Introduction

Python is a programming language that is easy to learn, which is why many novice coders choose it as their first language. Because it was built as a general-purpose language, it is not limited to just one type of development - you can use it for anything from analyzing data, to creating games. Python has also become incredibly popular in the scientific community because scientists use it to calculate complicated equations and analyze data.

Whether you are new to programming or simply interested in learning another language, we have the resources to help you get started. Take a look at our comprehensive guide that introduces you to the basics of the language, and then takes you all the way through creating your first programs.

We recommend starting at the top and working your way through each section, even if you have prior experience.

Table of Contents

What is Python?

In technical terms, Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.

Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers.

Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy to import or export these modules.

One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.

If this description of Python over your head, don't worry. You'll understand it soon enough. What you need to take away from this section is that Python is a programming language used to develop software on the web and in app form, including mobile. It's relatively easy to learn, and the necessary tools are available to all free of charge.

That makes Python accessible to almost anyone. If you have the time to learn, you can create some amazing things with the language.

How is Python Used?

Python is a general-purpose programming language, which is another way to say that it can be used for nearly everything. Most importantly, it is an interpreted language, which means that the written code is not actually translated to a computer-readable format at runtime. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects.

The concept of a "scripting language" has changed considerably since its inception, because Python is now used to write large, commercial style applications, instead of just banal ones. This reliance on Python has grown even more so as the internet gained popularity. A large majority of web applications and platforms rely on Python, including Google's search engine, YouTube, and the web-oriented transaction system of the New York Stock Exchange (NYSE). You know the language must be pretty serious when it's powering a stock exchange system.

In fact, NASA actually uses Python when they are programming their equipment and space machinery. Pretty neat, right?

Python can also be used to process text, display numbers or images, solve scientific equations, and save data. In short, it is used behind the scenes to process a lot of elements you might need or encounter on your device(s) - mobile included.

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

Benefits of Learning Python

There are many benefits of learning Python, especially as your first language, which we will discuss.

It is a language that is remarkably easy to learn, and it can be used as a stepping stone into other programming languages and frameworks. If you're an absolute beginner and this is your first time working with any type of coding language, that's something you definitely want.

Python is widely used, including by a number of big companies like Google, Pinterest, Instagram, Disney, Yahoo!, Nokia, IBM, and many others. The Raspberry Pi - which is a mini computer and DIY lover's dream - relies on Python as its main programming language too. You're probably wondering why either of these things matter, and that's because once you learn Python, you'll never have a shortage of ways to utilize the skill. Not to mention, since a lot of big companies rely on the language, you can make good money as a Python developer.

Other benefits include:

1. Python can be used to develop prototypes, and quickly because it is so easy to work with and read.
2. Most automation, data mining, and big data platforms rely on Python. This is because it is the ideal language to work with for general purpose tasks.
3. Python allows for a more productive coding environment than massive languages like C# and Java. Experienced coders tend to stay more organized and productive when working with Python, as well.
4. Python is easy to read, even if you're not a skilled programmer. Anyone can begin working with the language, all it takes is a bit of patience and a lot of practice. Plus, this makes it an ideal candidate for use among multi-programmer and large development teams.
5. Python powers Django, a complete and open source web application framework. Frameworks - like Ruby on Rails - can be used to simplify the development process.
6. It has a massive support base thanks to the fact that it is open source and community developed. Millions of like-minded developers work with the language on a daily basis and continue to improve core functionality. The latest version of Python continues to receive enhancements and updates as time progresses. This is a great way to network with other developers.

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

Python Environment Setup

One of the most important things you'll do when working with any programming language is setup a development environment which allows you to execute the code you write. Without this, you will never be able to check your work and see if your website or application is free of syntax errors.

With Python, you also need something called an interpreter that converts your code - which makes up the entirety of your application - to something the computer can read and execute. Without this interpreter, you'll have no way to run your code.

To convert your code, you must first use a Python shell, which calls upon the interpreter through something called a "bang" line.

As for creating an application or file, there are two ways to do this. You can create a program using a simple text editor like WordPad, or Notepad++. You can also create a program using a Python shell. There are advantages and disadvantages to each method, which we'll discuss next.

Python Shell versus Text File

A shell is a program or tool that can be used to interact with a system. For instance, the Windows operating system shell can be tapped into by using a "terminal" or command line to submit commands and arguments.

With Python, things work a bit differently than an operating system shell. The Python shell is used to interact with an interpreter, which feeds code to a computer in a form that it can understand.

When you execute a Python program that you've written, the interpreter reads the code and converts it into usable commands. The important thing to note is that all of this is done after the program has been executed.

With a shell, the interpreting - or conversion - happens in real-time as you type the code into the computer or system. This means that the actual program is executing as you type. This gives you some idea of how your final code will look, and what your program is actually going to do.

When you write code in a text file, none of that happens until you feed the document into an interpreter. If you have Python installed on your computer you can call upon the interpreter using a command line, but this step is done after you've already written the code.

This makes it more difficult to spot errors in your code, and it can also be frustrating if the interpreter runs into issues, because they may not be as apparent as they would if you had used a shell. Still, a lot of developers prefer to use a text editing tool because it is simple and easy to do.

There are text editors with increased functionality - like Notepad++ - which were specifically developed with programming in mind.

The Best Place to Start

Before you can do anything with a programming language, you first need to configure the development environment. Now, we're going to cover how to setup Python and the interpreter that will execute your custom programs.

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

Python comes pre-installed on Mac and a majority of Linux distributions. However, you may need to download an updated version depending on how old your system is.

TIP: You can easily check your Python version by opening the terminal and running the following command:

python -V

If you're running Windows, you'll need to download Python from the Python Software Foundation (link below).

How Do I Get Python?

If you need to download Python, the best place to go is the official site. You'll need to download the version specific to your operating system and processor (32 or 64 bit).

Mac: Most Mac OS X computers already have Python 2.x installed, which is perfectly fine to use. The best way to install the latest version is to use Homebrew to activate and manage them. You'll find instructions on how to do that here. You can also download Python directly from the PSF if you prefer.

Linux: Python is included with most distributions of Linux. Check your current version and be sure to upgrade using the package manager, if necessary.

Windows: Just download Python from the PSF.

TIP: If you're using Windows, be sure to select the option that adds Python.exe to your system path during installation. You can do this by selecting the option next to "Add python.exe to path," and then choosing your local hard drive as the install location.

Which Version Should I Use?

There are two main versions of Python, which can make things confusing for beginners and novice coders. The two versions are Python 2.x and 3.x. The good news is that when it comes to syntax, they are pretty much identical, and it's acceptable to develop with both versions.

Version 3.5.x of Python (or greater) is currently in active development. This means that it is constantly receiving new features and functionality, as the open source community continues to develop it. If you want bleeding edge in terms of features and support, then 3.5.x is the way to go.

Python versions from 2.7.x to 3.4.x (3.2.x included) are still actively maintained by the community. This is important if you need help or encounter problems. Because it's been around so long, 2.7.x also has the most support from third-party libraries.

While looking at libraries, if you see they have not been ported to a newer version of Python, you'll want to stick with the older version.

Most importantly, once you learn one version of Python it's not difficult to make the jump to another version. If you're moving up - as in moving to a newer version - you'll simply need to learn the new features and functions. If you're moving backward you shouldn't encounter any problems although you will have to figure out what functions are incompatible with the older version.

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

Long story short, it's entirely up to you which version of Python you use! There's no right or wrong answer, and it's painless to make the jump from one version to another should you realize you need to do so later on.

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

What Features Does Python Offer?

Python is often comparable to Perl, Ruby, PHP, Scheme, and Java. This is because it is is an incredibly powerful object-oriented language.

Python also has several notable features which make it an enticing language to work with for developers.

1) Python makes use of an elegant syntax, meaning the programs you write are much easier to read. This is because they are closer to the human language, or how we write our words, instead of a language that computers use to read and interpret code. For example, the "print" command will display anything proceeding it - and in quotes - at runtime.
2) Python is simple and easy-to-use, which means that it's much easier to get your programs up and running. That is why Python is considered ideal for prototype development and similar ad-hoc programming tasks. It does not compromise maintainability either.
3) It comes with the Standard Python Library, offering integrated support for a variety of common programming tasks like syncing with web servers, searching through text, and modifying files. For a majority of other languages, you have to create this content from scratch.
4) It includes an interactive mode that simplifies testing for short snippets of code. There's even a development environment bundled with it called IDLE. The dev environment makes setup so much easier and faster.
5) The language can be extended by adding new modules, even if they've been compiled in C or C++. Even better, the modules can be used as shortcuts in future projects once they've been created.
6) Python can be embedded into an application, which will provide a programmable interface for users of that app. This is a great feature if you're putting together an app that will teach coding, or requires working with Python in a terminal.
7) It is compatible with a long list of computers and operating systems like Windows, Linux, MacOS, many brands of Unix, OS/2, and more. Furthermore, it uses a similar interface on each one of those platforms, which means you can jump between them easily if necessary.
8) It is truly free because it doesn't cost anything to download or use, and there are no licensing fees. Plus, it can be freely modified and redistributed, since the language is available under an open source license - despite the fact that it is copyrighted.

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

What is Django?

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

This is how websites - even simple ones designed by a single person - can still include advanced functionality like authentication support, management and admin panels, contact forms, comment boxes, file upload support, and more. In other words, if you were creating a website from scratch you would need to develop these components yourself. By using a framework instead, these components are already built, you just need to configure them properly to match your site.

The official project site describes Django as "a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source."

Django offers a big collection of modules which you can use in your own projects. Primarily, frameworks exist to save developers a lot of wasted time and headaches and Django is no different.

You might also be interested in learning that Django was created with front-end developers in mind. "Django's template language is designed to feel comfortable and easy-to-learn to those used to working with HTML, like designers and front-end developers. But it is also flexible and highly extensible, allowing developers to augment the template language as needed."

If you're going to be working with Python, especially for web applications or web design, you'll want to remember the Django framework. It will certainly come in handy.

CherryPy is another Python-based framework that is great to work with, although it is designed with the absolute minimalist in mind. It's a framework you'll want to explore after you already have some experience working with Python.

How Does Python Differ from Other Languages Like PHP or Ruby?

Python stands out because it is easy to learn and easy to understand. Many consider Ruby a great place to start, like Python, yet the latter has a four-year head start. This means that it has a big foothold in the enterprise world, and it's much more popular with C developers. This is because it's easy to crossover between the two languages.

Both Ruby and Python share a significant amount of growth in the job market, so choosing either language would be beneficial in terms of a career. PHP is also used often though the application is different.

Ultimately, it comes down to what you will be developing, as each language has its niche.

What Are the Languages Used for?

PHP is a server scripting language that is primarily used to create dynamic and interactive websites. It is the best language for creating HTML content, and can be used to build anything from a simple blog to a huge, corporate style website.

Python is a high-level, object-oriented general-purpose language, that is versatile and can be used for nearly anything. It is commonly used to develop web and mobile applications, website crawlers, indexers, daemons, and desktop GUI apps.

Ruby is a high-level, object-oriented language that is used to work with web application and data entities; it exists to take the focus away from query tasks. Ruby is most famous for its dynamic type system which performs type checking during runtime. It also features automatic memory management.

> *TIP: When a language is referred to as "high-level" it is because the syntax and commands it recognizes are closer to human language instead of that of a computer. The term high-level was initially used to describe languages that are not locked down to a particular type of computer.*

Out of the three languages, Python is the best for absolute beginners and is often recommended by programmers because it uses a syntax that emphasizes simplicity and ease of use. Whereas, Ruby is better used by programmers that have experience with other languages. PHP, on the other hand, is best suited for developers who are used to working with C languages.

They all have specific purposes and use-cases as you can see from the descriptions. To provide a better understanding, we'll offer some examples of who uses these languages.

Who Uses PHP?

In short, PHP was designed for web development and the creation of dynamic web pages. Brands that use PHP to power their products include:

- Udemy
- Wikipedia
- Facebook

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

- Google
- NASA

Who Uses Python?

In short, Python was designed to emphasize productivity, readability, and ease of use. Brands that use Python to power their products include:

- YouTube
- Google
- Yahoo! Map
- Shopzilla
- Ultraseek

Who Uses Ruby?

In short, Ruby was designed specifically for programmers to make the development process more fun and flexible. Brands that use Ruby to power their products include:

- Twitter
- Hulu
- Groupon
- IndieGoGo
- GitHub

So, Which Language is the Best to Learn?

It doesn't matter who you ask, the answer to this question will always be the same. There is no "best" in the world of programming because each language serves a purpose. To make it even more confusing, with the three languages discussed here, you can do just about anything as they're all general purpose languages.

There is no right or wrong answer.

Also, there are frameworks available for each one of these languages, and those make development easier. PHP has several content management systems like WordPress, Drupal, and Joomla. Python has Django and CherryPy. Finally, Ruby has Rails or Ruby on Rails.

You can't go wrong with any of them. Of course, we recommend starting with Python.

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

6 Python Programming Projects for Beginners

Once you have Python installed, you can move on to working with the language and learning the basics. To get you started, we're going to discuss several projects you can attempt, even if you have no prior programming experience.

Keep in mind, you must have Python already installed to participate.

1) Hello World

Ah, the all familiar "hello world," exercise that you do every time you start learning a new programming language. The goal here is to output a small message to introduce yourself to the language.

In Python, this is incredibly simple. All you need to do is open the interpreter and type the following:

```
print("Hello World")
print("My name is") #add your name after the word "is" obviously
```

If all goes well, you should see something like this:

```
> python3 #to call upon Python on MAC OS X use this command, for Windows use "python"
Python 3.5.1 (default, Jan 14 2016, 06:54:11)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
>>> print("My name is Bob")
Hello World
My name is Bob
```

Clearly, the command print is used to display content on the screen. Remember this command because you'll be using it often.

The text you see after the # symbol is called a comment. Comments do not appear at runtime, and are instead meant for the developers who will be working with the code. The comment we left above provides instructions for adding your name to the message. More often than not, comments will provide

Where Can I Learn Python?

- Data Camp (Python Training)
- Python TechDegree (Treehouse)
- The Complete Python Bootcamp (Udemy)

labels or quick descriptions for a snippet of code, so you can easily identify what a particular section is for.

2) Performing Calculations

Next, let's take a simple calculation and feed it through the interpreter to see what happens. Enter the following:

```
7 + 2
```

After typing in the equation above and hitting enter - to submit - you should see something like the following:

```
>>> 7 + 2
9
```

Notice how the interpreter automatically answers the equation and spits out the result?

3) Creating Your First String

A string is a sequence of characters that can be processed by a computer. The string is usually stored for manipulation later.

Strings must always begin and end with the same character, this is a requirement. In addition you can use either single or double quotations to signify a string, there is no difference between the two. The quotation marks only serve to inform Python that what's inside of them is a string.

Let's save your name as a string to call upon later. To do that, type the following into the interpreter:

```
>>> "Bob"
'Bob'
```

Congrats! You just created your first string, and this is signified by the information sent back to you. We can see that the name was saved as a string.

Now, we want to test out this string and see what kinds of things we can do with it. First, let's use multiple strings in tandem. Enter the following into the interpreter:

```
>>> "Hello there " + "my name is " + "Bob"
'Hello there my name is Bob'
```

Notice how Python adds the strings together before outputting the content?

Another neat trick you can do is multiply strings or manipulate them through equations.

```
>>> "Bob" * 4
'BobBobBobBob'
```

This may seem silly right now, as you would probably never need to multiply your name like this in the real world. However, this type of manipulation can really come in handy when you're working on large projects in Python that have a lot of strings.

To see your name in upper case - instead of using caps - try working with the following command:

```
>>> "Bob".upper()
'BOB'
```

Pretty cool, right?

4) Return the Length of a Phrase or Word

Normally, if you want to know the number of letters in a word or phrase you would just count them yourself, but that's no fun! There's actually a dedicated command to do just this!

To determine the number of letters in a word or string, type the following into the interpreter:

```
>>> len("BobIsTheGreatestEver")
20
```

You can also calculate the length (size) of a list using the same command.

```
>>> players = ['bryan', 'john', 'chris']
```

```
>>> len(players)
3
```

5) Storing Variables

Each entry in the list of "players" we created above is called a variable. Variables are nothing more than names or titles for a particular set of data so that you can store them and call upon them later. For example, the variable in the tutorial above was "players" because that's what we used to store the names of the players.

Let's create a new variable of our own:

```
>>> movie = "Terminator"
```

Our variable is "movie" and in that variable we stored the data "Terminator," as you can see.

One thing you'll notice about variables is that the interpreter doesn't return anything once the information is stored. You may be wondering how we know the variable was actually stored?

You can test this by simply entering "movie" in the interpreter and hitting enter. It should return the data stored in that particular variable, like so:

```
>>> movie
'Terminator'
```

Good job! You created your first variable! Feels great, doesn't it?

But, let's say we get sick of seeing "Terminator" as the data stored in that variable. We can change this easily.

```
>>> movie = "Cinderella"
>>> movie
'Cinderella'
```

Sweet! No more crazy robots or androids! Just a compassionate, naive girl named Cinderella who will finish all our chores for us!

You can store just about anything inside a variable, including numbers, equations, and more.

6) Comparisons

One remarkably useful - yet underrated - thing you can do with a programming language is compare sets of data. Let's try that now, using numbers.

```
>>> 7 > 2
True
>>> 9 < 1
False
>>> 6 > 2 * 4
False
>>> 3 == 3
True
>>> 5 != 2
True
```

Notice how we used two equal signs (==) to check if sets of data are equal? You must always use two equal signs if that's what you are trying to do. This is because a single equal sign, or "=" is used to assign a value to a variable.

In addition, if you want to check whether or not two values are unequal you can use an exclamation mark followed by an equal sign like so: "!="

The World is Your Oyster

If you don't like shellfish - or you're allergic - we apologize. Still, you should be pleased to know that you've completed the basic tutorials! If you feel comfortable you can move on to a full-featured tutorial that will walk you through the steps, from beginner to advanced.

The world is your oyster! Get out there and... well, get out there and shuck it!