

[< How to create strings](#)[How to store decimal numbers >](#)

How to store whole numbers

Paul Hudson  [@twostraws](#) May 12th 2022

Updated for Xcode 16.4

How to store whole numbers – Swift for Complete Beginners



When you're working with whole numbers such as 3, 5, 50, or 5 million, you're working with what Swift calls *integers*, or **Int** for short – “integer” is originally a Latin word meaning “whole”, if you were curious.

Making a new integer works just like making a string: use **let** or **var** depending on whether you want a constant or variable, provide a name, then give it a value. For example, we could create a **score** constant like this:

```
let score = 10
```

Integers can be really big – past billions, past trillions, past quadrillions, and well into *quintillions*, but they can be really small too – they can hold *negative* numbers up to quintillions.

When you're writing out numbers by hand, it can be hard to see quite what's going on. For example, what number is this?

```
let reallyBig = 100000000
```

If we were writing that out by hand we'd probably write “100,000,000” at which point it's clear that the number is 100 million. Swift has something similar: you can use underscores, _, to break up numbers however you want.

So, we could change our previous code to this:

```
let reallyBig = 100_000_000
```

Swift doesn't actually care about the underscores, so if you wanted you could write this instead:

```
let reallyBig = 1_00_00_00_00
```

The end result is the same: **reallyBig** gets set to an integer with the value of 100,000,000.

Of course, you can also create integers from other integers, using the kinds of arithmetic operators that you learned at school: **+** for addition, **-** for subtraction, ***** for multiplication, and **/** for division.

For example:

```
let lowerScore = score - 2
let higherScore = score + 10
let doubledScore = score * 2
let squaredScore = score * score
let halvedScore = score / 2
print(score)
```

Rather than making new constants each time, Swift has some special operations that adjust an integer somehow and assigns the result back to the original number.

For example, this creates a **counter** variable equal to 10, then adds 5 more to it:

```
var counter = 10
counter = counter + 5
```

Rather than writing **counter = counter + 5**, you can use the shorthand operator **+ =**, which adds a number directly to the integer in question:

```
counter += 5
print(counter)
```

That does exactly the same thing, just with less typing. We call these *compound assignment operators*, and they come in other forms:

```
counter *= 2
print(counter)
counter -= 10
print(counter)
counter /= 2
print(counter)
```

Before we're done with integers, I want to mention one last thing: like strings, integers have some useful functionality attached. For example, you can call **isMultiple(of:)** on an integer to find out whether it's a multiple of another integer.

So, we could ask whether 120 is a multiple of three like this:

```
let number = 120
print(number.isMultiple(of: 3))
```

I'm calling `isMultiple(of:)` on a constant there, but you can just use the number directly if you want:

```
print(120.isMultiple(of: 3))
```

BLACK FRIDAY

50% OFF BOOKS + VIDEOS

SAVE 50% All our books and bundles are half price for Black Friday, so you can take your Swift knowledge further for less! Get my all-new book **Everything but the Code** to make more money with apps, get the **Swift Power Pack** to build your iOS career faster, get the **Swift Platform Pack** to builds apps for macOS, watchOS, and beyond, or get the **Swift Plus Pack** to learn Swift Testing, design patterns, and more.

Save 50% on all our books and bundles!



Code got you started. This gets you paid.

You don't need more tutorials, you need a *plan*. That's where this book comes in: it has everything you need to go from Xcode to App Store, from finding killer ideas, to launch strategy, to breakout success.

Learn how to design, price, position, and promote your app so it doesn't just launch – it *lands*.

[Get it here](#)



[< How to create strings](#)

[How to store decimal numbers >](#)

Was this page useful? Let us know!



Average rating: 4.6/5

[Click here to visit the Hacking with Swift store >>](#)



Twitter



Mastodon



Email



Sponsor the site

About

Glossary

Code License

Privacy Policy

Refund Policy

Update Policy

Code of

Conduct

Swift, SwiftUI, the Swift logo, Swift Playgrounds, Xcode, Instruments, Cocoa Touch, Touch ID, AirDrop, iBeacon, iPhone, iPad, Safari, App Store, watchOS, tvOS, visionOS, Mac and macOS are trademarks of Apple Inc., registered in the U.S. and other countries. Pulp Fiction is copyright © 1994 Miramax Films.

Hacking with Swift is ©2025 Hudson Heavy Industries.



You are not logged in

[Log in or create account](#)