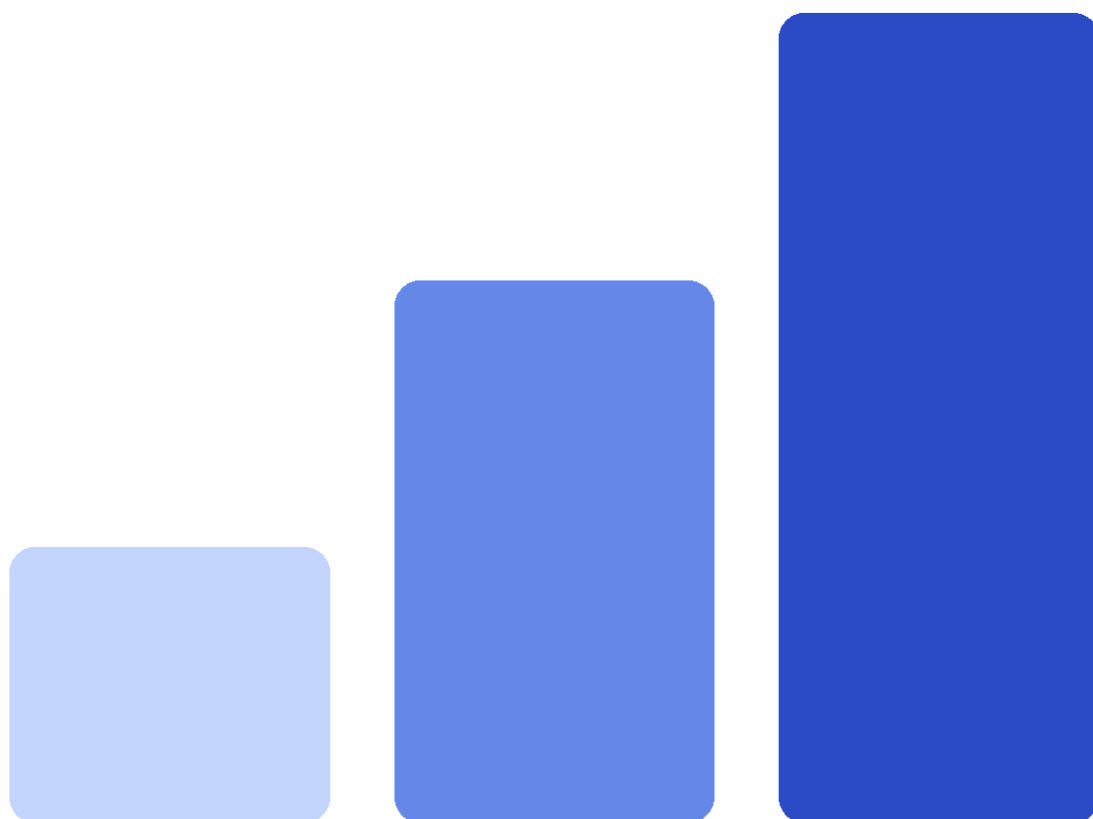


A THREE-DAY PRACTICE BOOK

Vibe Coding

Stepping Stones

From Zero Code to Your Own Data Page Online
in 3 Days



Vibe Coding Stepping Stones

From Zero Code to Your Own Data Page Online in 3 Days

by Jay Jung Lim

BuildnWrite

Contents

Preface	4
Why vibe coding needs these three days.....	4
What this book promises.....	4
What this book doesn't promise.....	5
What you need.....	5
How to read this book.....	5
 Day 1. First steps in Python, in the browser	7
Chapter 1. Run Python without installing anything	8
Exercise 1-1: Set up Colab	8
Chapter 2. Values and variables	12
Exercise 1-2: Variables and output	12
Exercise 1-3: Working with strings.....	14
 Full table of contents	17
 Next steps	20
Get the free checklist	20
The practice kit	20
The author's agent system	20
A 60-minute call	20

Preface

Why vibe coding needs these three days

Vibe coding means building programs by telling an AI what you want instead of typing the code yourself. You write “read this file and compute the average” in a chat window, and code comes out. That is why so many people say you no longer need to know any code.

Then you actually start, and you get stuck on day one. You don’t know where to paste the code the AI gave you. You don’t know what “run it in the terminal” means. When an error appears, you can’t tell whether it is your mistake or the AI’s. When you want to show someone the result, you don’t know where to put it. The AI writes the code for you, but taking that code, running it, checking it, and publishing it is still a person’s job. This book is about that part.

The material in this book comes from a three-day introductory Python course for adults without a technical background, taught in Seoul. The course ran five hours a day for three days, and every participant finished with their own data page on the internet. It has been rewritten so you can follow it alone, without a class.

This book is adapted from the author’s Korean course materials. The practice data is a real open dataset of Seoul apartment sales, and the data keeps its original Korean column names and district names. You never need to read Korean: every Korean word you meet is data you copy and paste, and the book tells you what each one means where it first appears.

What this book promises

Three things.

First, you will be able to read Python code. Variables, lists, conditionals, loops, dictionaries, functions. Most of the code an AI writes for you is a combination of these six. You type each of the six yourself and trigger the errors yourself.

Second, you will have a workspace on your own computer. You open a terminal, install Python, create a file, and run it, until that flow is in your hands. Every vibe coding tool runs on top of that flow.

Third, a page built from real public data will have its own internet address. You read 38,846 Seoul apartment sales records, pull out the numbers for a district you care

about, put the result on a web page, and publish it with GitHub Pages. By the end of Day 3 you have an address of your own that opens on your phone.

What this book doesn't promise

Reading this book won't make you a developer. It doesn't cover all of Python, and topics like classes and exception handling don't appear. It doesn't teach you to build a web service or an app. It makes no promise about making money.

What this book does is lay one stepping stone. When these three days are over, you will understand what vibe coding tools are saying to you, and you will be able to decide for yourself what to learn next.

What you need

- One computer. macOS or Windows, either is fine. Where the commands differ, both are written side by side.
- A Google account. You use it to sign in to the Day 1 practice environment (Google Colab).
- The practice data file `seoul_apr_2026H1.csv`. You use it from Day 2. Download it from https://buildnwrite.com/downloads/seoul_apr_2026H1.csv.
- A GitHub account. You use it on Day 3 to publish your page. Signing up in advance makes Day 3 easier.
- One AI chat. ChatGPT, Claude, or Gemini, whichever you can use for free. You start using it in the second half of Day 2.
- Three to five hours per day. Each day has five chapters, and each chapter takes 30 minutes to an hour. You can spread one day's work over several days.

Free with this book: a 10-minute checklist that tells you whether your computer is ready for a coding agent such as Claude Code or Codex. Leave your email and you get the checklist and news of new editions of this book: <https://jaylim5.gumroad.com/l/vibe-coding-checklist>

How to read this book

This isn't a book to read. It is a book to follow along with. Every exercise has the same shape.

- **Starting point:** where you begin and what you should be looking at
- **Done when:** what must appear on the screen for the exercise to be finished

- **Why:** what the thing you just did means
- **If you get stuck:** the common sticking points and the way out

When the screen matches what “Done when” describes, move on. If it doesn’t, read “If you get stuck”. If that doesn’t help either, paste the error message into an AI chat exactly as it appears and ask. That is part of vibe coding too.

The exercises often tell you to trigger an error on purpose. Don’t skip those. You are meeting the errors you will run into later, in a safe place, first.

The appendix at the end of each day has a dictionary of the errors you met that day and reference solutions for the exercises. Don’t look at the solutions first. Use them to compare when “Done when” isn’t showing up.

Don’t just read the code. Type it. Copying and pasting is faster, but the typos and errors you meet while typing by hand are what this book teaches.

Day 1. First steps in Python, in the browser

Today has one goal: run Python in the browser without installing anything, and read and write code that stores values, makes decisions, and repeats.

Chapter	Topic	Exercises
Chapter 1	Setting up	Exercise 1-1
Chapter 2	Values and variables	Exercises 1-2, 1-3
Chapter 3	A container for many values: the list	Exercises 1-4, 1-5
Chapter 4	Making decisions: conditionals	Exercises 1-6, 1-7
Chapter 5	Repeating: loops	Exercises 1-8, 1-9

There are two kinds of exercises

- **Guided exercises** (1-1, 1-2, 1-3, 1-4, 1-6, 1-8): the code is printed in this book. You check your own work against the “Done when” of each exercise.
- **Problem solving** (1-5, 1-7, 1-9): you solve problems on Programmers, a Korean coding-practice site with automatic grading. The book gives only the problem links and hints. Any similar practice site works if you prefer one in English.

Not finishing every problem is normal. If one problem holds you for more than 8 minutes, read the hint, and if that doesn’t help, move on. Many problems click into place when you come back after finishing the next chapter.

Today you install nothing on your computer. Every computer has a different operating system and different settings, and if you start with installation you spend most of the first day on “why does it not work on mine”. Installation is on Day 2.

Chapter 1. Run Python without installing anything

Exercise 1-1: Set up Colab

What you do and why

- You make yourself a workspace where you can use Python in the browser.
- You install nothing on your own computer. You use a computer Google lends you, through the browser.

Goals

- ☐ Open Colab and create a new notebook
- ☐ Turn off AI autocompletion
- ☐ Write code in a cell, run it, and see the result
- ☐ Know which part of the screen is for writing code and which part shows results

Starting point

You need only two things: a browser and a Google account. Go to the address below, sign in, and start from there.

`colab.research.google.com`

You start from an empty screen. Creating the notebook yourself is half of this exercise.

Four steps.

Step	What to do	Where to click
1	Open Colab with your Google account	<code>colab.research.google.com</code>
2	Create a new notebook	The blue New notebook button near the top
3	Turn off AI autocompletion	See the box below
4	Write code in a cell and run it	Click the cell, then <code>Shift + Enter</code>

Turn off AI autocompletion. This matters.

When it is on, typing just `pri` fills in the rest in gray. Press Tab and the code is written for you, and the time you were going to spend typing today disappears. The errors you meet while typing are today's lesson.

How to turn it off

1. Click the **gear icon** at the top right of the screen (to the left of the Share button)
2. In the list on the left, click **AI Assistance**
3. Uncheck the top item, **Show AI-powered inline completions**
4. Click **Close**

There are three items; only the top one needs to be off. Leave the other two as they are. The layout may change over time. If the names differ, look for the item that mentions "inline completions" and turn that off.

How to run a cell

Click the box (the cell) where you wrote the code and press `Shift + Enter`. Clicking the play button to the left of the cell works too.

The first run takes a few seconds to a few tens of seconds while Colab connects to a Google server. When the word **Connect** at the top right turns into a green check mark, you are connected.

Done when

Type the following into the empty cell of your new notebook, yourself, and run it. When all three results appear, you are done.

```
[cell]
print("Hello Python")
```

```
[output]
Hello Python
```

```
[cell]
3 + 4
```

```
[output]
7
```

```
[cell]
print(3 + 4)
```

```
[output]
7
```

If the title at the top left looks like `Untitled0.ipynb`, the notebook is being saved automatically to your Google Drive.

Why do two cells show the same result in different ways

The second cell shows the result without `print`. The third prints it with `print`.

Colab shows the **last line** of a cell even without `print`. On Day 2, when you run files on your own computer, nothing appears without `print`. So from today, make `print` a habit.

If you get stuck

You can't find the New notebook button. On the first screen of `colab.research.google.com` there is a blue **New notebook** button near the top, with an Upload notebook button next to it. If you don't see the buttons, you aren't signed in. Sign in with your Google account at the top right. The Recents list lower down shows notebooks you made before. If this is your first time, it is empty, and that is normal.

You can't find the gear icon. It is at the top right of the notebook screen, just left of the blue **Share** button. If you don't see the menu, maximize the browser window. In a narrow window the icons collapse. You can also find Settings under the **Tools** menu at the top.

You ran a cell and nothing happened. If the top right still says **Connect**, you aren't connected to the server yet. The first run takes a few seconds to a few tens of seconds. Wait a moment. If it still doesn't run, open the **Runtime** menu at the top, restart the runtime, and run again from the top cell.

You want to make a new cell. Click the + **Code** button near the top and a new cell appears below. Hovering between cells also shows an add button. The shortcut is `Ctrl + M` then `B` (on a Mac, `Cmd + M` then `B`). Cells must be run in order from top to bottom. Skipping around causes problems. See "When Colab will not listen" in the Day 1 appendix for details.

Reference solution

None. You're done when the screen opens and the cell runs.

The practice kit includes a completed notebook with all of the Day 1 exercises (`day1/day1-python-basics.ipynb`). When your own notebook gets tangled, open it with **Upload notebook** in Colab's **File** menu and compare. Don't use it at first. Build your own from an empty notebook.

Screen path check: the path to Colab's AI autocompletion setting was confirmed on the actual screen on July 30, 2026. Google's official documentation doesn't describe this path, so it couldn't be re-confirmed from the documentation on October 9, 2026.

Chapter 2. Values and variables

Exercise 1-2: Variables and output

What you do and why

- You give a value a name, store it in a variable, and take it out again by that name. This is the most common thing in programming.
- Think of one spreadsheet cell holding one value. A Python variable is simpler than that. One box, one value, one label on the box.

Goals

- ☐ Put a value in a variable and take it out with `print()`
- ☐ Know the difference between a number and a string (`3` and `"3"` are different)
- ☐ Check the type of a value with `type()`
- ☐ Read an error message and find the cause

Starting point

Type the following into a new cell, exactly as written.

```
name = "Jay"
age = 38

print(name)
print(age)
print(type(name))
print(type(age))
```

Done when

```
[output]
Jay
38
<class 'str'>
<class 'int'>
```

`str` means string and `int` means integer.

Once that works, try the following in a new cell, one at a time.

#	What to do	Expected result
1	Change <code>name</code> to your own name and run again	Your name appears
2	<code>print(age + 10)</code>	48
3	<code>print(name + age)</code>	An error
4	<code>print(name + str(age))</code>	Name and age joined together

The error in step 3 is this one.

```
TypeError: can only concatenate str (not "int") to str
```

Why trigger an error on purpose

In plain words the message says “you can only attach a string to a string, not an integer”. Text and numbers are different kinds of values, so they can’t be added. Turn the number into text with `str(age)` and they join.

Over the next three days you will meet hundreds of errors. An error isn’t a breakdown. It is Python telling you something. Start practicing reading them now.

If you get stuck

You get a `NameError`.

```
NameError: name 'nmae' is not defined
```

A typo in a variable name. If the name you used to store the value and the name you used to take it out differ by even one letter, Python says it doesn’t know that name. Look at the name in the error message and find where it went wrong. Uppercase and lowercase are different names too. `Name` and `name` aren’t the same. If there is no typo and you still get this error, you didn’t run the cell above. See “When Colab will not listen” in the Day 1 appendix.

You get a `SyntaxError`.

```
SyntaxError: unterminated string literal (detected at line 1)
```

You wrote only one of the two quotation marks, like `"Jay` with no closing quote. A string is always wrapped in a pair of quotes. The line number in the error message tells you which line has the problem.

You want to print several values at once. Separate them with commas. This way works even when the kinds differ.

```
print(name, age)
```

+ joins values; a comma prints them side by side. They aren't the same thing.

Reference solution

Exercise 1-2 under "Reference solutions" in the Day 1 appendix.

Exercise 1-3: Working with strings

What you do and why

- You cut, join, and count values made of characters.
- There is a reason strings come first. A string is a row of characters in order, and you can pull them out by position number.
- The list you learn in Chapter 3 works on the same principle. Learn it on strings and lists become easy.

Goals

- ☐ Take a specific character out of a string by its position number, called its index
- ☐ Know that numbering starts at 0
- ☐ Cut out part of a string with slicing
- ☐ Count the length with `len()`

Starting point

```
word = "hello world"

print(word[0])
print(word[1])
print(len(word))
print(word[0:3])
```

Done when

```
[output]
h
e
```

```
11
hel
```

Why count from 0

To get the first character you write `word[0]` , not `word[1]` . People count from 1; Python counts from 0. With 11 characters, the numbers run from 0 to 10.

`word[0:3]` means from number 0 **up to but not including number 3**. So you get three characters: 0, 1, and 2. The end number isn't included. Finding this confusing is normal. Change the numbers a few times and check, and it sticks.

Once that works, try these one at a time.

#	What to do	What to look for
1	<code>print(word[-1])</code>	The first character from the end
2	<code>print(word[6:])</code>	From number 6 to the end
3	<code>print(word[:3])</code>	From the start up to number 3
4	<code>print(word * 2)</code>	What happens when you multiply a string
5	<code>print(word[20])</code>	An error
6	<code>print(word.replace("hello", "goodbye"))</code>	Swap one part for another

The error in step 5 is this one.

```
IndexError: string index out of range
```

It means the number is out of range. There are 11 characters and you asked for number 20.

Step 4 connects to the first problem in Chapter 3. Multiply a string by a number and it repeats that many times. Keep that in mind.

If you get stuck

You lose count of the numbers. Write the characters out on paper one by one and number them from 0 underneath.

```
h e l l o   w o r l d
0 1 2 3 4 5 6 7 8 9 10
```

A space counts as a character too. Number 5 is the space.

You can't memorize the slicing rule. Don't memorize it. Check it in a cell. That is the advantage of Colab.

```
s = "abcde"
print(s[1:3])
print(s[:2])
print(s[2:])
```

Working out the rule from the results is faster than memorizing it.

replace doesn't change the original. `replace` returns a new, changed result and leaves the original alone. Check it.

```
word.replace("hello", "goodbye")
print(word)
```

The original is unchanged. To use the changed version, you have to catch it.

```
new_word = word.replace("hello", "goodbye")
print(new_word)
```

Reference solution

Exercise 1-3 under “Reference solutions” in the Day 1 appendix.

Full table of contents

This sample contains the preface, Day 1 chapters 1 and 2, and Next steps. The full book is organized as follows.

Get the full book: <https://leanpub.com/vibe-coding-stepping-stones> (also on Gumroad: <https://jaylim5.gumroad.com/1/vibe-coding-stepping-stones>)

- **Preface**
 - Why vibe coding needs these three days
 - What this book promises
 - What this book doesn't promise
 - What you need
 - How to read this book
- **Day 1. First steps in Python, in the browser**
- **Chapter 1. Run Python without installing anything**
 - Exercise 1-1: Set up Colab
- **Chapter 2. Values and variables**
 - Exercise 1-2: Variables and output
 - Exercise 1-3: Working with strings
- **Chapter 3. A container for many values: the list**
 - Exercise 1-4: Working with lists
 - Exercise 1-5: Problem solving
- **Chapter 4. Making decisions: conditionals**
 - Exercise 1-6: Working with conditionals
 - Exercise 1-7: Problem solving
- **Chapter 5. Repeating: loops**
 - Exercise 1-8: Working with loops
 - Exercise 1-9: Putting it all together
- **Day 1 appendix**
 - Errors you met today
 - When Colab won't listen
 - Reference solutions
- **Day 2. Moving to your own computer**
- **Chapter 6. The terminal and paths**
 - Exercise 2-1: Finding your way in the terminal

- Exercise 2-2: Building the vibe-lab workspace
- **Chapter 7. Installing Python and running your first file**
 - Exercise 2-3: Installing Python with uv
 - Exercise 2-4: Running your first file with VS Code
- **Chapter 8. A container you search by label: the dictionary**
 - Exercise 2-5: Working with dictionaries
 - Exercise 2-6: Dictionary problem solving
- **Chapter 9. Functions and modules**
 - Exercise 2-7: Bundling code into a function
 - Exercise 2-8: Using other people's code with import
- **Chapter 10. Coding with AI**
 - Exercise 2-9: Getting the AI to write what you wrote by hand
 - Exercise 2-10: Reading generated code and fixing one thing
- **Day 2 appendix**
 - Errors you met today
 - When installation gets stuck
 - Terminal commands at a glance
 - Reference solutions
- **Day 3. Build a data page and show it to the world**
- **Chapter 11. Your first HTML page**
 - Exercise 3-1: Getting the AI to build your first page skeleton
 - Exercise 3-2: Opening it in a browser, checking, and changing something
- **Chapter 12. Analyzing the data you will show**
 - Exercise 3-3: Writing a question about your district and getting the AI to code it
 - Exercise 3-4: Checking whether the result makes sense, then going further
- **Chapter 13. Putting the result on the page**
 - Exercise 3-5: Adding the table to index.html
 - Exercise 3-6: Checking the finished page locally
- **Chapter 14. Uploading to GitHub**
 - Exercise 3-7: Creating a GitHub repository
 - Exercise 3-8: Uploading index.html and confirming one commit
- **Chapter 15. Deploying with GitHub Pages**
 - Exercise 3-9: Turning on GitHub Pages
 - Exercise 3-10: Confirming your URL is public

- **Day 3 appendix**
 - Errors you met today
 - When deployment fails
 - Reference solutions
- **Next steps**
 - Get the free checklist
 - The practice kit
 - The author's agent system
 - A 60-minute call
- **About the author**
- **Copyright**

Next steps

Three days are done. You have a workspace on your computer, you can read, run, and fix code an AI wrote, and you have an address of your own on the internet. From here there are four ways to keep going.

Get the free checklist

A 10-minute checklist to rerun on any new computer before you install a coding agent (Claude Code, Codex, and the like). It goes item by item through whether the workspace you built in this book is the starting point those tools expect. Leave your email and you get the checklist and news of new editions of this book.

<https://jaylim5.gumroad.com/l/vibe-coding-checklist>

The practice kit

The bundle of practice files for this book. It contains the completed Day 1 notebook, the Day 2 Python files, the Day 3 page template and example page, the processed Seoul apartment sales data, a document combining the error dictionaries of all three days, and the collection of AI prompts from the book. It is for comparing when your own files get tangled while following the book, and for asking different questions of the same data after you finish. The kit is \$29.

<https://jaylim5.gumroad.com/l/vibe-coding-stepping-stones-kit>

The author's agent system

Once your first page is online, the next step is making AI do your work the way you work. The author's agent system is the set of 20 skills, 6 reviewer and worker subagents, and a 6-role one-person operations team he runs his business with, ready to install in Claude Code or Codex. It comes bundled with his other book, The Claude Code & Codex Skills Guidebook (\$59).

<https://jaylim5.gumroad.com/l/guidebook-skill-pack>

A 60-minute call

A paid 60-minute call with the author to set up Claude Code on your machine and plan your first automation (\$199). We untangle the place where installation wouldn't finish,

or take one task you want to automate and design which tool to use and in what order.
Book the call on this page.

<https://jaylim5.gumroad.com/l/setup-consulting>