2

Welcome App

Dive-Into® Xcode: Introducing Visual User Interface Design with Cocoa Touch, Interface Builder, Storyboarding and Auto Layout, Universal Apps, Accessibility, Internationalization

Objectives

In this chapter you'll:

- Learn the basics of the Xcode integrated development environment (IDE), which you'll use to write, test and debug your iOS apps.

- Use the **Single View Application** project template to quickly begin developing a new app.

- Create a universal app that can run on iPhones, iPod touches and iPads.

- Design an app's UI visually (without programming) using Interface Builder, storyboarding and auto layout.

- Display text and an image in a UI.

- Support both portrait and landscape orientations.

- Edit the attributes of Cocoa Touch UI components.

- Build and launch an app in the iOS simulator.

- Make the app more accessible to visually impaired people by specifying string descriptions for use with iOS's VoiceOver.

- Support internationalization so your app can display strings in different languages based on the user's device settings.

Self-Review Exercises

2.1 Introduction

2.1   (True/False) You can create apps that display text and images without writing any code.
    ANS: *True*.

2.2   Xcode's _____ allows you to create UIs using drag-and-drop techniques and no Swift programming.
    ANS: Interface Builder

2.3   (True/False) You must be a paid iOS Developer Program member to be able to run apps (that you're developing) on an iOS device.
    ANS: *True*.

2.2.1 Xcode and Interface Builder

2.4   You can use Interface Builder's _____ capability to specify how apps will transition between screens.
    ANS: storyboarding

2.2.3 Asset Catalogs and Image Sets

2.5   (True/False) You can create your own image sets to manage your app's image resources. If you do not provide icons for each size and resolution, iOS will issue an error.
    ANS: *False*. **Actually, if you do not provide icons for each size and resolution, iOS will**
            scale **the images that you do provide, using the image that's closest in size to what**
            it needs.

2.3.2 Projects and App Templates

2.6   Which Xcode iOS app template creates an app with a UI that displays a *master list* of items from which a user can choose one item to see its *details* (similar to the built-in Mail and Contacts apps)?
    ANS: Master-Detail Application

2.7   Which Xcode iOS app template creates an app in which everything is *displayed on one screen*?
    ANS: Single View Application

2.8   Which Xcode iOS app template creates an app with features that support game development with one of iOS's gaming APIs—SceneKit, SpriteKit, OpenGL ES or Metal?
    ANS: Game

2.3.3 Creating and Configuring a Project

2.9   When you create a new project, you can choose to use _____—a source-code control system for managing projects to which multiple developers contribute, but you can also use it yourself to manage and track the revisions you make to your app.
    ANS: Git

2.4.3 Utilities Area and Inspectors

2.10   The set of inspectors you can choose from depends on what you're doing in Xcode. By default, the top half of the Utilities area shows either the File inspector or the Quick Help inspector. The File inspector shows information about the currently selected file in the project. The Quick Help inspector provides _____ help—documentation that's based on the currently selected item in a

UI or the cursor position in the source code. For example, clicking on a method name shows a description of the method, its parameters and its return value.

> ANS: **context-sensitive**

2.4.5 Xcode Toolbar

2.11 (True/False) The Xcode toolbar contains options for executing your app, a display area that shows the progress of tasks executing in Xcode (e.g., project build status) and buttons for hiding and showing areas in the workspace window.

> ANS: *True.*

2.5 Storyboarding the Welcome App's UI

2.12 When you create a new app, Xcode creates a(n) _____ file that you use to design UIs that are appropriate for the user-interface idiom of each type of device.

> ANS: **.storyboard**

2.5.1 Configuring the App for Portrait and Landscape Orientations

2.13 (True/False) As you know, users can hold their devices in portrait (long edge vertical) or landscape (long edge horizontal) orientation. Many apps support both orientations by rearranging their UIs, depending on the current device orientation. Supporting both orientations is the default.

> ANS: *True.*

2.14 Why does Apple recommend that you do not support the **Upside Down** orientation in iPhone apps?

> ANS: **If the phone is upside down when the user receives a call, it's more difficult to answer the phone.**

2.5.2 Providing an App Icon

2.15 (True/False) By default, the **AppIcon** image set contains empty placeholders for various iPhone and iPad app icons. Each is labeled **1x**, **2x** or **3x**. These represent non-retina-display (**1x**) and retina-display (**2x** or **3x**) devices with difference pixel densities. The measurements are in points. For **1x** icons the relationship is one pixel = one point, for **2x** it's one pixel = two points and for **3x** (the iPhone 6 Plus) it's one pixel = three points.

> ANS: *False.* **Actually, for 1x icons the relationship is one point = one pixel, for 2x it's one point = two pixels and for 3x (the iPhone 6 Plus) it's one point = three pixels.**

2.5.4 Overview of the Storyboard and the Xcode Utilities Area

2.16 You design an app's UI in its storyboard. In a storyboard, each screen of information is represented as a(n) _____—designated by a white rectangular area.

> ANS: **scene**

2.17 (True/False) Once the storyboard is displayed, the bottom part of the **Utilities** area shows the **Library** window, which contains four library tabs: **File Template** (common file types for quickly adding files to a project), **Code Snippet** (code snippets for quickly inserting and customizing commonly used code, such as control statements, exception handling and more), **Object** (standard Cocoa Touch UI components for designing iOS apps) and **Media** (the project's images, audios and videos).

> ANS: *True.*

2.5.6 Using Inspectors to Configure the Image View

2.18 When you're designing a UI, the top of the **Utilities** area will have additional tabs for the following inspectors: the _____ inspector (used to specify an object's class and accessibility information and to provide a name for the object that appears in the list of objects to the left of the scene design area), the **Attributes** inspector (used to customize the selected object's attributes, such as the image to display in an **Image View**), the **Size** inspector (used to configure an object's size and position) and the **Connections** inspector (use to create connections between code and UI components, e.g., to respond to user interactions with particular components).
    ANS: Identity

2.5.8 Using Auto Layout to Support Different Screen Sizes and Orientations

2.19 (True/False) The document outline window shows you all of the UI components that make up your scene(s).
    ANS: *True.*

2.7.1 Enabling Accessibility for the Image View

2.20 (True/False) Some apps dynamically generate UI components in response to user interactions. For such UI components, you can programmatically set the accessibility text using properties from the UIAccessibility protocol.
    ANS: *True.*

2.8 Internationalizing Your App

2.21 By default, each app you create uses base internationalization—the string resources in your app are separated from your storyboard and used as a template for providing localized strings for other languages. The language you use during development is known as your app's base language. If you don't provide strings in the appropriate language for a given locale, iOS uses _____ by default.
    ANS: the base language strings

2.8.1 Locking Your UI During Translation

2.22 Each UI component has a(n) _____ that's used as part of the internationalization and localization process.
    ANS: unique ID

Short-Answer Questions

2.1 Introduction

2.23 _____ is Apple's suite of development tools for creating and testing Mac OS X and iOS applications.
    ANS: Xcode

2.24 You can make an app more accessible for people with impaired vision by providing accessibility strings that describe screen images to the user—iOS's _____ accessibility feature can speak the accessibility strings to the user.
    ANS: VoiceOver

2.2.1 Xcode and Interface Builder

2.25   Xcode's Interface Builder enables you to visually (i.e., without programming) lay out your UI. You can use it to _____ **Label**s, **Image View**s, **Button**s, **Text Field**s, **Slider**s and other UI components onto an app's UI.

> **ANS: drag and drop**

2.2.5 Accessibility

2.26   People with visual disabilities can use iOS's VoiceOver to allow a device to speak screen text (such as the text on a **Label** or **Button**) or text that you provide to help them understand the purpose and contents of a UI component. The user can touch the screen to hear VoiceOver speak _____.

> **ANS: what's on the screen near the touch**

2.2.6 Internationalization

2.27   iOS devices are used worldwide. To reach the largest possible audience with your apps, you should consider designing your app so that it can be customized for various *locales* and spoken languages—this is known as _____.

> **ANS: internationalization**

2.3.2 Projects and App Templates

2.28   Which Xcode iOS app template creates an app in which content is *displayed page by page* (similar to the built-in **iBooks** app)?

> **ANS: Page-Based Application**

2.29   Which Xcode iOS app template creates an app with a tab bar (similar to the built-in **Clock** app)? The user touches a tab to change screens.

> **ANS: Tabbed Application**

2.3.3 Creating and Configuring a Project

2.30   (True/False) Every app you create must be designed as a Universal app that runs on iPhones, iPod touches and iPads.

> **ANS:** *False.* **You also can create apps that are specifically for iPhones/iPod touches or iPads.**

2.4.1 Navigator Area

2.31   At the top of the **Navigator** area are icons for the navigators that can be displayed there, including: **Project** (shows the files and folders in your project), **Symbol** (allows you to browse your project by classes and their contents (methods, properties, etc.), **Find** (allows you to search for text throughout your project's files and frameworks) and **Issue** (_____).

> **ANS: shows you warnings and errors in your project by file or by type**

2.4.2 Editor Area

2.32   To the right of the **Navigator** area is the **Editor** area for editing source code and designing UIs. This area is *always* displayed in your workspace window. When you select a file in the project navigator, its contents are displayed in the **Editor** area. There are three editors: the _____ editor (shows the selected file's contents), the **Assistant** editor (shows the selected file's contents on the left and related file contents on the right—for example, if you're editing a class that extends another

class, the **Assistant** editor will also show the superclass) and the **Version** editor (allows you to compare different versions of the same file, e.g., old and new versions).

> **ANS:** Standard

2.4.3 Utilities *Area and Inspectors*

2.33 (True/False) At the right side of the workspace window is the **Utilities** area, which displays inspectors that allow you to view and edit information about items displayed in the **Editor** area.

> **ANS:** *True.*

2.4.4 Debug *Area*

2.34 When displayed, the _____ area appears at the bottom of the editor area and provides controls for stepping through code, inspecting variable contents and more.

> **ANS:** Debug

2.4.6 *Project Navigator*

2.35 The **Project** navigator provides access to all of a project's components. It consists of a series of groups (folders) and files. The most used group is the _____, which Xcode names the same as the project. This group contains your project's source files, media files and supporting files.

> **ANS:** project structure group

2.5.1 Configuring the App for Portrait and Landscape Orientations

2.36 With the exception of the _____ orientation for iPhones, Apple recommends supporting all possible device orientations.

> **ANS:** Upside Down

2.5.2 Providing an App Icon

2.37 To improve the user's experience in an app that take several seconds to load, you can also specify a(n) _____ screen that your app displays while it's loading, so the user does not see a blank screen. In prior iOS versions, this screen was an image. As of iOS 8, it can be a resizable UI that adjusts to fit the device on which the app is running.

> **ANS:** launch

2.5.3 Creating an Image Set for the App's Image

2.38 (True/False) As with app icons, you'll typically provide multiple versions of each image your app displays to accommodate various device sizes and pixel densities. Placing such images into the asset catalog as image sets allows iOS to choose the correct image for you based on the device resolution. Because images scale well in iOS 8 in particular, it's acceptable to provide one image and iOS 8 will scale it as necessary for the various device sizes and pixel densities.

> **ANS:** *False.* **Actually, images do not scale well so it's better to provide customized images.**

2.5.4 Overview of the Storyboard and the Xcode *Utilities Area*

2.39 (True/False) You drag and drop UI components from the **Code Snippet** library tab to add them to your scene.

> **ANS:** *False.* **Actually, you do this with the Object library.**

2.5.5 Adding an Image View to the UI

2.40   Dashed blue guide lines appear as you drag a UI component around a scene in the storyboard. These suggest component spacing and alignments that help you conform to Apple's _____, which include conventions for spacing between components, component positioning and alignment, gestures used to interact with apps and much more.

> **ANS: Human Interface Guidelines**

2.7.2 Confirming Accessibility Text with the Simulator's Accessibility Inspector

2.41   If you're not a paid member of the iOS Developer Program, you can use the simulator's _____ to ensure that your accessibility text is set correctly.

> **ANS: Accessibility Inspector**

2.8 Internationalizing Your App

2.42   To reach the largest possible audience with your apps, consider customizing them for various locales and spoken languages. Preparing your app to do this is known as internationalization, and creating the resources for each locale (such as text in different languages) is know as _____.

> **ANS: localization**

2.43   (True/False) Xcode supports XLIFF (XML Localization Interchange File Format) files for managing localized string resources. For translation purposes, Xcode can export an XLIFF file containing all of your app's localizable text.
True.

2.8.1 Locking Your UI During Translation

2.44   (True/False) Localization is best performed before you begin specifying your app's UI.
> **ANS: *False*. Actually, localization is best performed once you've completed your app's UI or when it's nearly complete.**

2.8.2 Exporting Your UI's String Resources

2.45   Xcode extracts the localizable strings from all the files in your project (not just the ones in the storyboard) and places them in the XLIFF file.
> **ANS: *True*.**

Programming Projects

2.46   *(Scrapbooking App)* Find four open source images of famous landmarks using websites such as Flickr. Create an app in which you arrange the images in a collage. Add text that identifies each landmark. Recall that image file names must use all lowercase letters.
> **ANS: This is nearly identical to the Welcome app, but consists of four Image Views and four Labels. Use Interface Builder, to place the controls. Select all of the user interface components then use the Resolve Auto Layout Issues tool to select Add Missing Constraints to add basic constraints to all the controls.**

2.47   *(Scrapbooking App with Accessibility)* Using the techniques you learned in Section 2.7, enhance your solution to Exercise 2.46 to provide strings that can be used with iOS's VoiceOver accessibility feature. If you have an iOS device available to you, test the app on the device with VoiceOver enabled.
> **ANS: This requires the same steps we demonstrated in Section 2.7 for the Welcome app. For this exercise, apply the steps to all four Image Views.**

2.48   *(Scrapbooking App with Internationalization)* Using the techniques you learned in Section 2.8, enhance your solution to Exercise 2.47 to define a set of strings for another spoken language. Use an online translator service, such as `translate.google.com` to translate the strings, then use the instructions in Section 2.8 to internationalize the app and import the localized strings. Test the app on the iOS simulator (or a device if you have one available to you).

> **ANS:** **This requires the same steps we demonstrated in Section 2.8 for the Welcome app. [*Note:* When localizing a for-sale app, Strings should be translated by someone with locale-specific expertise to ensure that the text makes sense in each spoken language and dialect.]**

2

Introduction to Swift Programming
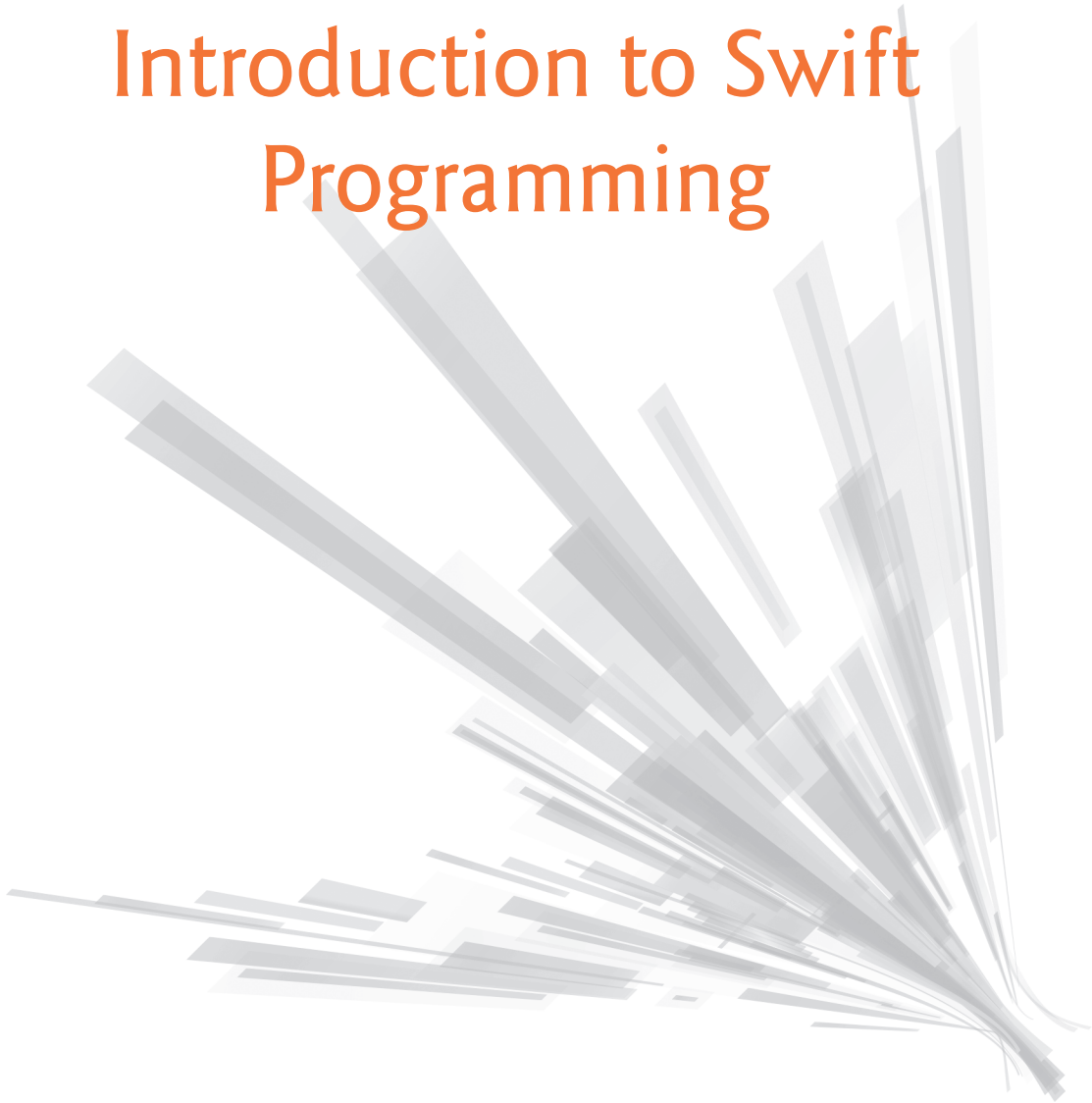
Self-Review Exercises

Section 2.2 A First Swift Program: Printing a Line of Text

2.1   (True/False) Like many C-based programming languages, Swift has a `main` function.
>   **ANS:** *False*. **Actually, unlike many C-based programming languages, Swift does *not* have a `main` function or method.**

2.2   (True/False) Swift Code files end with the `.src` filename extension.
>   **ANS:** *False*. **Actually, Swift Code files end with the `.swift` filename extension.**

2.3   (True/False) The compiler ignores comments.
>   **ANS:** *True*.

2.4   (True/False) A single-line comment can begin in the middle of a line, possibly after some code, and continue until before the end of that line, followed by more code.
>   **ANS:** *False*. **Actually, a single-line comment also can begin in the middle of a line, possibly after some code, and continue until *the end of that line*.**

2.5   Unlike as in most C-based languages, Swift's multiline comments may be _____, allowing you to easily comment-out large blocks of code (e.g., for debugging purposes) that contain multiline comments.
>   **ANS:** **nested**

2.6   String literals have the Swift type `string`.
>   **ANS:** *False*. **Actually, string literals have the Swift type `String` (with a capital S).**

2.7   Function `println` displays (or prints) a line of text to the standard output. This is one of the Swift Standard Library's _____ (global) functions.
>   **ANS:** **free**

2.8   (True/False) As with other C-based programming languages, Swift statements are required to end with a semicolon (;).
>   **ANS:** *False*. **Actually, *unlike* other C-based programming languages, Swift statements are *not* required to end with a semicolon (;), though you can use them if you like.**

2.9   (True/False) The toolbar cannot be displayed in a playground window.
>   **ANS:** *False*. **By default the toolbar is not displayed in a playground window. To display it, select View > Show Toolbar.**

Section 2.3 Modifying Your First Program

2.10   Each `print` or `println` statement resumes displaying characters from where the _____.
>   **ANS:** **last `print` or `println` statement stopped displaying characters**

2.11   A single statement can display multiple lines by using _____ characters, which indicate to the `print` and `println` functions when to position the output cursor at the beginning of the next line in the standard output.
>   **ANS:** **line-feed (\n)**

*Section 2.4 Composing Larger **Strings** with **String** Interpolation*

2.12   Describe what the following code does.

```
let color = "red"
println("My favorite color is \(color)!")
```

ANS: The first statement declares the constant `color` with an initial value of `"red"`. The second statement uses `String` interpolation to insert the value of `color` (`"red"`) into the `String` argument of `println` and the call to `println` outputs

```
My favorite color is red!
```

2.13   You use the var keyword to _____.
ANS: declare variables that can be modified

2.14   (True/False) It's considered good practice to assign a value to a variable before using that variable in your code.
ANS: *False.* **It's not an option—every variable *must* be assigned a value before it can be used in your code.**

2.15   (True/False) Values of the type `Boolean` are limited to `true` or `false`.
ANS: *False.* **The correct type name is `Bool`.**

2.16   (True/False) Identifiers for constants and variables typically begin with a lowercase first letter and may contain letters, numbers and most other Unicode characters—this includes even special characters like emojis, which you can access in the `Characters` dialog.
ANS: *True.*

2.17   Swift is _____—uppercase and lowercase letters are distinct—so `value` and `Value` are different (but both valid) identifiers.
ANS: case sensitive

Section 2.5 Another Application: Adding Integers

2.18   What is the type of `sum` in the code segment below?

```
let number1 = 45
let number2 = 72
let sum = number1 + number2
```

ANS: **The type of `sum` is inferred to be `Int` because `number1` and `number2` are inferred to be `Int` and the sum of two `Int`s is an `Int`.**

2.19   Floating-point literals are treated as type _____ (an eight-byte floating point number).
ANS: `Double`

2.20   The _____ integer types `Int8`, `Int16`, `Int32` and `Int64` are for positive and negative integer values.
ANS: signed

Section 2.6 Arithmetic

2.21   (True/False) One application of the remainder operator is determining whether the right operand is a multiple of the left operand.
ANS: *False.* **Actually, one application of the remainder operator is determining whether the *left* operand is a multiple of the *right* operand.**

Section 2.6.2 Operator Precedence

2.22   (True/False) All Swift operators associate either left to right or right to left.
ANS: *False.* **Some operators do not have associativity (such as the comparative operators).**

Section 2.7 Decision Making: The if Conditional Statement and the Comparative Operators

2.23   (True/False) Conditions in if statements can be formed by using the comparative operators (==, !=, >, <, >= and <=) . These operators all have the same level of precedence and left-to-right associativity.

>   ANS: *False*. **Actually, the comparative operators do not have associativity.**

2.24   (True/False) Unlike some other languages, Swift requires braces for an if statement's body, even if the body contains only one statement. Unfortunately, this "braces clutter" leads to common errors.

>   ANS: *False*. **Actually, this is one of several Swift requirements that *eliminate* common errors that occur in C-based languages.**

2.25   (True/False) In most C-based languages, a semicolon (;) by itself represents the empty statement. In those languages, an empty statement can be used as the one-statement body of a control statement. In the context of an if statement, that could lead to logic errors. For example, in C, the following code always executes *bodyStatement*, because the if statement's body is the empty statement (;) that follows the if's condition:

```
if (condition) ;
    bodyStatement
```

Because if statements and other control statements in Swift require braces ({}) around their bodies, the preceding code in Swift is a compilation error.

>   ANS: *True*.

2.26   The assignment operator, =, associates from right to left. However, unlike other C-based languages, it does not return a value—so an expression like x = y = 0 is a(n) _____.

>   ANS: **compilation error**

Short-Answer Exercises

Section 2.2 A First Swift Program: Printing a Line of Text

2.1   In an Xcode project, the file that contains the entry point for a Swift app must be named _____.

>   ANS: `main.swift`

2.2   (True/False) When you enter code in an Xcode playground, it compiles and executes as you complete each statement.

>   ANS: *True*.

2.3   A comment that begins with // is a single-line comment—it terminates _____.

>   ANS: **at the end of the line on which the // appears**

2.4   The following Swift code has multiline comments:

```
/* This is a multiline comment. It  */
/* can be split over multiple lines */
```

Rewrite this code using multiline comments as they were intended to be used.

>   ANS: `/* This is a multiline comment. It`
>   `can be split over multiple lines */`

2.5   (True/False) The characters between the quotation marks in "String literal" are a String literal.

>   ANS: *False*. **Actually, together, the quotation marks *and* the characters between them are a String literal.**

2.6   (True/False) White-space characters in `String` literals are ignored by the compiler.
ANS: *False*. **White-space characters in `String` literals are *not* ignored by the compiler.**

2.7   Function `println`'s output is followed by a(n) _____, which indicates that the next character output will be displayed at the beginning of the next line in the standard output.
ANS: **line break**

2.8   If you place more than one statement on the same line, they must be separated by _____.
ANS: **semicolons**

2.9   (True/False) You must explicitly tell Xcode when to compile your Swift code.
ANS: *False*. **Actually, as you write code in a playground or in a `.swift` file that's part of an Xcode project, *the compiler continuously compiles your code*.**

Section 2.3 Modifying Your First Program

2.10   (True/False) Unlike `println`, after displaying its argument, `print` does not issue a line break—the next character the program displays will appear immediately after the last character that `print` displays.
ANS: *True*.

2.11   Blank lines, spaces, tabs and line-feed characters are _____.
ANS: **whitespace**

2.12   When a(n) _____ character appears in a `String` being output, it causes the screen's output cursor to move to the beginning of the next line in the standard output.
ANS: **line-feed (`\n`)**

Section 2.4 Composing Larger `String`s with `String` Interpolation

2.13   Swift's `String` _____ enables you to insert values into `String` literals.
ANS: **interpolation**

2.14   Explain why the following statement is *false*: "A constant cannot be modified."
ANS: **Actually, a constant cannot be modified *after it's initialized*. When you first declare a constant, you are not required to initialize it. However, as soon as you assign it a value, then it is constant from that point forward.**

2.15   A type _____ explicitly specifies a constant's or variable's type. For example

```
var number1: Int
```

declares that the variable `number1` has type `Int`, which represents a whole-number integer value.
ANS: **annotation**

2.16   For the integer types, each type's minimum and maximum values can be determined with its _____ and _____ properties.
ANS: **`min`, `max`**

2.17   To perform _____ insert a backslash (\) followed by a set of parentheses containing the constant, variable, expression or literal value that you'd like to insert at that position in the `String` literal.
ANS: **`String` interpolation**

2.18   (True/False) Identifiers may not begin with a digit. Whitespace, math symbols and arrows are not allowed in identifiers.
ANS: *True*.

2.19 By convention, variable-name identifiers begin with a lowercase letter, and every word in the name after the first word begins with a capital letter. This naming convention is known as _____.

> **ANS: camel case, because the uppercase letters stand out like a camel's humps**

Section 2.5 Another Application: Adding Integers

2.20 Types `Float` and `Double` are for holding _____ numbers.
ANS: real (or floating-point)

2.21 (True/False) Type `Float` represents an eight-byte floating-point number.
ANS: *False*. Actually, type `Float` represents a *four*-byte floating-point number.

Section 2.6 Arithmetic

2.22 (True/False) The `%` operator can be used only with integers.
ANS: *False*. Actually, the `%` operator is most commonly used with integers but it *can also be used with floating-point types*.

Section 2.6.1 Automatic Arithmetic Overflow Checking

2.23 (True/False) Overflow can lead to unexpected results, so by default a calculation that overflows results in a runtime error that terminates execution.
ANS: *True*.

Section 2.6.2 Operator Precedence

2.24 The operators in arithmetic expressions are applied in a precise sequence determined by the rules of _____, which are generally the same as those in algebra.
ANS: operator precedence

Section 2.7 Decision Making: The `if` Conditional Statement and the Comparative Operators

2.25 (True/False) If the condition in an `if` statement is *true*, the body of the `if` statement executes. If the condition is *false*, the body does not execute.
ANS: *True*.

2.26 (True/False) An `if` statement always begins with keyword `if`, followed by a condition in parentheses then a required set of braces (`{}`) containing the statements to execute if the condition is `true`.
ANS: *False*. Actually, unlike other C-based languages, Swift does not require that an `if` statement's condition be enclosed in parentheses.

2.27 (True/False) Unlike some other languages, it is not possible in Swift to accidentally use = where == should be used. This is one of several Swift language features that eliminate common errors that occur in C-based languages.
ANS: *True*.

2.28 For your convenience, Apple organizes the keywords into several categories—keywords used in declarations, keywords used in statements, keywords used in expressions and types and keywords reserved in particular contexts (known as _____ keywords).
ANS: context-sensitive

Exercises

Note: Code for the programming exercises is located in the `Chapter02` folder with the instructor's manual solutions.

2.1   Assuming that x = 2 and y = 3, what does each statement display?
a) `println("x = \(x)")`
ANS: x = 2
b) `println("Value of \(x) + \(x) is \(x + x)")`
ANS: Value of 2 + 2 is 4
c) `println("\(x + y) = \(y + x)")`
ANS: 5 = 5

2.2   What does the following code print?

```
println("*\n**\n***\n****\n*****\n")
```

```
ANS: *
     **
     ***
     ****
     *****
```

2.3   What does the following code print?

```
println("*")
println("***")
println("*****")
println("****")
println("**")
```

```
ANS: *
     ***
     *****
     ****
     **
```

2.4   What does the following code print?

```
print("*")
print("***")
print("*****")
print("****")
println("**")
```

ANS: ***************

2.5   What does the following code print?

```
print("*")
println("***")
println("*****")
print("****")
println("**")
```

```
ANS: ****
     *****
     ******
```

2.6   *(Arithmetic)* Write a program that for two integers prints their sum, product, difference and quotient (division).

2.7   *(Largest and Smallest Integers)* Write a program that for three integers determines and prints the largest and smallest integers in the group. Use only the programming techniques you learned in this chapter.

2.8   *(Diameter, Circumference and Area of a Circle)* Write a program that for a given radius of a circle displays the circle's diameter, circumference and area using the constant 3.14159 for π. Use the following formulas (*r* is the radius):

diameter = 2*r*
circumference = 2π*r*
area = π*r*²

2.9   *(Separating the Digits in an Integer)* Write a program that for a five-digit number separates the number into its individual digits and displays the individual digits. For example, the number 42339 should be displayed as:

4   2   3   3   9

Use both division and remainder operations to "pick off" each digit.

2.10   *(Body Mass Index Calculator)* We introduced the body mass index (BMI) calculator in Exercise 1.10. The formulas for calculating BMI are

$$BMI = \frac{weightInPounds \times 703}{heightInInches \times heightInInches}$$

or

$$BMI = \frac{weightInKilograms}{heightInMeters \times heightInMeters}$$

Create a BMI calculator that—based on a weight in pounds and height in inches (or, if you prefer, the user's weight in kilograms and height in meters)—calculates and displays the user's body mass index. Also, display the following information from the Department of Health and Human Services/National Institutes of Health so the user can evaluate his/her BMI:

```
BMI VALUES
Underweight: less than 18.5
Normal:      between 18.5 and 24.9
Overweight:  between 25 and 29.9
Obese:       30 or greater
```