

[< How to use a for loop to repeat work](#)[How to skip loop items with break and continue >](#)

# How to use a while loop to repeat work

[Paul Hudson](#) [@twostraws](#) October 25th 2021*Updated for Xcode 16.4*

How to use a while loop to repeat work – Swift for Complete Beginners



Swift has a second kind of loop called **while**: provide it with a condition, and a **while** loop will continually execute the loop body until the condition is false.

Although you'll still see **while** loops from time to time, they aren't as common as **for** loops. As a result, I want to cover them so you know they exist, but let's not dwell on them too long, okay?

Here's a basic **while** loop to get us started:

```
var countdown = 10

while countdown > 0 {
    print("\(countdown)...\")
    countdown -= 1
}

print("Blast off!")
```

That creates an integer counter starting at 10, then starts a **while** loop with the condition **countdown > 0**. So, the loop body – printing the number and subtracting 1 – will run continually until **countdown** is equal to or below 0, at which point the loop will finish and the final message will be printed.

**while** loops are really useful when you just don't know how many times the loop will go around. To demonstrate this, I want to introduce you to a really useful piece of functionality that **Int** and **Double** both have: **random(in:)**. Give that a range of numbers to work with, and it will send back a random **Int** or **Double** somewhere inside that range.

For example, this creates a new integer between 1 and 1000

```
let id = Int.random(in: 1...1000)
```

And this creates a random decimal between 0 and 1:

```
let amount = Double.random(in: 0...1)
```

We can use this functionality with a **while** loop to roll some virtual 20-sided dice again and again, ending the loop only when a 20 is rolled – a critical hit for all you Dungeons & Dragons players out there.

Here's the code to make that happen:

```
// create an integer to store our roll
var roll = 0

// carry on looping until we reach 20
while roll != 20 {
    // roll a new dice and print what it was
    roll = Int.random(in: 1...20)
    print("I rolled a \(roll)")
}

// if we're here it means the loop ended – we got a 20!
print("Critical hit!")
```

You'll find yourself using both **for** and **while** loops in your own code: **for** loops are more common when you have a finite amount of data to go through, such as a range or an array, but **while** loops are really helpful when you need a custom condition.

# 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.



## 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 use a for loop to repeat work](#)

[How to skip loop items with break and continue >](#)

Was this page useful? Let us know!



Average rating: 4.7/5

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



Twitter



Mastodon



Email



Sponsor the site



You are not logged in

Log in or create account