# **The Best Python Tutorials**

[Python](https://www.python.org/) is a general purpose programming language which is dynamically typed, interpreted, and known for its easy readability with great design principles.

freeCodeCamp has one of the most popular courses on Python. It's completely free (and doesn't even have any advertisements). You can [watch it on YouTube here](https://www.youtube.com/watch?v=rfscVS0vtbw).

## **Want to learn more?**

The web is a big place, there’s plenty more to explore:

* Python Practice Book: <http://anandology.com/python-practice-book/index.html>
* Think Python: <http://greenteapress.com/thinkpython/html/index.html>
* Practical Business Python: <http://pbpython.com/>
* Another course: [https://realpython.com/?utm*source=fsp&utm*medium=promo&utm\_campaign=bestresources](https://realpython.com/?utm_source=fsp&utm_medium=promo&utm_campaign=bestresources)
* General: <https://www.fullstackpython.com/>
* Learn the Basics: <https://www.codecademy.com/learn/learn-python>
* Computer science using Python: [https://www.edx.org/course/introduction-computer-science-mitx-6-00-1x-11?ref=hackernoon#!](https://www.edx.org/course/introduction-computer-science-mitx-6-00-1x-11?ref=hackernoon" \l "!)
* List of more resources for learning python: <https://github.com/vinta/awesome-python>
* Interactive Python: <http://interactivepython.org/runestone/static/thinkcspy/index.html>
* Developer’s Guide to Python: <https://devguide.python.org/>

## **What Is Python Used for?**

Python can easily be used for small, large, online and offline projects. The best options for utilizing Python are web development, simple scripting and data analysis. Below are a few examples of what Python will let you do:

##### **Web Development:**

You can use Python to create web applications on many levels of complexity. There are many excellent Python web frameworks, including Pyramid, Django and Flask, to name a few.

##### **Data Analysis:**

Python is the leading language of choice for many data scientists. Python has grown in popularity within the field due to the availability of many excellent libraries focused on data science (of which NumPy and Pandas are two of the most well-known) and data visualisation (like Matplotlib and Seaborn). Pyton really made Data crunching fun with all its numerous available libraries. Ipython with JupyterLab is another form of Python that improves the usage of Python in the field of data science.

##### **Automation:**

Python is a very flexible language that can be used to automate boring or repetitive tasks. System administrators often use it by writing scripts which can be easily executed from the terminal. Python can also be used to create bots which automate some of our daily tasks.

##### **Artificial Intelligence:**

Python is also used extensively in the growing field of Artificial Intelligence (AI). Google selected Python to be one of the first well-supported programming languages for training and interacting with models using Tensorflow.

#### **Mobile app development**

Mobile apps and games can be created with python using Kivy, Pygame and PyQt.

#### **Security And Networking:**

Python is used to build Networking tools and security-tools that are widely used. Python’s remote automation is the most secure, fast and efficient for cloud-testing of frameworks. That’s why professional developers use python for making the most secure frameworks and for socket-programming.

##### **Machine Learning, Deep Learning**

Python is one of the best languages suited for machine learning, deep learning and data analytics with a stronghold in all of them.

There are specialized languages best suited for various role such as R and MATLAB, but when it comes to overlapping fields of applications python wins hands down due to its flexibility and rapid prototyping and availability of libraries.

#### **Telegram Bots Development**

You can use Python and some Python-libraries for developing your own [Telegram Bots](https://core.telegram.org/bots).

#### **Data Gathering by crawling and scraping**

Python can also be used to parse pages' source codes and retrieve its data. Using some python modules, such as Scrapy and also (for some pages which use javascript) Selenium should do the trick!

#### **Python is generally used for**

* Web & Internet Development
* Educational Advancement
* Scientific Studies/Computing
* Desktop development
* Numeric Computing
* Software development
* Business Application development
* Machine Learning
* IOT
* Game Development
* Rapid Prototyping
* Browser Automation
* Data analysis
* Scraping data from websites
* Image Processing

#### **Some articles covering the usability of python**

* [10 Major Uses of Python](http://www.dummies.com/programming/python/10-major-uses-of-python/)
* [Applications for Python](https://www.python.org/about/apps/)
* [Where is the Python Language Used?](https://stackoverflow.com/questions/3043085/where-is-python-language-used)
* [What is Python used for?](https://stackoverflow.com/questions/1909512/what-is-python-used-for)

The official package index for python is [here](https://pypi.python.org/pypi).

## **Should you use Python 2 or Python 3?**

The two versions are similar. If you know one, switching to writing code in the other is easy.

* [Python 2.x will not be maintained past 2020.](https://www.python.org/dev/peps/pep-0373/)
* 3.x is under active development. This means that all recent standard library improvements, for example, are only available by default in Python 3.x.
* Python ecosystem has amassed a significant amount of quality software over the years. The downside of breaking backwards compatibility in 3.x is that some of that software (especially in-house software in companies) still doesn’t work on 3.x yet.

## **Installation**

Most \*nix based operating systems come with Python installed (usually Python 2, Python 3 in most recent ones). Replacing your system's default installation of Python is not recommended and may cause problems. However, different versions of Python can be safely installed alongside your system's default version. See [Python Setup and Usage](https://docs.python.org/3/using/index.html).

Windows doesn’t come with Python, but the installer and instructions can be found [here](https://docs.python.org/3/using/windows.html).

## **Python Interpreter**

The Python interpreter is what is used to run Python scripts.

If it is available and in Unix shell’s search path, it's possible to start it by typing the command python followed by the script name. This will invoke the interpreter and run the script.

hello\_campers.py

print('Hello campers!')

From the terminal:

$ python hello\_campers.py

Hello campers!

When multiple versions of Python are installed, calling them by version is possible depending on the install configuration. In the Cloud9 IDE custom environment, they can be invoked like:

$ python --version

Python 2.7.6

$ python3 --version

Python 3.4.3

$ python3.5 --version

Python 3.5.1

$ python3.6 --version

Python 3.6.2

$ python3.7 --version

Python 3.7.1

## **Python Interpreter Interactive Mode**

Interactive mode can be started by invoking the Python interpreter with the -i flag or without any arguments.

Interactive mode has a prompt where Python commands can be entered and run:

$ python3.5

Python 3.5.1 (default, Dec 18 2015, 00:00:00)

GCC 4.8.4 on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> print("Hello campers!")

Hello campers!

>>> 1 + 2

3

>>> exit()

$

## **The Zen of Python**

Some of the principles that influenced the design of Python are included as an Easter egg and can be read by using the command inside Python interpreter interactive mode:

>>> import this

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than \*right\* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

## **Pros and Cons of Python**

### **Pros**

1. Interactive language with a module support for almost all functionality.
2. Open Source: So, you can contribute to the community, the functions you have developed for future use and to help others
3. A lot of good interpreters and notebooks available for better experience like jupyter notebook.

#### **Cons**

1. Being open source, many different ways have developed over the years for the same functions. This sometimes creates chaos for others to read someone's else code.
2. It is a slow language. So it's a very bad language to use for developing general algorithms.

## **Documentation**

[Python is well documented](https://docs.python.org/3/). These docs include tutorials, guides, references and meta information for language.

Another important reference is the Python Enhancement Proposals ([PEPs](https://www.python.org/dev/peps/)). Included in the PEPs is a style guide for writing Python code, [PEP 8](https://www.python.org/dev/peps/pep-0008/).

## **Debugging**

Inline print statements can be used for simple debugging:

… **often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.**  
  
[Executive Summary](https://www.python.org/doc/essays/blurb/)

Python also includes more powerful tools for debugging, such as:

* logging module, [*logging*](https://docs.python.org/3/library/logging.html)
* debugging module, [*pdb*](https://docs.python.org/3/library/pdb.html)

Just note that these exist for now.

## **Hello World!**

Going back to the docs, we can read about the [print](https://docs.python.org/3/library/functions.html" \l "print) function, a [*built-in function*](https://docs.python.org/3/library/functions.html) of the [Python Standard Library](https://docs.python.org/3/library/index.html).

print(\*objects, sep=' ', end='\n', file=sys.stdout, flush=False)

The built-in functions are listed in alphabetical order. The name is followed by a parenthesized list of formal parameters with optional default values. Under that is a short description of the function and its parameters are given and there is occasionally an example.

The [print](https://docs.python.org/3/library/functions.html" \l "print) function in Python 3 replaces the [print](https://docs.python.org/2/reference/simple_stmts.html" \l "print) statement in Python 2.

>>> print("Hello world!")

Hello world!

A function is called when the name of the function is followed by (). For the Hello world! example, the print function is called with a string as an argument for the first parameter. For the rest of the parameters the defaults are used.

The argument that we called the print function with is a str object or *string*, one of Python’s *[built-in types](https://docs.python.org/3/library/stdtypes.html" \l "text-sequence-type-str)*. Also the most important thing about python is that you don’t have to specify the data type while declaring a variable; python’s compiler will do that itself based on the type of value assigned.

The objects parameter is prefixed with a \* which indicates that the function will take an arbitrary number of arguments for that parameter.