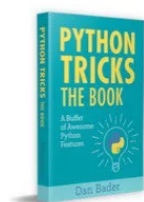


# 11 Beginner Tips for Learning Python Programming

by [Krishelle Hardson-Hurley](#) ⌚ Apr 03, 2018 💬 📌 [basics](#) [python](#)

## Table of Contents

- [Make It Stick](#)
  - [Tip #1: Code Everyday](#)
  - [Tip #2: Write It Out](#)
  - [Tip #3: Go Interactive!](#)
  - [Tip #4: Take Breaks](#)
  - [Tip #5: Become a Bug Bounty Hunter](#)
- [Make It Collaborative](#)
  - [Tip #6: Surround Yourself With Others Who Are Learning](#)
  - [Tip #7: Teach](#)
  - [Tip #8: Pair Program](#)
  - [Tip #9: Ask “GOOD” Questions](#)
- [Make Something](#)
  - [Tip #10: Build Something, Anything](#)
  - [Tip #11: Contribute to Open Source](#)
- [Go Forth and Learn!](#)



**“I don’t even feel like I’ve scratched the surface of what I can do with Python”**

**Write More Pythonic Code »**

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.

Improve Your Python

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

**Free PDF Download: [Python 3 Cheat Sheet](#)**

## 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 every day. It may be hard to believe, but everyday will really help develop that m with 25 minutes everyday and working y

Check out the [First Steps With Python G](#)

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

### Improve Your Python

...with a fresh  **Python Trick**  code snippet every couple of days:

Email Address

Send Python Tricks »

### 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. You can find more specific directions [here](#).

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():*

Python

```
>>> 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:

Python

```
>>> 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 more about these built-in methods under “[Manipulating strings](#)” in this tutorial.

*Learn the type of an element:*

Improve Your Python

Python

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

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

Python

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

Import libraries and play with them:

Python

```
>>> from datetime import datetime
>>> dir(datetime)
['__add__', ..., 'weekday', 'year']
>>> datetime.now()
datetime.datetime(2018, 3, 14, 23, 4
```

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

## Improve Your Python

...with a fresh  **Python Trick**   
code snippet every couple of days:

Email Address

Send Python Tricks »

Run shell commands:

Python

```
>>> 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

Improve Your Python

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 some ways to do this: whiteboarding with others, recording videos in which you explain something, or writing articles. These strategies will solidify your understanding.

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

### Improve Your Python

...with a fresh  **Python Trick**   
code snippet every couple of days:

[Send Python Tricks »](#)

### Tip #8: Pair Program

[Pair programming](#) is a technique that involves two developers switch between being the “navigator” helps guide the problem solver and the “driver” who writes the code.

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:

[Improve Your Python](#)

- 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 sources are shared with the community. You can contribute to Python libraries that are open-source projects. This means you can work on projects that interest you or help companies.

[Contributing to an open-source Python project](#) says you decide to submit a bug fix request.

Next, the project managers will review your request and provide you with the best practices for Python programming.

For additional tips and tactics that will help you break into the open-source world, check out the video embedded below:

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

## Improve Your Python



...with a fresh  **Python Trick**   
code snippet every couple of days:

[Send Python Tricks »](#)

## Go Forth and Learn!

Now that you have these strategies for learning, you are ready to begin your Python journey! Find Real Python’s Beginners Roadmap for Learning [here](#)! We also offer a beginner’s level [Python course](#), which uses interesting examples to help you learn programming and web development.

Happy Coding!

 Python Tricks 

Get a short & sweet **Python Trick** delivered to your inbox every couple of days. No spam ever. Unsubscribe any time. Curated by the Real Python team.


Improve Your Python



Email Address


```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  Python Trick  code snippet every couple of days:

Email Address


About **Krishelle Hardso**




After 6 years of teaching high school math, Krishelle switched careers and now works as a Site Reliability Engineer at Dropbox in San Francisco, CA.

[» More about Krishelle](#)


Each tutorial at Real Python is created by a team of developers so that it meets our high quality standards. The team members who worked on this tutorial are:



Dan





Joanna



Kyle

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python with  Python Tricks 

Get a short & sweet Python code snippet delivered to your inbox every couple of days:

[» Click here to see examples](#)

What Do You Think?

**Real Python Comment Policy:** The most useful comments are those written with the goal of learning from or helping out other readers—after reading the whole article and all the earlier comments. Complaints and insults generally won’t make the cut here.

Improve Your Python

https://realpython.com/python-beginner-tips/#make-it-stick

6/9

7 Comments


Real Python

1 Login

Recommend 1

Share

Sort by Best



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

**Kashif** • 5 months ago

Great tips. Tip#2 clicks me with a with Python.

3 ^ | v • Reply • Share ›

**Derrick Kearney** • 5 months ago

Good tips Krishelle

2 ^ | v • Reply • Share ›

**Bert Kurdes** • 12 days ago

Love your tip about "teaching" :) manage to teach someone that n teach properly I know my exact s just with a video camera and a s skills as well. And if you're serious about it use real people as your test subjects by taking advantage of youtube or educational platforms like bitdegree

^ | v • Reply • Share ›

**lingen2** • 2 months ago

These are all great tips.

One more thing I cannot recommend highly enough: Take the "python challenge" (first result on google). It consists of a number of riddles that can be solved using all sorts of Python language features. Once you've solved a puzzle, you can view other solutions and compare to what you came up with. This is a great way to learn.

^ | v • Reply • Share ›

**Vladimir Astrelin** • 3 months ago

Thanks a lot for your great tips, they are really helpful !

^ | v • Reply • Share ›

**The Red Rose** • 4 months ago

There is so much to learn and I do not know where to start! Your tips will be extremely helpful as go along ..

^ | v • Reply • Share ›

**Godson Chijioke** • 4 months ago

These tips are really helpful....Especially taking notes.

^ | v • Reply • Share ›

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

## Improve Your Python

...with a fresh  **Python Trick**   
code snippet every couple of days:

Email Address

Send Python Tricks »

### ALSO ON REAL PYTHON

#### Advanced Git Tips for Python Developers

6 comments • 5 days ago

**Dan Bader** — Ah, so you'd prefer to get just the title and description instead of the full article text included in the ...

#### What Can I Do With Python?

5 comments • 2 months ago

**Luis Orduz** — Yeah, for the web frontend we're pretty much limited to JS and its derivates (TypeScript,

#### Variables in Python – Real Python


10 comments • 2 months ago

**John Sturtz** — Hi Prashant. Thanks for the question. Let me see if I can clarify this.When you initially make a variable

#### Building and Documenting Python REST APIs With Flask and

17 comments • 2 months ago

**Doug Farrell** — Thanks for the feedback Matt. Postgress is a great database, but would require readers to

 Subscribe  Add Disqus to your siteAdd DisqusAdd



“I wished I had access to a book like this when I started learning Python many years ago”

Learn More »

Keep Reading

basics python

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh Python Trick code snippet every couple of days:

Email Address

Send Python Tricks »

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

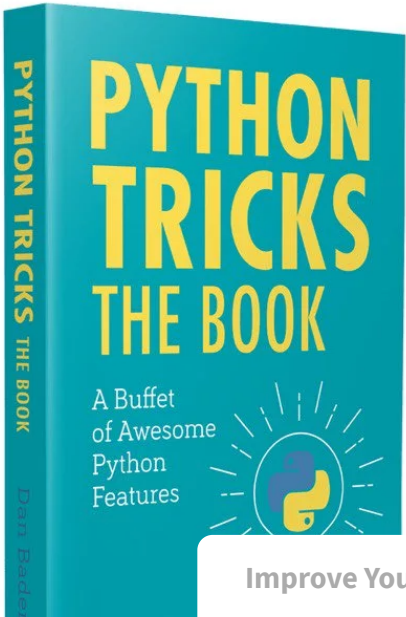
Email...

Get Python Tricks »

No spam. Unsubscribe any time.

All Tutorial Topics

- advanced api basics best-practices community databases data-science
- devops django docker flask front-end intermediate machine-learning
- python testing tools web-dev web-scraping



Improve Your Python





# Download a Free Chapter »

## Table of Contents

- [Make It Stick](#)
- [Make It Concrete](#)
- [Make Some Noise](#)
- [Go Forth and Teach](#)

```
1 # How to merge two dicts
2 # in Python 3.5+
3
4 >>> x = {'a': 1, 'b': 2}
5 >>> y = {'b': 3, 'c': 4}
6
7 >>> z = {**x, **y}
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

## Improve Your Python



...with a fresh **Python Trick** code snippet every couple of days:

[Send Python Tricks »](#)