter = 0;

}
```

```
public class SharedObject {

    public volatile int counter = 0;

}
```

From Java Volatile Keyword

Yes. A placeholder from an earlier step was overwritten.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    // Add other library dependencies here (see the next step)
}
```

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

    // Google Play Services
    compile 'com.google.android.gms:play-services:9.8.0'
```

From Migrate to Android Studio

Yes. Existing code is moved into a new structure (an if-block).

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
      sourceApplication:(NSString *)sourceApplication annotation:(id)annotatio
{
  NSLog(@"Calling Application Bundle ID: %@", sourceApplication);
  NSLog(@"URL scheme:%@", [url scheme]);
  NSLog(@"URL query: %@", [url query]);

  return YES;
}
```

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
      sourceApplication:(NSString *)sourceApplication annotation:(id)annotatic
{
  // Check the calling application Bundle ID
  if ([sourceApplication isEqualToString:@"com.3Sixty.CallCustomURL"])
  {
    NSLog(@"Calling Application Bundle ID: %@", sourceApplication);
    NSLog(@"URL scheme:%@", [url scheme]);
    NSLog(@"URL query: %@", [url query]);

    return YES;
  }
  else
    return NO;
}
```

From The Complete Tutorial on iOS/iPhone Custom URL Schemes

Yes. It appears that the code introduced in the first block is rewritten as it's refactored into the final program. We can infer that the line is meant to be the same line by seeing the second code block's description that it's putting "all of that together," and it's also clear from the fact that both GetMethod instances are initialized with the same data.

```
HttpMethod method = new GetMethod("http://www.apache.org/");
```

When we put all of that together plus a little bit of glue code we get the program below.

```
import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.methods.*;
import org.apache.commons.httpclient.params.HttpMethodParams;

import java.io.*;

public class HttpClientTutorial {

  private static String url = "http://www.apache.org/";

  public static void main(String[] args) {
    // Create an instance of HttpClient.
    HttpClient client = new HttpClient();

    // Create a method instance.
    GetMethod method = new GetMethod(url);
```

From HttpClient Tutorial

Yes. The order of calling `webEngine.load` and `toolBar = new HBox()` has changed.

```
// load the home page
        webEngine.load("http://www.oracle.com/products/index.html");

// create the toolbar
        toolBar = new HBox();
        toolBar.getStyleClass().add("browser-toolbar");
        toolBar.getChildren().addAll(hpls);

        //add components
        getChildren().add(toolBar);
        getChildren().add(browser);
```

```
        // create the toolbar
        toolBar = new HBox();
        toolBar.setAlignment(Pos.CENTER);
        toolBar.getStyleClass().add("browser-toolbar");
        toolBar.getChildren().addAll(hpls);
        toolBar.getChildren().add(createSpacer());

        //set action for the button
        showPrevDoc.setOnAction(new EventHandler() {
            @Override
            public void handle(Event t) {
                webEngine.executeScript("toggleDisplay('PrevRel')");
            }
        });

        // process page loading
        webEngine.getLoadWorker().stateProperty().addListener(
            new ChangeListener<State>() {
                @Override
                public void changed(ObservableValue<? extends State> ov,
                    State oldState, State newState) {
                        toolBar.getChildren().remove(showPrevDoc);
                        if (newState == State.SUCCEEDED) {
                            if (needDocumentationButton) {
                                toolBar.getChildren().add(showPrevDoc);
                            }
                        }
                    }
                }
        );

        // load the home page
        webEngine.load("http://www.oracle.com/products/index.html");
```

Yes. The second block fills a placeholder from the first block with “var listener…”

KOTLIN JAVA

```
public class FragmentA : ListFragment() {
    ...
    // Container Activity must implement this interface
    interface OnArticleSelectedListener {
        fun onArticleSelected(articleUri: Uri)
    }
    ...
}
```

Then the activity that hosts the fragment implements the `OnArticleSelectedListener` interface and overrides `onArticleSelected()` to notify fragment B of the event from fragment A. To ensure that the host activity implements this interface, fragment A's `onAttach()` callback method (which the system calls when adding the fragment to the activity) instantiates an instance of `OnArticleSelectedListener` by casting the `Activity` that is passed into `onAttach()`:

KOTLIN JAVA

```
public class FragmentA : ListFragment() {

    var listener: OnArticleSelectedListener? = null
    ...
    override fun onAttach(context: Context) {
        super.onAttach(context)
        listener = context as? OnArticleSelectedListener
        if (listener == null) {
            throw ClassCastException("$context must implement OnArticle
        }

    }
    ...
}
```

From Fragments

Yes. While it's unclear if the code from these two blocks are meant to be put in the same file, the second block overwrites code from the first, and is offered as a refinement to the first.

```
double t = 0.0;
double dt = 1.0 / 60.0;

while ( !quit )
{
    integrate( state, t, dt );
    render( state );
    t += dt;
}
```

```
    double t = 0.0;

    double currentTime = hires_time_in_seconds();

    while ( !quit )
    {
        double newTime = hires_time_in_seconds();
        double frameTime = newTime - currentTime;
        currentTime = newTime;

        integrate( state, t, frameTime );
        t += frameTime;

        render( state );
    }
```

From Fix Your Timestep!

Yes. Between these code blocks, a comment has been removed. On the one hand, this might have been an oversight of the author. On the other hand, it's hard to infer the author's intention in this case and they may well have wanted to remove the comment, so to be safe we label *Yes*.

```
import info.androidhive.sqlite.database.model.Note;

/**
 * Created by ravi on 15/03/18.
 */

public class DatabaseHelper extends SQLiteOpenHelper {
```

```
import info.androidhive.sqlite.database.model.Note;

public class DatabaseHelper extends SQLiteOpenHelper {
```

From Android SQLite Database Tutorial

No. Between these code blocks, new statements are *added*, but not at the sub-statement level, and no code is rewritten or deleted. So the code is not "rewritten" between this pair of blocks.

```
public class MyClass {
    private int years;
    private int months
    private volatile int days;


    public void update(int years, int months, int days){
        this.years  = years;
        this.months = months;
        this.days   = days;
    }
}
```

```
public class MyClass {
    private int years;
    private int months
    private volatile int days;

    public int totalDays() {
        int total = this.days;
        total += months * 30;
        total += years * 365;
        return total;
    }

    public void update(int years, int months, int days){
        this.years  = years;
        this.months = months;
        this.days   = days;
    }
}
```

From Java Volatile Keyword

No. The comment is rewritten, but it's a trivial enough difference without a real semantic change that I'm assuming this is a copy-and-paste error.

```
<!-- Goolge Maps API Key -->
<meta-data
      android:name="com.google.android.maps.v2.API_KEY"
      android:value="AIzaSyBZMlkOv4sj-M5JO9p6wksdax4TEjDVLgo" />
```

```
        <!-- Goolge API Key -->
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyBZMlkOv4sj-M5JO9p6wksdax4TEjDVLg
```

From Android working with Google Maps v2

No. The second code block may just be a standalone snippet; there's no context to suggest that the second code block should be used to rewrite the first code block.

C# Copy

```
[Queryable(PageSize=10)]
public IQueryable<Product> Get()
{
    return products.AsQueryable();
}
```

C# Copy

```
[Queryable(AllowedQueryOptions=
    AllowedQueryOptions.Skip | AllowedQueryOptions.Top)]
```

From Supporting OData Query Options in ASP.NET Web API 2

No. These are two equivalents implementations, this is not a rewrite.

will do nothing, as shown in the following:

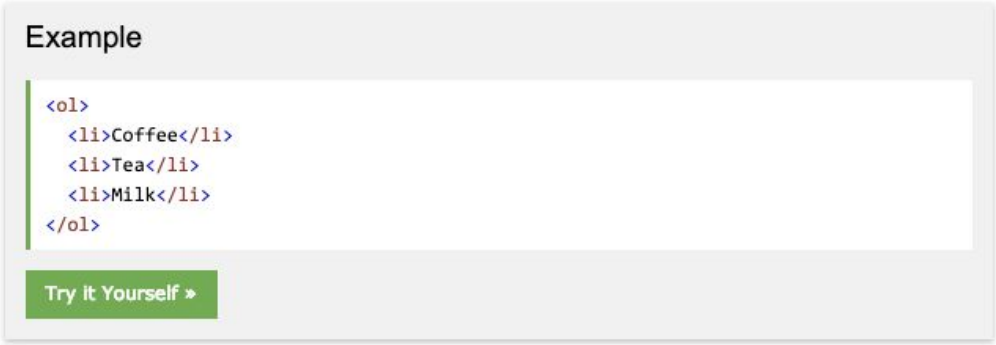
```
implicit val ec = ExecutionContext.fromExecutor(
                    Executors.newFixedThreadPool(4))
Future {
  blocking { blockingStuff() }
}
```

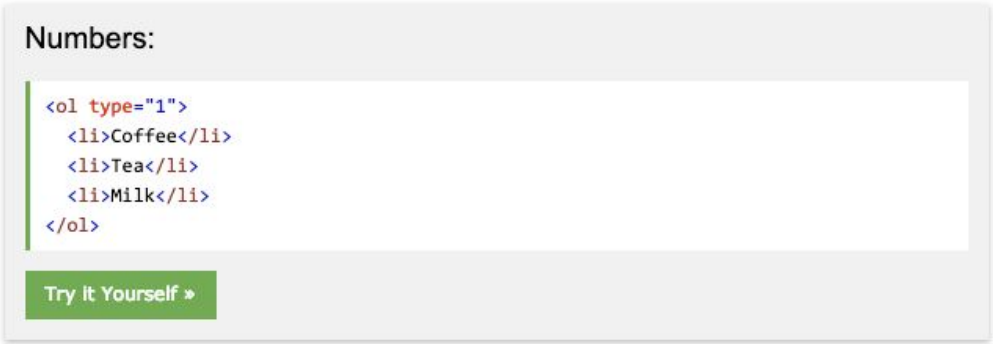
Has the same effect as

```
Future { blockingStuff() }
```

From [Futures and Promises | Scala Documentation]

No. These examples are in different sections of the tutorial. While the second code block on the surface refines a line that's in the first block, it's not clear if it's meant to, or if it's an independent example.





From [HTML Lists]

VI. Code Style

These codes only apply to code blocks that are supposed to be written to a file, and not to commands that are supposed to be executed in a shell (Linux, cmx, Terminal, Python, SQL, etc.). Analyze the following variables only if the answer to "File Contents" was Yes *or* Unclear.

Comments

There are comments in the code. This includes *any* type of comment, including comments that are only meant to be read by machines, like crunch-bang directives ("#! /usr/bin/env python") at the top of files.

Examples

Yes. The following code block includes descriptive comments of what the codes does:

```
        // Provide a reference to the views for each data item
        // Complex data items may need more than one view per item, ar
        // you provide access to all the views for a data item in a vi
        // Each data item is just a string in this case that is shown
        class MyViewHolder(val textView: TextView) : RecyclerView.Viev
```

From “Create a List with RecyclerView” (no link available)

Yes (Tricky). Even something as simple as a couple of ellipses still counts as a comment. It’s a comment as long as this part of the code would be processed as a comment by a compiler or interpreter.

```
build.gradle

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    // ..

    implementation 'com.android.support:recyclerview-v7:26.1.0'
}
```

From Android SQLite Database Tutorial

Yes (Tricky). Comments that only contain labels still count as comments.

```
override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -
> UITableViewCell {
  let cell = tableView.dequeueReusableCellWithIdentifier("CellIdentifier", forIndexPath: index-
Path) as! UITableViewCell

  //1
  if cell.accessoryView == nil {
    let indicator = UIActivityIndicatorView(activityIndicatorStyle: .Gray)
    cell.accessoryView = indicator
  }
  let indicator = cell.accessoryView as! UIActivityIndicatorView
```

From Operation and OperationQueue Tutorial in Swift

Labels

Numeric or alphabetic labels in the code. Only count these if they are referenced from the text. Only count labels that have a number of a simple letter.

Examples

Yes. See the comments "//1", "//2", and "//3" in the code below. These numeric labels are referenced from the text that comes after the code block.

```
override func viewWillAppear(_ animated: Bool) {
  super.viewWillAppear(animated)

  //1
  guard let appDelegate =
    UIApplication.shared.delegate as? AppDelegate else {
      return
  }

  let managedContext =
    appDelegate.persistentContainer.viewContext

  //2
  let fetchRequest =
    NSFetchRequest<NSManagedObject>(entityName: "Person")

  //3
  do {
    people = try managedContext.fetch(fetchRequest)
  } catch let error as NSError {
    print("Could not fetch. \(error), \(error.userInfo)")
  }
}
```

COPY

Step by step, this is what the code does:

1. Before you can do anything with Core Data, you need a managed object context. Fetching is no different! Like before, you pull up the application delegate and grab a reference to its persistent container to get your hands on its `NSManagedObjectContext`.
2. As the name suggests, `NSFetchRequest` is the class responsible for fetching from Core Data. Fetch requests are both powerful and flexible. You can use fetch requests to fetch a set of objects meeting the provided criteria (i.e. give me all employees living in Wisconsin and have been with the company at least three years), individual values (i.e. give me the longest name in the database) and more.

From "Getting Started with Core Data Tutorial" (Link not available)

Cuts

Code shown in one block (or present in a sample project) is hidden in a later block, behind a symbol that suggests text is hidden. The symbol can be an ellipsis ("...") or something else (e.g., a comment). If there is no symbol hiding code (i.e. if the other code is just missing), then this is not a cut.

Examples

Yes. An ellipses in the comment "// .." suggests that some of the code before the last "implementation" line has been hidden. The tutorial starts with the reader having no code at all, and there's no indication in the comment that the reader is supposed to add extra dependencies in place of the "// ..", so we can infer that this is not a placeholder for a reader's code or for later code in the tutorial; instead, there are probably already dependencies in the build.gradle file that are being hidden.

```
build.gradle
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    // ..

    implementation 'com.android.support:recyclerview-v7:26.1.0'
}
```

From Android SQLite Database Tutorial

Yes. This is a cut: the text right below the code indicates that the code has been omitted.

```
UIBezierPath *trackPath = [UIBezierPath bezierPath]
[trackPath moveToPoint:CGPointMake(160, 25)];
// ... Path drawing operations ...
```

I've omitted the actual path drawing code because it's pretty verbose, but you can download the source code and sift

From "Animating Interfaces with Core Animation: Part 4" (no link available).

Yes. From the context of the article, it seems likely that "// code omitted" refers to actual source code that the author had actually written when the author was writing the article, and that perhaps was made

available along with the magazine where the tutorial was originally published. It doesn't seem like a placeholder for user-provided code, but instead code omitted to keep the code block brief.

> The section between the first %% and the second %% is called the Rules Section. This section describes the non-terminals in the grammar (those things which are built from tokens and other non-terminals). The Rules Section in parser1.y is long and I have omitted a number of rules. The lines displayed below are representative of what you find in a rules section and they contain all the code of interest (including the bugs).

```
parser1.y (Rules Section)

nt_program
:
        nt_statement_list nt_quit_statement
;

nt_quit_statement
:
        T_QUIT T_SEMICOLON
;

nt_statement_list
:
        /* empty */
|
        nt_statement
|
        nt_statement_list nt_statement
;

        //code omitted
```

From [Using Flex and Bison]

No. At first glance, "Lines removed for clarity" makes is sound like lines were removed from the code. Actually, there is not code that is removed for clarity from earlier in the tutorial, and there is no sample project that includes that is excluded here. This should instead be read as "your file may have something else here," or "you will have code here," which is a placeholder and not a cut.

```xml
<system.web>
    <compilation debug="true" />
    <!-- Lines removed for clarity. -->
</system.web>
```

No. Even though code is hidden in the second snippet, it isn't hidden behind a symbol.

```
6  my_http.createServer(function(request,response){
7    var my_path = url.parse(request.url).pathname;
8    var full_path = path.join(process.cwd(),my_path);

1  my_http.createServer(function(request,response){}).lis
```

From Beginner's Guide to Node.js (Server-side JavaScript)

Placeholders

Placeholders meant to be filled in with later code, user-supplied code, or future generated code. They are explicit indications, within code, of where code will be added in the future, for example:

- an ellipsis ("...")
- a comment ("// Insert your code here")
- pseudocode describing what code should do
- a method name that suggests an action taking place somewhere else
- or a prominently styled (caps, highlighted, etc.) variable or literal value.

Note that a placeholder need not be filled later in the tutorial to count as a placeholder. This label is based on the "abstract placeholders" described in Buse and Weimer's "Synthesizing API Usage Examples."

Examples

Yes. A comment and the ellipsis suggests a place where the reader should write their own code:

```
        // set the view's size, margins, paddings and layout param
        ...
        return MyViewHolder(textView)
    }
```

From "Create a List with RecyclerView" (no link available)

Yes. A comment suggests code the reader may want to add in the future:

```
    private void toggleEmptyNotes() {
        // you can check notesList.size() > 0

        if (db.getNotesCount() > 0) {
            noNotesView.setVisibility(View.GONE);
        } else {
            noNotesView.setVisibility(View.VISIBLE);
        }
    }
```

From Android SQLite Database Tutorial

Yes. A fake function name like “doSomething” doesn’t refer to visible code, and is probably meant to be replaced with a reader’s own code.

```
val consumer = Future {
  startDoingSomething()
  f foreach { r =>
    doSomethingWithResult()
  }
}
```

From Futures and Promises | Scala Documentation

Yes. The text and file contents make it clear that JSON_FILE_NAME and FIREBASE_URL are placeholders for values the reader will supply.

7. Edit `src/main/webapp/WEB-INF/web.xml` and edit the initialization parameters as follows:

    - Replace `JSON_FILE_NAME` with the name of the JSON key file you downloaded.
    - Replace `FIREBASE_URL` with the Firebase URL you noted earlier.

```
<init-param>
  <param-name>credential</param-name>
  <param-value>/WEB-INF/JSON_FILE_NAME</param-value>
</init-param>
<init-param>
  <param-name>databaseUrl</param-name>
  <param-value>FIREBASE_URL</param-value>
</init-param>
```

From Build an Android App Using Firebase and the App Engine Flexible Environment

Yes. The comments suggest a specific piece of code that's supposed to be rewritten.

Your **`DocumentRoot`** **may be different, of course..**

```
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/var/www/htdocs">
#
```

From .htaccess tricks and tips

No (Tricky). Even though the reader should probably replace 'example.com' with their own host name, we only mark a placeholder if there's something that makes it stylistically distinct from the rest of the code---like capital letters, or being in a comment, or long descriptive variable names.

```
<%= url_for(host: 'example.com',
            controller: 'welcome',
            action: 'greeting') %>
```

From Action Mailer Basics

No (Tricky). This one is subtle: generated "<li>" code is never actually added to the HTML to replace the comment—it's added to the DOM, which could be viewed as HTML, but it's not the same document.

```
            <ul>
                <!-- The file uploads will be shown here -->
            </ul>
```

the transferred files. You can see the markup generated for a file upload below:

```
<li class="working">
    <input type="text" value="0" data-width="48" data-height="48" data-fgColor="#0788a5" data
    <p>Sunset.jpg <i>145 KB</i></p>
    <span></span>
</li>
```

Changed Code Has a Special Style

At least one code block has some code that was styled to show what code was added, updated, or removed to that block, relative to an earlier version of the code. This earlier version of the code need not have been shown before in the tutorial.

Examples

Yes. As confirmed by the surrounding text, the bolded code represents new code added to a file.

A table model can have a set of listeners that are notified whenever the table data changes. Listeners are instances of `TableModelListener`. In the following example code, `SimpleTableDemo` is extended to include such a listener. New code is in bold.

```
import javax.swing.event.*;
import javax.swing.table.TableModel;

public class SimpleTableDemo ... implements TableModelListener {
    ...
    public SimpleTableDemo() {
        ...
        table.getModel().addTableModelListener(this);
        ...
    }

    public void tableChanged(TableModelEvent e) {
        int row = e.getFirstRow();
        int column = e.getColumn();
        TableModel model = (TableModel)e.getSource();
        String columnName = model.getColumnName(column);
        Object data = model.getValueAt(row, column);

        ...// Do something with the data...
    }
    ...
}
```

From How to Use Tables

Yes. In this tutorial, the code added in each step is opaque, and the rest of the code is transparent.

Example app-level build.gradle (excerpt)

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.google.android.gms:play-services-ads:15.0.0
}
```

Add the line in bold above, which instructs Gradle to pull in the latest version of the Mobile Ads SDK. Once that's done, save the file and perform a Gradle sync.

From Get Started | Google Mobile Ads SDK for Android

Yes. The second piece of code has a green color to show where code has been rewritten.

Iterating over a collection is uglier than it needs to be. Consider the following method, which takes a collection of timer tasks and cancels them:

```
void cancelAll(Collection<TimerTask> c) {
    for (Iterator<TimerTask> i = c.iterator(); i.hasNext(); )
        i.next().cancel();
}
```

The iterator is just clutter. Furthermore, it is an opportunity for error. The iterator variable occurs three times in each loop: that is two chances to get it wrong. The for-each construct gets rid of the clutter and the opportunity for error. Here is how the example looks with the for-each construct:

```
void cancelAll(Collection<TimerTask> c) {
    for (TimerTask t : c)
        t.cancel();
}
```

From The For-Each Loop

Yes. Red font color highlights an important changed line in the code.

Another thing you can do is get a slice of an array.

```
#!/usr/bin/ruby
presidents = ["Ford", "Carter", "Reagan", "Bush1", "Clinton", "Bush2"]
p123=presidents[1..3]
p123.each { |i| print i, "\n"}
```

Another way to slice an array is with a start and a count instead of a range. The following is another way to write basically the same code as the preceding code:

```
#!/usr/bin/ruby
presidents = ["Ford", "Carter", "Reagan", "Bush1", "Clinton", "Bush2"]
p123=presidents[1,3]
p123.each { |i| print i, "\n"}
```

From Ruby Basic Tutorial

Notable Code Has a Special Style

At least one code block has some code that was styled to call attention to important elements of the code. You may have to look at the surrounding text to tell whether the code was styled to call attention to the important parts, or for some other reason.

If the styling highlights a change in the code relative to an earlier code version, instead use the label "Changed Code Has a Special Style."

Examples

Yes. An important line is bolded:

Setting up a combo box as an editor is simple, as the following example shows. The bold line of code sets up the combo box as the editor for a specific column.

```
TableColumn sportColumn = table.getColumnModel().getColumn(2);
...
JComboBox comboBox = new JComboBox();
comboBox.addItem("Snowboarding");
comboBox.addItem("Rowing");
comboBox.addItem("Chasing toddlers");
comboBox.addItem("Speed reading");
comboBox.addItem("Teaching high school");
comboBox.addItem("None");
sportColumn.setCellEditor(new DefaultCellEditor(comboBox));
```

From How to Use Tables

No. The style for line "p123=presidents[1,3]" shows what has changed relative to an earlier version of the code, so instead use the label "Changed Code Has a Special Style."

Another thing you can do is get a slice of an array.

```
#!/usr/bin/ruby
presidents = ["Ford", "Carter", "Reagan", "Bush1", "Clinton", "Bush2"]
p123=presidents[1..3]
p123.each { |i| print i, "\n"}
```

Another way to slice an array is with a start and a count instead of a range. The following is another way to write basically the same code as the preceding code:

```
#!/usr/bin/ruby
presidents = ["Ford", "Carter", "Reagan", "Bush1", "Clinton", "Bush2"]
p123=presidents[1,3]
p123.each { |i| print i, "\n"}
```

From Ruby Basic Tutorial

VII. Prose

Task-Relevant Headers

Headers are lines of text with a distinct style from the rest of the text, appearing either at the start of a paragraph, or on a line of their own. Does the tutorial include headers (at least one)? If so, do these headers provide some indication of what the code below them does?

Say *Yes*, as long as headers provide *some* information about what the code below them does (including brief descriptions, e.g., "Controller," or language names or file names for multi-language tutorials). Ignore headers for sections that don't include any code. Ignore the tutorial title. And ignore headers that are only file names.

Examples

Yes. "Add the support library" in "Create a List with RecyclerView" (no link available).

Yes. "2.1 Writing SQLite Helper Class" in Android SQLite Database Tutorial.

Yes. "Step 2) Configure Email" is a header. Even though it is the same size as the body text, it has a prominent style distinct from the rest of the text, and on a line of its own.

> **Step 2) Configure Email:**
>
> After the Account and the Profile are created successfully, we need to configure the Database Mail. To configure it, we need to enable the Database Mail XPs parameter through the sp_configure stored procedure, as shown here:

From SQL SERVER – 2008 – Configure Database Mail – Send Email From SQL Database

Yes (Tricky). "Instruction Reordering Challenges" in Java Volatile Keyword. This header provides *some* indication of what the code below it is for—specifically that the code will show examples of code for which instruction reordering will and will not be an issue.

Yes (Tricky). "The jQuery Way" in Sticky Position (Bar) with CSS or jQuery. This header provides some indication of what's in the code below, specifically that this is the jQuery implementation of the sticky functionality (and not the CSS implementation).

No. Custom ListView items and adapters has no headers at all.

No. The only headers accompanying code in Html5 File Upload with Progress are "The Minimalistic Solution" and "Let's get started," which convey nothing about the code in those sections.

No. All of the code in Tutorial: Real-time chat with SignalR 2 is under headings of "Set up the project" or "Examine the code." These tell you nothing about the contents of the code.

VIII. Generated Outputs

Console Output

Console output or errors generated by running the code. This includes console output and errors that appear right below executed shell commands in the same code block. Ignore console output that appears in text paragraphs or in code comments.

This includes screenshots of console output, if the only thing showing in those screenshots is the console. If other windows are showing in the screenshot, code this as "Images of Output."

Animated GIFs should be considered videos, not images.

Examples

Yes. Console output appears within a block as a response to a shell command.

```
>>> nbrs.kneighbors_graph(X).toarray()
array([[1., 1., 0., 0., 0., 0.],
       [1., 1., 0., 0., 0., 0.],
       [0., 1., 1., 0., 0., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 0., 1., 1.]])
```

From Nearest Neighbors

Yes (Tricky). PHP echo's and error output to otherwise blank HTML pages count, as this is the typical way of emulating console output and errors for PHP.

```
    echo 'Connected to database<br />';

    /*** INSERT data ***/
    $count = $dbh->exec("INSERT INTO animals(animal_type, animal_name

    /*** echo the number of affected rows ***/
    echo $count;

    /*** close the database connection ***/
    $dbh = null;
    }
catch(PDOException $e)
    {
    echo $e->getMessage();
    }
?>
```

The output of the script above will look like this:

```
Connected to database
1
```

From "Introduction to PHP-PDO" (no link available)

No. If console output or errors are only described in the text, they don't count.

In the case that the future fails, the caller is forwarded the exception that the future is failed with. This includes the `failed` projection– blocking on it results in a `NoSuchElementException` being thrown if the original future is completed successfully.

From Futures and Promises | Scala Documentation

No (Tricky). If console output appears in code comments, it doesn't count.

This would generate URLs like this:

```
echo GenerateUrl ("Pâtisserie (Always FRESH!)"); //returns "patisserie-always-fresh"
```

Copy

From Introduction to URL Rewriting

No (Tricky). A response returned by a web server is not console output. You may want to make a comment about this tutorial that you observed another type of generated output, however.

Now that the service is up, visit http://localhost:8080/greeting, where you see:

```
{"id":1,"content":"Hello, World!"}
```

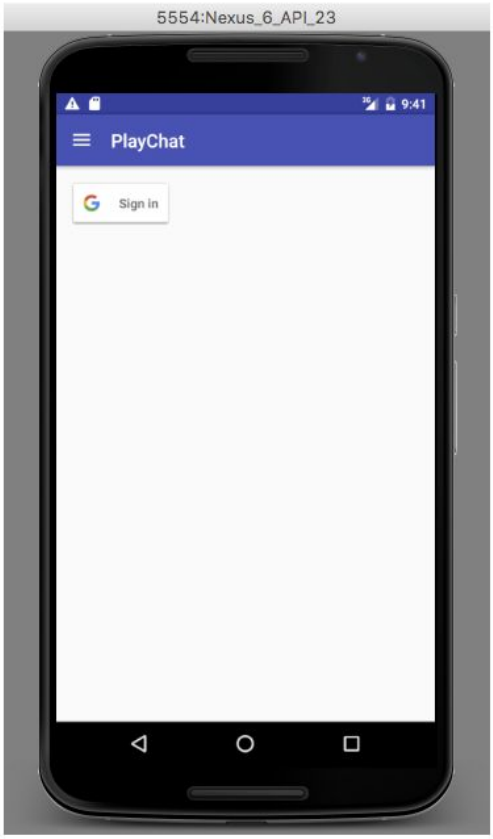
From Enabling Cross Origin Requests for a RESTful Web Service

Images of Output

An embedded image that required the author to run code from the tutorial. Examples include screenshots of user interfaces built in the tutorial, and visualizations generated from code in the tutorial.

Examples

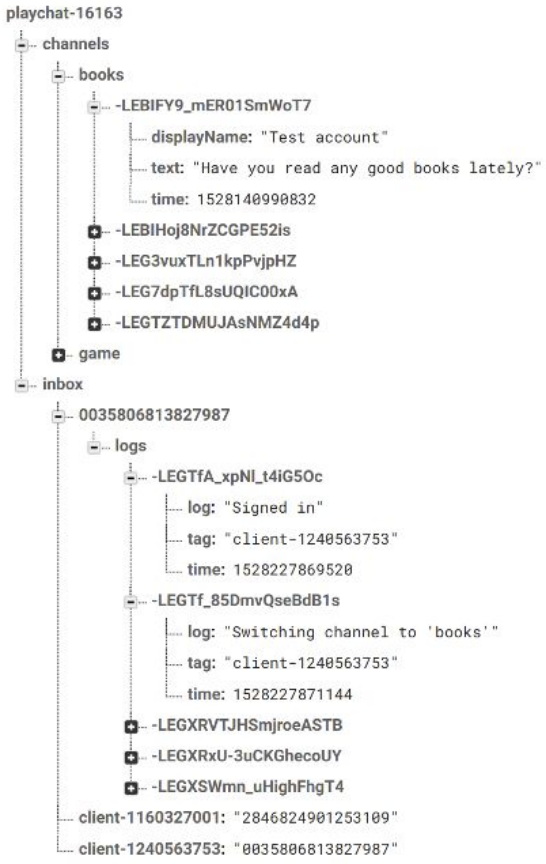
Yes. This is a screenshot of an app, the code of which is the focus of the tutorial.



From [Build an Android App Using Firebase and the App Engine Flexible Environment]

Yes (Tricky). This is not a screenshot of the user interface built in the tutorial. However, it does show data produced from the tutorial's code ("displayName", "logs", etc.)

At the bottom of the Firebase Realtime Database, under the `/inbox/` data location, there is a group of nodes prefixed by `client-` and followed by a randomly generated key that represents a user's account login. The last entry in this example, `client-1240563753`, is followed by a 16-digit identifier of the servlet currently listening to log events from that user, in this example `0035806813827987`.



From Build an Android App Using Firebase and the App Engine Flexible Environment

No (Tricky). These *are* screenshots of user interfaces, but it's not clear from the context of the tutorial whether they were generated from the code. It's more likely they're just screenshots of other UIs.

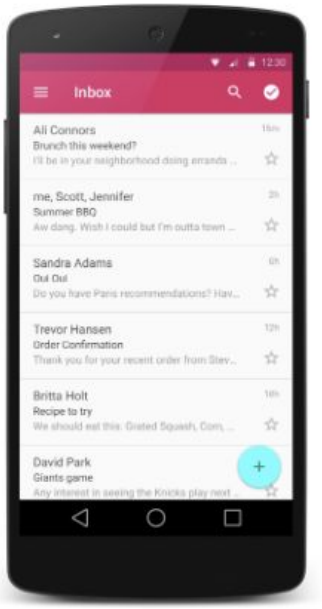


Figure 1. A list using `RecyclerView`

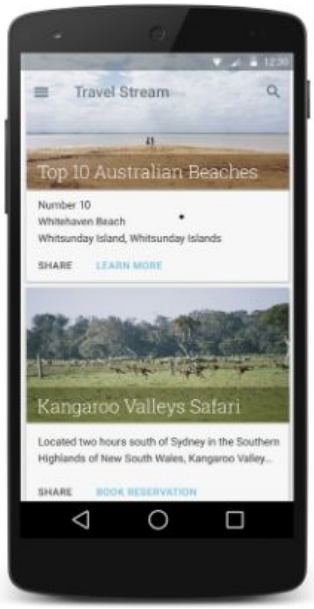


Figure 2. A list also using `CardView`

From “Create a List with RecyclerView” (no link available)

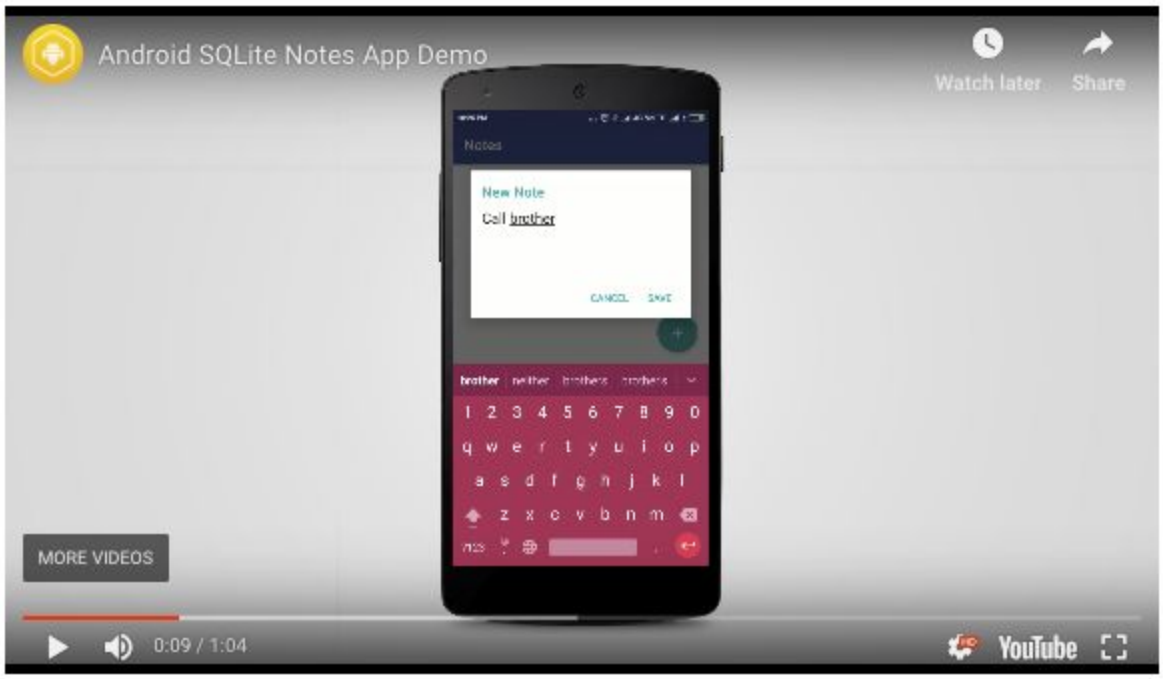
Videos of Output

An embedded video of program output (e.g., footage of using a user interface built in the tutorial, or footage of console logs generated from running the code). This does not include screencasts of someone writing the code, unless the screencasts show program outputs.

Examples

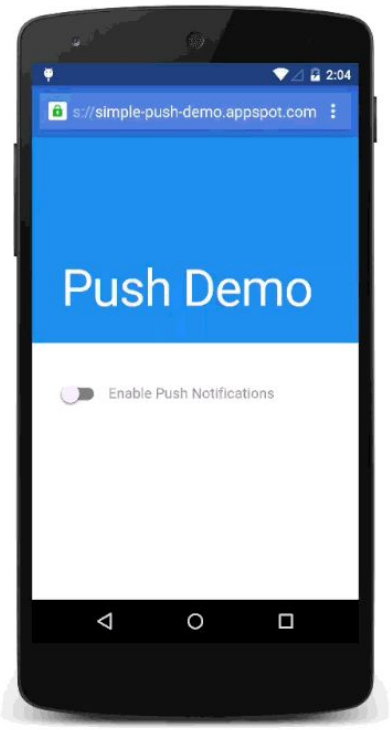
Yes. This YouTube video was probably produced as a recording of running the code.

VIDEO DEMO



From [Android SQLite Database Tutorial]

Yes. This GIF of an app developed in the tutorial (it's not animated here, but it was in the tutorial).



From [Push Notifications on the Open Web]

File Contents

A partial or complete listing of a text file generated by the program. Note that this can and often will count as both an output *and* as a code block. One example is inspecting the contents of a file that has been downloaded using a 'git clone' command.

Examples

Yes. A partial listing of the contents of a text file (below the text "Your file contains…"). The text file was modified by running code from an earlier block (the *git merge* command).

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Anything that has merge conflicts and hasn't been resolved is listed as unmerged. Git adds standard conflict-resolution markers to the files that have conflicts, so you can open them manually and resolve those conflicts. Your file contains a section that looks something like this:

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

From Git Branching - Basic Branching and Merging

Yes. A template for a file is generated by running a command line command.

2.1.1 Create the Mailer

```
$ bin/rails generate mailer UserMailer
create  app/mailers/user_mailer.rb
create  app/mailers/application_mailer.rb
invoke  erb
create    app/views/user_mailer
create    app/views/layouts/mailer.text.erb
create    app/views/layouts/mailer.html.erb
invoke  test_unit
create    test/mailers/user_mailer_test.rb
create    test/mailers/previews/user_mailer_preview.rb
```

```
# app/mailers/application_mailer.rb
class ApplicationMailer < ActionMailer::Base
  default from: "from@example.com"
  layout 'mailer'
end

# app/mailers/user_mailer.rb
class UserMailer < ApplicationMailer
end
```

From Action Mailer Basics

No (Tricky). Although the contents are the file are shown, it's through output on the console. So this should be annotated as a console output, instead of file contents.

```
$ cat /tmp/id_rsa.john.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCB007n/ww+ouN4gSLKssMxXnBOvf9LGt4L
ojG6rs6hPB09j9R/T17/x4lhJA0F3FR1rP6kYBRsWj2aThGw6HXLm9/5zytK6Ztg3RPKK+4k
Yjh6541NYsnEAZuXz0jTTyAUfrtU3Z5E003C4oxOj6H0rfIF1kKI9MAQLMdpGW1GYEIgS9Ez
Sdfd8AcCIicTDWbqLAcU4UpkaX8KyGlLwsNuuGztobF8m72ALC/nLF6JLtPofwFBlgc+myiv
O7TCUSBdLQlgMVOFq1I2uPWQOkOWQAHukEOmfjy2jctxSDBQ220ymjaNsHT4kgtZg2AYYgPq
dAv8JggJICUvax2T9va5 gsg-keypair
```

From [Git on the Server - Setting Up the Server]

Live Demo

The tutorial shows code used in a live, executable demo. The demo is either shown within the page, or at a URL linked prominently (from a big, easily-visible button or a hyperlink on its own line).

Examples

Yes. If the user has Java Web Start installed, this runs a completed version of the code shown in the tutorial. This button is prominent and easy-to-see.

From [How to Use Tables]

Yes. The user can click a button to run the code in the page.

JavaScript Code:

```
<script type="text/javascript">
function changeText(){
      document.getElementById('boldStuff').innerHTML = 'Fred Flinstone';
}
</script>
<p>Welcome to the site <b id='boldStuff'>dude</b> </p>
<input type='button' onclick='changeText()' value='Change Text'/>
```

Display:

From [javascript innerhtml]

Yes (Tricky). If you inspect the tutorial's HTML, it looks like only half of the code is demo'd—the part that processes the HTML form. The other half of the code that generates the form HTML appears to have been run beforehand, or simulated by writing up the HTML by hand. Though as long as *part* of the code is being demonstrated live, we say this is a live demo.

```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
first_name = form.getvalue('first_name')
last_name  = form.getvalue('last_name')

print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Hello - Second CGI Program</title>"
print "</head>"
print "<body>"
print "<h2>Hello %s %s</h2>" % (first_name, last_name)
print "</body>"
print "</html>"
```

Let us take again same example as above which passes two values using HTML FORM and submit button. We use same CGI script hello_get.py to handle this input.

```
<form action = "/cgi-bin/hello_get.py" method = "post">
First Name: <input type = "text" name = "first_name"><br />
Last Name: <input type = "text" name = "last_name" />

<input type = "submit" value = "Submit" />
</form>
```

Here is the actual output of the above form. You enter First and Last Name and then click submit button to see the result.

From [Python - CGI Programming]

No. The code below is so bare and leaves so much for the user to fill in that it should not be considered to be code used in any of the three live demos for PFold.

For using the plugin, the following structure is expected:

```
<div id="uc-container" class="uc-container">
  <div class="uc-initial-content">
    <!-- initial content -->
  </div>
  <div class="uc-final-content">
    <!-- final content -->
  </div>
</div>
```

The final content's size will depend on the initial content's size (set in the CSS), the folding directions and the number of folding steps. For example, having an initial element of 200px width and height, a folding direction of bottom and left and two folding steps will create a final area of 400px in width and height.

The plugin can be called like this:

```
$( '#uc-container' ).pfold();
```

From PFold: Paper-Like Unfolding Effect

Editable Demo

Only applicable if Live Demo is "Yes."

Code for the live demo can be edited, and the code can be re-executed to see the effects of changing the code.

Examples

Yes. Clicking on the cog in the upper right corner of the code example leads the reader to an interactive C++ program editor, where they can run the code, edit it, and re-run it to see the new output.

Here is the complete example of class Rectangle:

```
// classes example
#include <iostream>
using namespace std;

class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    int area() {return width*height;}
};

void Rectangle::set_values (int x, int y) {
  width = x;
  height = y;
}

int main () {
  Rectangle rect;
  rect.set_values (3,4);
  cout << "area: " << rect.area();
  return 0;
}
```

```
area: 12
```

C++ shell

```
// classes example
#include <iostream>
using namespace std;

class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    int area() {return width*height;}
};

void Rectangle::set_values (int x, int y) {
  width = x;
  height = y;
}

int main () {
  Rectangle rect;
  rect.set_values (3,5);
  cout << "area: " << rect.area();
  return 1;
}
```

Get URL | Run

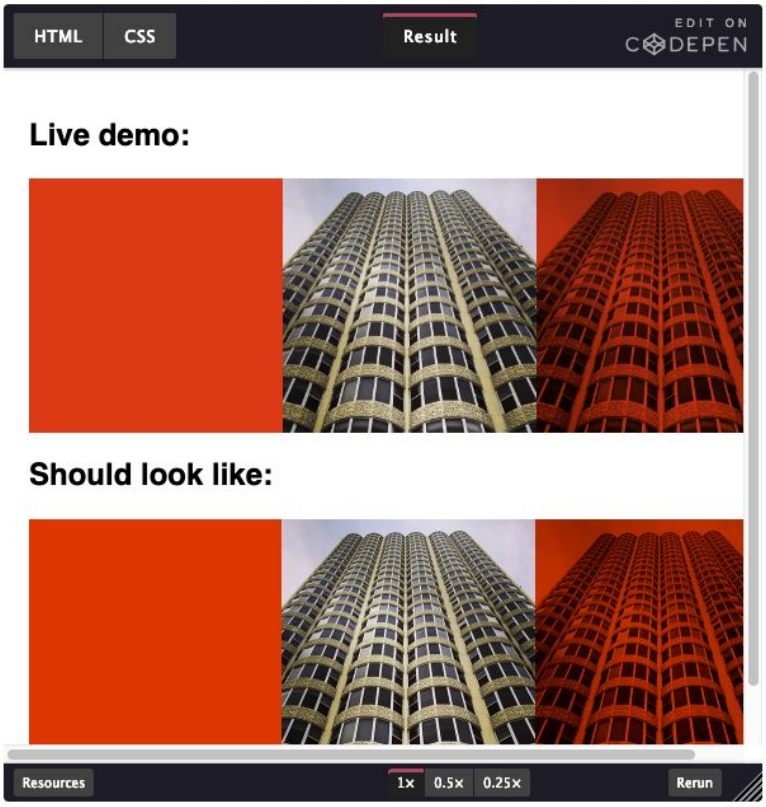
options | compilation | execution

```
area: 15
```

```
Exit code: 1
```

From Classes (I)

Yes. If you click on “Edit on CodePen,” you can edit the code for this demo.



From “Basics of CSS Blend Modes” (link not available)

IX. Visuals

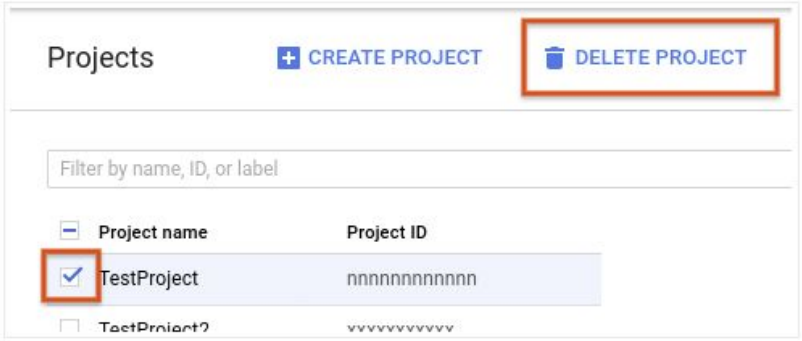
User Interface Instruction Images

Screenshots of other user interfaces (besides those being built in the tutorial) that show the user how to manage their development environment, run their code, or check on results. Must be related to the code blocks in the tutorial—images that don’t help set up the environment for writing the code blocks, running them, or evaluating their results, shouldn’t be counted.

Examples

Yes. This image describes actions the user should take in a user interface, in this case to clean up and delete their project at the end of the tutorial. Despite the fact that there are annotations on this image, it should be tagged as a user interface instruction image instead of a diagram.

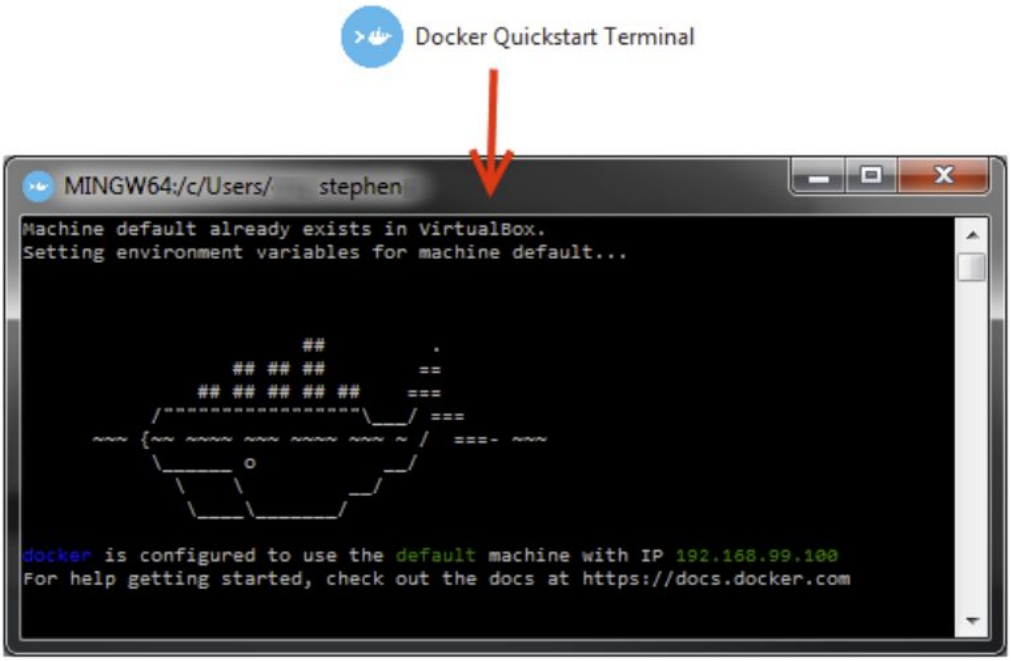
2. In the project list, select the project you want to delete and click **Delete project**.



From [Build an Android App Using Firebase and the App Engine Flexible Environment]

Yes. While this image doesn't show someone a user interface action to take, it does show the state of a user interface after the user is asked to take action (i.e. open the Quickstart Terminal).

Then I opened up the Docker Quickstart Terminal.



From [How to install and run TensorFlow on a Windows PC]

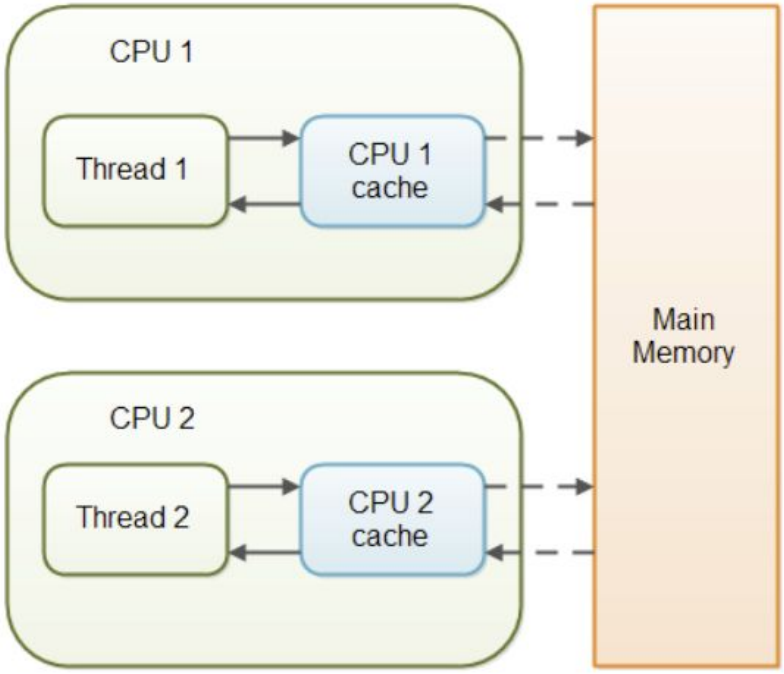
Diagrams

Diagrams are images or text art that helps a reader understand the goals of the tutorial, the technologies used in the tutorial, or the code they will write while following the tutorial. This includes, for example,

sketches of user interfaces, and of programming components and how they relate. It does not include screenshots, screenshots with annotations on them, tables, formulas, or plaintext equations.

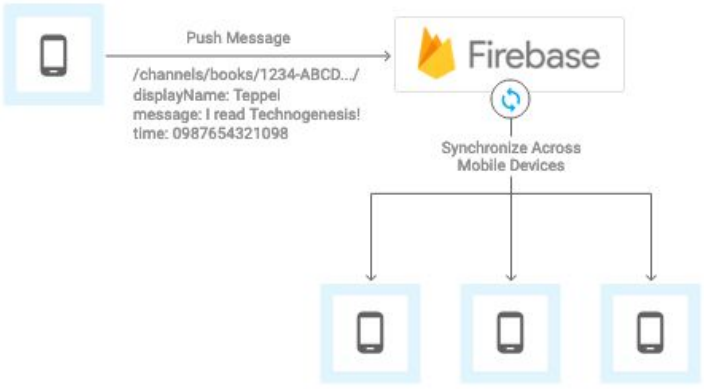
Examples

Yes. this diagram of memory management in Java.



From Java Volatile Keyword

Yes. This diagram of information flow between mobile apps and Firebase.



From Build an Android App Using Firebase and the App Engine Flexible Environment

Yes (Tricky). File listings usually aren't diagrams, but this one is, as it's annotated and used to describe some generic idea, in this case the typical file structure of an Android Studio project.

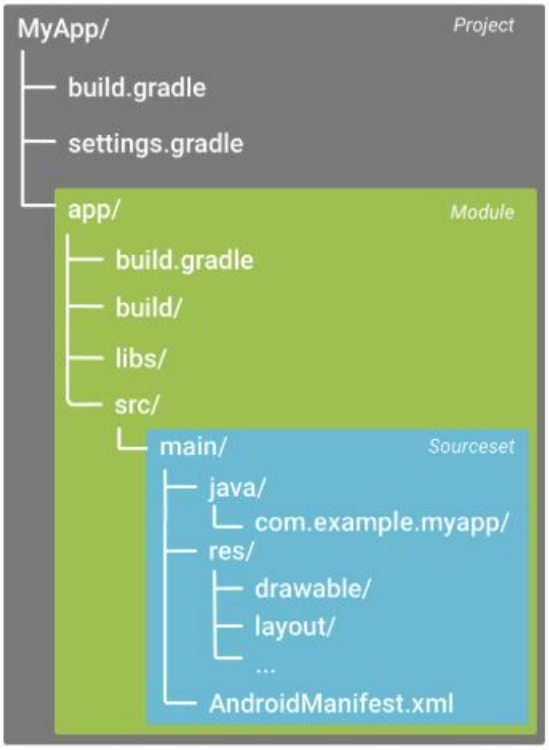
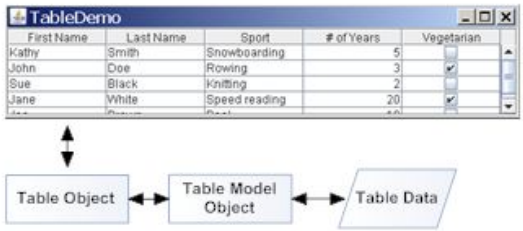


Figure 1. The default project structure for an Android app module.

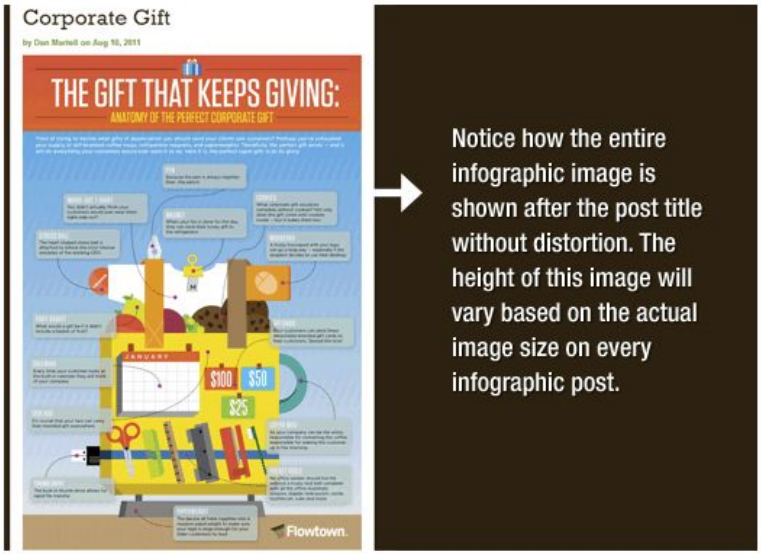
Yes (Tricky). Part of this figure contains a screenshot that could have been generated by running code from the tutorial. However, the generated output is not at all important in this figure. The screenshot is just there as a stand-in for any table.

Every table object uses a *table model object* to manage the actual table data. A table model object must implement the `TableModel` interface. If the programmer does not provide a table model object, `JTable` automatically creates an instance of `DefaultTableModel`. This relationship is illustrated below.



From How to Use Tables

No. This figure is just a screenshot of a website with the caption on the side.



From [How to Create Additional Image Sizes in WordPress]

Other Images

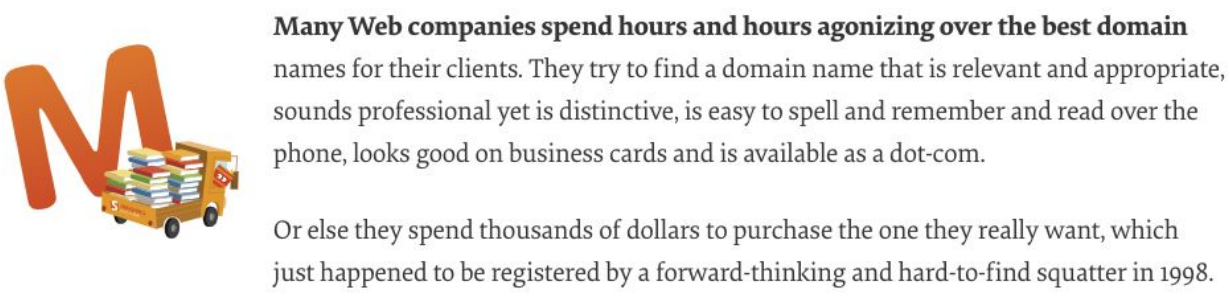
Any other images that don't fall into the categories above in "Generated Outputs" and "Visuals." Examples include memes and project logos. These must appear in the body of the tutorial—for example, images in the page nav or ads off to the side of the tutorial do not count.

Don't include ads beneath the last paragraph of the tutorial text, formulas count formulas as images, and icons (e.g., icons denoting important information, or difficulty of sections).

Do count screenshots of file browsers, and splash images (those that appear right above, under, or alongside the title, and are unique to this tutorial).

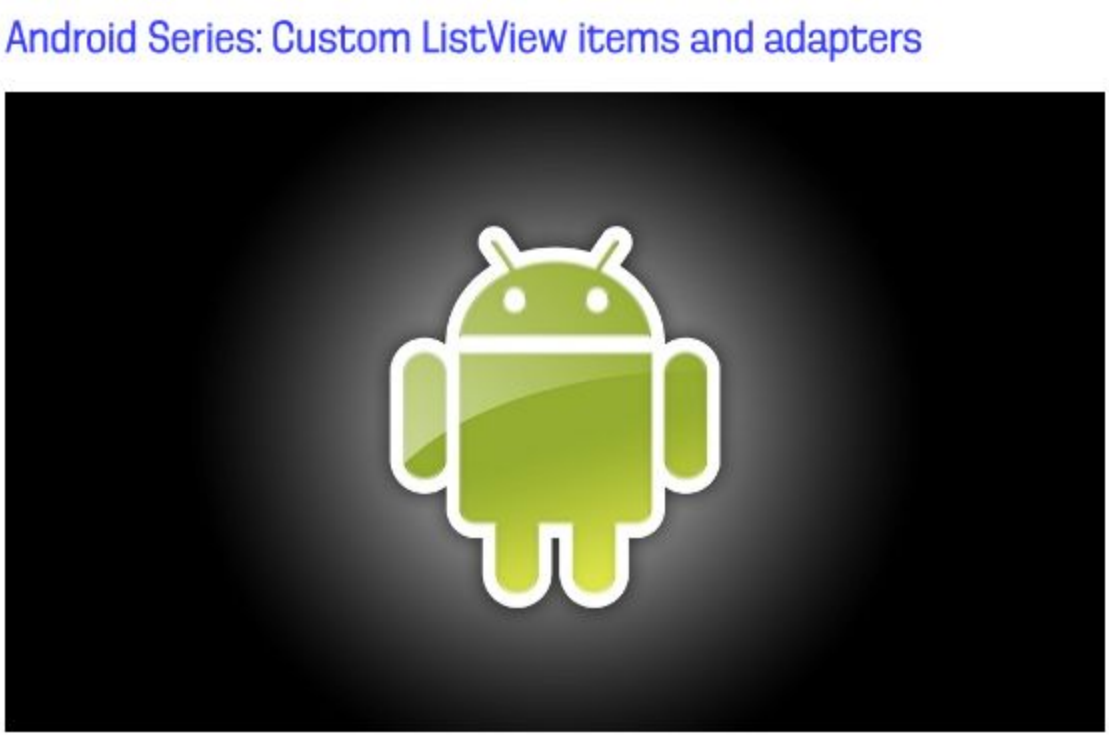
Examples

Yes. This image was not generated from code, and isn't used to describe any concepts.



From [Introduction to URL Rewriting]

Yes. This image has no conceptual purpose, and is likely just decoration.



From [Custom ListView items and adapters]

Yes (Tricky). These screenshots show the types of apps one could create using code from the tutorial. However, it's unlikely these apps were actually written using code from the tutorial.

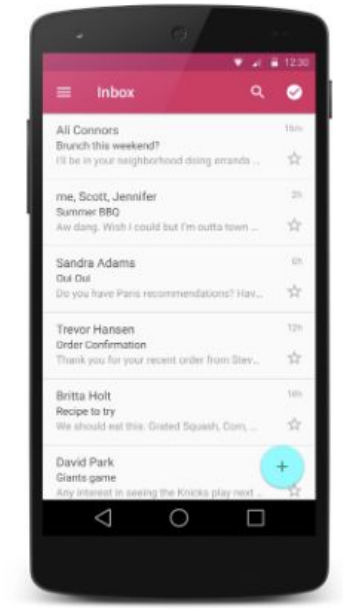


Figure 1. A list using `RecyclerView`

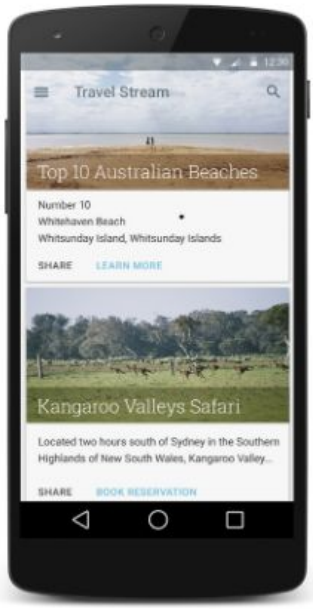
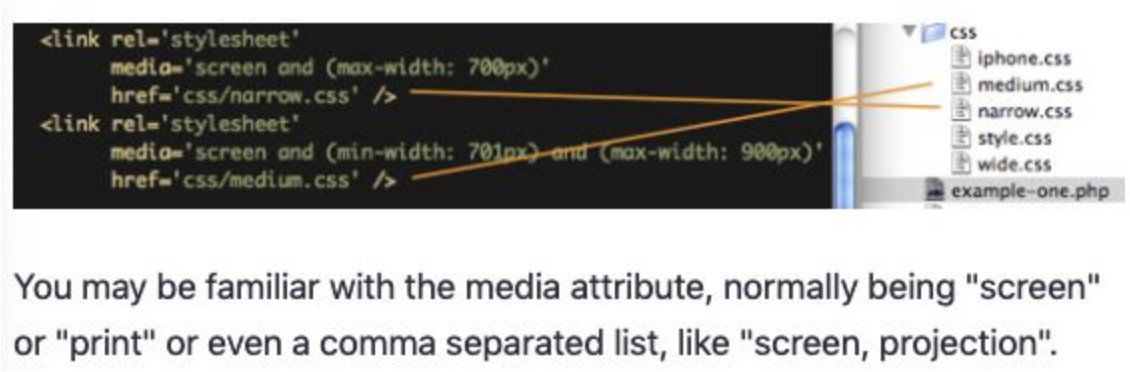


Figure 2. A list also using `CardView`

From “Create a List with RecyclerView” (no link available)

Yes (Tricky). It’s not clear from the tutorial what this image conveys. It doesn’t seem to indicate anything about the code or technologies described in the tutorial, so we do not classify it as a diagram.



From [CSS Media Queries & Using Available Space]