
**Learning Python Programming with 11 Beginner Tips**

We are so excited that you have decided to embark on the journey of learning Python! One of the most common questions we receive from our readers is “What’s the best way to learn Python?”

I believe that the first step in learning any programming language is making sure that you understand how to learn. Learning how to learn is arguably the most critical skill involved in computer programming.

Why is knowing how to learn so important? The answer is simple: as languages evolve, libraries are created, and tools are upgraded. Knowing how to learn will be essential to keeping up with these changes and becoming a successful programmer.

In this article, we will offer several learning strategies that will help jump start your journey of becoming a rockstar Python programmer!

**Table of Contents**



## Make It Stick

Here are some tips to help you make the new concepts you are learning as a beginner programmer really stick:

### Tip #1: Code Everyday

Consistency is very important when you are learning a new language. We recommend making a commitment to code every day. It may be hard to believe, but muscle memory plays a large part in programming. Committing to coding everyday will really help develop that muscle memory. Though it may seem daunting at first, consider starting small with 25 minutes everyday and working your way up from there.

### Tip #2: Write It Out

As you progress on your journey as a new programmer, you may wonder if you should be taking notes. Yes, you should! In fact, research suggests that taking notes by hand is most beneficial for long-term retention. This will be especially beneficial for those working

towards the goal of becoming a full-time developer, as many interviews will involve writing code on a whiteboard.
Once you start working on small projects and programs, writing by hand can also help you plan your code before you move to the computer. You can save a lot of time if you write out which functions and classes you will need, as well as how they will interact.

### Tip #3: Go Interactive!

Whether you are learning about basic Python data structures (strings, lists, dictionaries, etc.) for the first time, or you are debugging an application, the interactive Python shell will be one of your best learning tools. We use it a lot on this site too!
To use the interactive Python shell (also sometimes called a "Python REPL", first make sure Python is installed on your computer. We've got a step-by-step tutorial to help you do that. To activate the interactive Python shell, simply open your terminal and run `python` or `python3` depending on your installation.

Now that you know how to start the shell, here are a few examples of how you can use the shell when you are learning:

**Learn what operations can be performed on an element by using dir():**

```
>>> my_string = 'I am a string'
>>> dir(my_string)
['__add__', ..., 'upper', 'zfill']  # Truncated for readability
```

The elements returned from `dir()` are all of the methods (i.e. actions) that you can apply to the element. For example:

```
>>> my_string.upper()
>>> 'I AM A STRING'
```

Notice that we called the `upper()` method. Can you see what it does? It makes all of the letters in the string uppercase!

**Learn the type of an element:**

```
>>> type(my_string)
>>> str
```

**Use the built-in help system to get full documentation:**

```
>>> help(str)
```

**Import libraries and play with them:**

```
>>> from datetime import datetime
>>> dir(datetime)
['__add__', ..., 'weekday', 'year']  # Truncated for readability
>>> datetime.now()
datetime.datetime(2018, 3, 14, 23, 44, 50, 851904)
```

**Run shell commands:**

```
>>> import os
```

```
>>> os.system('ls')
python_hw1.py python_hw2.py README.txt
```

### Tip #4: Take Breaks

When you are learning, it is important to step away and absorb the concepts.
The Pomodoro Technique is widely used and can help: you work for 25 minutes, take a short break, and then repeat the process. Taking breaks is critical to having an effective study session, particularly when you are taking in a lot of new information.
Breaks are especially important when you are debugging. If you hit a bug and can't quite figure out what is going wrong, take a break. Step away from your computer, go for a walk, or chat with a friend.
In programming, your code must follow the rules of a language and logic exactly, so even missing a quotation mark will break everything. Fresh eyes make a big difference.

### Tip #5: Become a Bug Bounty Hunter

Speaking of hitting a bug, it is inevitable once you start writing complex programs that you will run into bugs in your code. It happens to all of us! Don't let bugs frustrate you. Instead, embrace these moments with pride and think of yourself as a bug bounty hunter.
When debugging, it is important to have a methodological approach to help you find where things are breaking down. Going through your code in the order in which it is executed and making sure each part works is a great way to do this.
Once you have an idea of where things might be breaking down, insert the following line of code into your script `import pdb; pdb.set_trace()` and run it. This is the Python debugger and will drop you into interactive mode. The debugger can also be run from the command line with `python -m pdb <my_file.py>`.

### Make It Collaborative

Once things start to stick, expedite your learning through collaboration. Here are some strategies to help you get the most out of working with others.

### Tip #6: Surround Yourself With Others Who Are Learning

Though coding may seem like a solitary activity, it actually works best when you work together. It is extremely important when you are learning to code in Python that you surround yourself with other people who are learning as well. This will allow you to share the tips and tricks you learn along the way.
Don't worry if you don't know anyone. There are plenty of ways to meet others who are passionate about learning Python! Find local events or Meetups or join PythonistaCafe, a peer-to-peer learning community for Python enthusiasts like you!

### Tip #7: Teach

It is said that the best way to learn something is to teach it. This is true when you are learning Python. There are many ways to do this: whiteboarding with other Python lovers, writing blog posts explaining newly learned concepts, recording videos in which you explain something you learned, or simply talking to yourself at your computer. Each of these strategies will solidify your understanding as well as expose any gaps in your understanding.

### Tip #8: Pair Program

Pair programming is a technique that involves two developers working at one workstation to complete a task. The two developers switch between being the “driver” and the “navigator.” The “driver” writes the code, while the “navigator” helps guide the problem solving and reviews the code as it is written. Switch frequently to get the benefit of both sides.
Pair programming has many benefits: it gives you a chance to not only have someone review your code, but also see how someone else might be thinking about a problem. Being exposed to multiple ideas and ways of thinking will help you in problem solving when you got back to coding on your own.

### Tip #9: Ask “GOOD” Questions

People always say there is no such thing as a bad question, but when it comes to programming, it is possible to ask a question badly. When you are asking for help from someone who has little or no context on the problem you are trying to solve, its best to ask GOOD questions by following this acronym:

- **G**: Give context on what you are trying to do, clearly describing the problem.
- **O**: Outline the things you have already tried to fix the issue.
- **O**: Offer your best guess as to what the problem might be. This helps the person who is helping you to not only know what you are thinking, but also know that you have done some thinking on your own.
- **D**: Demo what is happening. Include the code, a traceback error message, and an explanation of the steps you executed that resulted in the error. This way, the person helping does not have to try to recreate the issue.

Good questions can save a lot of time. Skipping any of these steps can result in back-and-forth conversations that can cause conflict. As a beginner, you want to make sure you ask good questions so that you practice communicating your thought process, and so that people who help you will be happy to continue helping you.

### Make Something

Most, if not all, Python developers you speak to will tell you that in order to learn Python, you must learn by doing. Doing exercises can only take you so far: you learn the most by building.

### Tip #10: Build Something, Anything

For beginners, there are many small exercises that will really help you become confident with Python, as well as develop the muscle memory that we spoke about above. Once you have a solid grasp on basic data structures (strings, lists, dictionaries, sets), object-oriented programming, and writing classes, it's time to start building!

What you build is not as important as how you build it. The journey of building is truly what will teach you the most. You can only learn so much from reading Real Python articles and courses. Most of your learning will come from using Python to build something. The problems you will solve will teach you a lot.

There are many lists out there with ideas for beginner Python projects. Here are some ideas to get you started:

- Number guessing game
- Simple calculator app
- Dice roll simulator
- Bitcoin Price Notification Service

If you find it difficult to come up with Python practice projects to work on, watch this video. It lays out a strategy you can use to generate thousands of project ideas whenever you feel stuck.

### Tip #11: Contribute to Open Source

In the open-source model, software source code is available publicly, and anyone can collaborate. There are many Python libraries that are open-source projects and take contributions. Additionally, many companies publish open-source projects. This means you can work with code written and produced by the engineers working in these companies.

Contributing to an open-source Python project is a great way to create extremely valuable learning experiences. Let's say you decide to submit a bug fix request: you submit a "pull request" for your fix to be patched into the code.

Next, the project managers will review your work, providing comments and suggestions. This will enable you to learn best practices for Python programming, as well as practice communicating with other developers.

### Go Forth and Learn!

Now that you have these strategies for learning, you are ready to begin your Python journey!

Happy Coding!