In this lesson, you're going to learn the basics of setting up a simple system that you operate yourself.

Now, as we go through this lesson, remember all the things you've learned up to this point. All the productivity rules you've been practicing will help you make your new systems and routines as effective as possible. Those rules will help you design systems that flow smoothly and are driven by consistent action.

Over the next several lessons, you'll learn about the two major categories of systems we'll be focusing on for the rest of the course—simple systems and complex systems.

I think those names are pretty self-explanatory, but let's just go over what separates the two so that we're on the same page.

There are three criteria that I go by to determine whether a system is classified as simple or complex.

The first criteria is the number of goals involved. A simple system has only one goal. That means that all pieces and steps of the system are focused on producing just one output.

An example might be writing a blog post. I have a system for how I produce a piece of content for my website. And when I operate that system, the only outcome at the end of it is a completed piece of writing. It's very straightforward.

On the other hand, a complex goal could have multiple goals or outcomes that are intertwined. For instance, if I wanted to promote or market a piece of my content while it's being produced, that would significantly change the way that I would need to go about creating it, making the whole process more, well, complex.

The second criteria for a simple system is that it's linear. And what I mean by that is, when you operate it, it flows directly from the beginning to the end. The first step leads to the second step. The second step leads to the third step. So on and so forth until you reach the end.

At the end of each step in the system, you simply move onto the next one.

A basic checklist is a relatable example. When you finish one step, you check it off and move onto the next. There's a clear linear progression from start to finish. No jumping around.

In a more complex system, you might repeat steps, loop back to previous steps, or even skip steps. There are lots of good reasons to do that based on what the goal of

the system is. If a system requires that kind of operation, then it would be a complex system.

Third, a simple system contains no complicated logic. That means there's no decision making at any point in the process. To operate the system, you don't have to do any analysis or branch off into a different directions based on different conditions or anything.

Again, it's a simple progression. Step one. Then step two. Then step three.

This is what my simple writing system looks like. The steps to complete a piece of content are clear and I don't really need to make any decisions along the way to know what step to do next. You could really just call this a writing checklist. Like we just learned, a checklist is a simple system that's designed to make sure you don't forget anything as you operate it.

But I also have a more complex writing system. When I'm creating something that requires a lot of research, for instance, it's really hard to compile all the necessary source material in one step. I might get it all on the first try, but I also might not.

So, my more complex writing system includes some controls. At different points in the process, I have to sort of switch roles from writer to editor and go over the piece to decide if more research is needed and, if so, where.

This kind of creates a branch in the system that prevents it from always taking a direct path from beginning to end. And it requires some role-switching and some decision-making.

It works really well, but it's more involved, so that's why I call it a complex system.

So, those are the three criteria to determine if a system is simple or complex. One: Does it have only one goal? Two: Can you proceed straight from the first step to the last step without any interruption? And three: Can you operate it without using any conditional logic between steps?

If you can answer yes to those three questions, you have a simple system. And if you answer no to any of them, you have a complex system.

Now, why does any of this matter? Who cares if a system is simple or complex?

Defining simple and complex systems is more of an academic exercise than a practical one, but it *is* useful to understand the differences, and here's why:

When you're setting out to create a new system it's almost always best to try to start with a simple one. One goal. Straight path. No logic.

Now, that won't always work. Lots of times you *need* a complex system to get the best results. But it's much easier to get started and test a new system by creating a simple one.

And sticking to those three criteria are a great way to limit yourself when you're designing a new system. It's a sort of self-regulation that keeps you from overcomplicating things when you're working from a blank slate.

That better ensures that you actually get a system created and have something to use and test quickly. It allows you to iterate faster without getting bogged down in the details.

So, when is a simple system your best choice?

Like I just said, it's almost always the best choice when you're creating a *new* system. And there are plenty of times when creating a simple system is all you need, and there's no use turning it into a complex one.

Whenever the result you're trying to get from your system is not complex, then a simple system is probably your best bet.

A morning routine that gets you ready for the day is a good example. There isn't really any complex result you're trying to achieve. You just want to get some important things done every morning that help you have the best day you're capable of.

In all likelihood, your morning routine can be a simple system that never needs to become more complex.

A simple system is also going to be all you need when you're in control of the outcome of each step of the system. There's really no need for any conditional logic or decision making when you know with certainty that, if you do a step, you're going to get a defined result.

A meal planning system could be another example. You could pick 7 dinner recipes from a list each week and, once you have that, you know exactly what to do next—go to the grocery store and buy the ingredients. Once you have the ingredients, you cook them on their designated days.

Each step kind of stands alone. There's only one result that can happen when you complete it, so there's no need to evaluate the results of a step and decide on a course of action. You just move onto the next one.

Of course, certain ingredients you need could be out of stock at the grocery store. Or you might need to change your meal plan mid-week based on feedback from other people you might be cooking for.

There are certainly lots of ways that the system could become a complex one. But, when you're first creating it, it's conceivable that it could be—and continue to be—a simple system for a long time because you control the important variables.

Alright, now that we understand simple systems pretty well, how do we create one?

Well, you're already very well prepared to create some simple systems because much of what you've been practicing up to now has been doing exactly that. You've probably created and even improved a simple system or two already.

Let's go over the critical steps of simple system creation and actually mock one up as we go so that you can see all the pieces come together.

For this exercise, we'll assume that we really like to write and we want to build a simple writing system to help us write a really great blog post as quickly as possible. (Gee, I wonder where I got that idea?)

Now, we may have some preconceived ideas about what a great blog post is since we've all read many of them. But our first step is going to be to forget about that and go back to our first principles. We're going to build from the ground up.

So, what is the ultimate goal of a blog post? What are the first principles?

Many writers might answer that question differently but, when I write, my goal is often to persuade. I want to teach people something they didn't know before and convince them it's important.

So, I'm going to design my writing system to meet that goal.

That means, before I start writing, I need to do some pre-work. So, the first steps of my system are going to include things like deciding what topic I want to write about, making an inventory of what I already know about the topic and listing the things I need to learn more about. Then, I'll do some research to fill out or expand my knowledge in those areas so that I can make a compelling argument.

Then, I build an outline of the article around whatever angle it is I want to present using the research from the last step.

Next, I'd flesh out the article into a first draft. Then, I would do a high level edit just to find any parts that either need more information to be complete or to delete any extra information I included that doesn't really help my argument and doesn't need to be there.

I want look professional, so the next thing I do is run the whole piece through a spell-checker and look for any grammar mistakes.

After that, I'll create headings throughout the article so that people can scan it easily to find the most interesting information to them.

I also like to explain things visually, so another step towards the end of the process will include illustrating critical points with a simple diagram.

Finally, I'll load the article into my website and follow all the little technical steps to get it published.

And there you have it, a simple system to complete a blog post.

We built it from first principles to make sure it was unique to us and did what we wanted it to. We identified all the steps that need to completed in order to get it published, and we put those steps in the right order so that the whole process has a clean, linear flow.

We have to do creative and sometimes challenging work within each step, but we always know exactly what to do next after finishing each one.

And since this is a new system, there's one more step we need to complete, but we're going to do it over time. We need to give the system a test run and look for problems.

If you're a naturally analytical person, this step kind of happens intuitively as you work your system and find sticking points. But, if you're not naturally analytical, you'll want to schedule that kind of analysis in.

New systems almost always need at least a few revisions to get them running smoothly and, if you don't make that a priority, you'll be more likely to get frustrated with your system and give up on it when it doesn't work right rather than fix it.

So, that's a runthrough for how to build a simple system. In reality, I've been creating content for many years now, so my writing system has developed into a much more complex one that takes into consideration a lot of different factors.

That's normal. As you develop your own systems, you'll probably find that the most important ones turn into more complex ones because complex systems are better at handling diversity and changing circumstances.

But that's what we'll go over in our next lesson.

For now, why don't you pick a job you regularly complete and build your own simple system using the same constraints and principles we went over in this lesson.

As a reminder, the constraints are a singular goal, a linear progression, and no decision making between steps.

Try to engineer your system so that you either complete the work faster or you get more reliable results from it.

Good luck. When you're done, I'll see you in the next lesson.