PYTHON IN UNDER 60 MINUTES

THE DISASTER TO MASTER GUIDE TO PYTHON PROGRAMMING

MATT ELECK

Python in Under 60 Minutes

The Disaster to Master Guide to Python Programming

By Matt Eleck

Introduction

I want to thank you and congratulate you for downloading the book: “*Python in Under 60 Minutes: The Disaster to Master Guide to Python Programming.*”

Python is a simple programming language that can be used by any person. From a beginner to an expert, Python is the language for you. It is simple but powerful, yet fun to use.

This eBook will help you understand the different aspects of this programming language. This guide will be mainly targeted to beginners, and I will ease you through to this language. You do not have to be a programmer in any other language. If you know anything about computers, you can learn Python programming.

Python is available on the Windows, Mac, and Linux operating systems. Due to it being simple to use, it is very versatile in getting the job done.

You may wonder why it is named Python. You might think it is named after the reptile and wonder why. It is not; it is named after the BBC show *Monty Python’s Flying Circus*.

Let us now get into the details of this language. It is important to play around with the Python interpreter as we discuss the different aspects of using it. The only way to learn about a language – and indeed anything – is to use it firsthand.

Thanks again for downloading this book. I hope you enjoy it!

Table of Contents

My Free Gift to You

>>Click here to grab your FREE copy of “Online Scam Survival Guide.”<<

Chapter 1
Features of the Python Language

Python is a simple language that reflects the growing trends in software development. They characteristics that define this language make it very popular among all levels of users. The ability of Python to create high-level solutions to problems without much focus on the language itself makes it stand above the others. Let us look at these characteristics, as they will go a long way in helping us understand the concepts behind how this language works.

Simple

This is one of the great strengths of this language. It is simple and minimalistic. It has been described that reading Python is similar to reading English. If you know anything about code, you know that this can sound unreal, but it really shows how simple Python is. The fact that Python allows you to concentrate on the solution to the problem at hand rather than the language is quite a thing.

Easy to Learn

As you will find out as you go through this guide, Python is very easy to learn. I already alluded to this in the introduction, and believe me – this language is simple to learn.

Free and Open Source

Python is one of the software that is becoming more popular over the Internet. Free and open source software and programs are the “in thing” today. You can view the source code of Python make changes to it and distribute it. The open source community has really helped in developing Python further, and this is one of the good things about it. There is a wide community constantly improving this language to make it more powerful in providing solutions, yet remaining simple.

Python is a High-Level Language

Being a high-level language means that when you write programs in this language, you do not bother on low-level details such as memory use.

Portable

In programming languages, portability means that the programs can be used across different operating systems. Programs created in Python are highly portable. They will perform across all platforms easily. You only need to avoid using any system dependent features in the programs. You can use Python on GNU/Linux, Windows, and Macintosh.

Python is Interpreted

Programming languages need to be interpreted to a language that the machine can understand. Computers use binary code (0s and 1s), so for a program to work, it has to be converted from the language it is currently in to this language using a compiler.

When using Python, this is not needed, since Python itself converts the source code into the intermediate form known as bytecodes before translating it to the binary form readable by the computer. This is part of the reason programs using Python are highly portable - they can be used on any computer easily. The creator of the program just needs to link the proper libraries.

Supports Object-oriented Programming.

By object-oriented, we mean that Python supports object-oriented programming as well as procedure-oriented programming. In object-oriented programming, the program combines data and functionality. Python is quite powerful in this compared to the other languages. In procedure-oriented programming, the languages uses functions or procedures, which are just pieces of programs.

Embeddable

This makes it possible to embed Python within other language-supported programs to enable scripting by users of the programs.

Extensible

This means that if you want a certain piece of code to not open, you can code it in C or C++, then use it on the Python programs

Extensive Libraries

The standard library distributed with Python is very extensive. It has a wide range of facilities and built-in modules that all provide access to system functionality. These provide solutions to most common problems that programmers face. These modules, at least some of them, are designed to enhance portability of Python programs.

For Windows users, the libraries are included in the Windows installer, while for Unix users, Python will be provided as a collection of packages. You choose to obtain all or some optional components. Other than the standard library, they are other collection of components including programs, modules, and development frameworks available from the Python Package Index.

I hope you can now appreciate why Python is a powerful yet simple programming language. With all these features, you should learn and use this language. I know some of the features might not seem straightforward to a beginner, but you will get to understand them much more as we learn more about this language.

Chapter 2

Installing Python in the Different Systems

Python, being open source software, is readily available for download at no cost to you. You will need to visit the Python.org website to find the download. Python can be installed on over 20 operating systems. In this guide, we shall be looking to install it on the three main operating systems: Windows, Unix, and Mac OS.

Installation on Windows

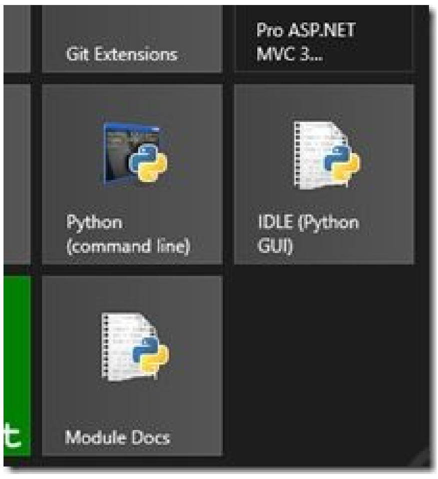
You will need to download the latest version of Python from the official website first. To be sure you are using the most recent version, use the Windows installer link from the homepage of the Python website. The Windows package will be provided as an MSI package. Double-click it and install the file. You can also automate the installation with standard tools.

A directory with the version number embedded will appear after the installation. For instance, if you install the latest version, 2.7, a directory C:\Python27\ will appear. This helps when you install multiple versions of Python. However, only one of them will be the default.

Setup Tools is third-party software that you will need to install before you start using Python. It extends the packaging and installation facilities provided in the Python standard library. It helps in the use of third-party libraries. It also enables network installation capabilities. Once you install Setup Tools, you can install any Python software with just a single command.

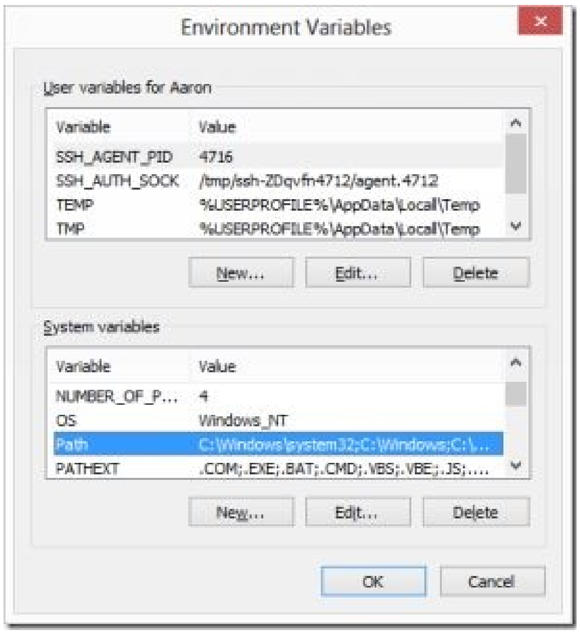
Python 2.7 is the most widely used. There is Python 3, but this one introduced many changes to the Python language with which the Python ecosystem has not yet caught up. I would advise that you stick to Python 2.7, as it will work just fine.

Once you have run the Python installer, you will see some icons appear on the Start Menu. This will signify that you have successfully installed Python and we can move to the next step



The next step is to add the Python directory to the system path environment variable to make it possible to access it from any command line prompt.

Go to Control Panel > System Properties > Environment Variables > Select Path Variable


Environment Variables

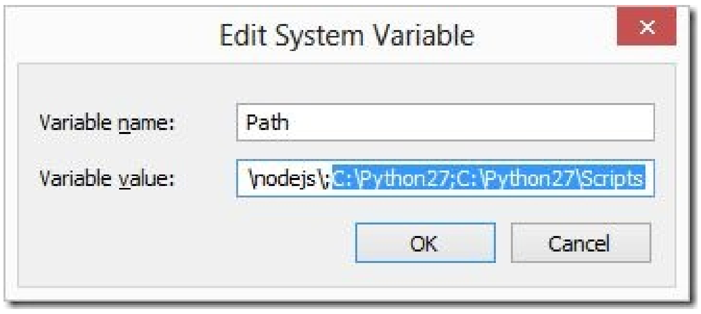
User variables for Aaron

Variable	Value
SSH_AGENT_PID	4716
SSH_AUTH_SOCK	/tmp/ssh-ZDqvfn4712/agent.4712
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
NUMBER_OF_P...	4
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

Click Edit Variable and append the Python path at the end of the string.


Edit System Variable

Variable name: Path

Variable value: \nodejs\;C:\Python27;C:\Python27\Scripts

Include the scripts, too, as this will be where the package management tools will exist. You can now access Python interpreter from any command prompt.

Install the Virtual Environment package as well. This helps in creating a local Python virtual environment for your projects. Packages in Python are installed globally. If a package dependency changes for one project running on one Python environment, it changes all. This is not good. Virtual Environment helps in avoiding this conflict. If you have already installed pip, it easy to do so – just run the following from the command line:

```
C:\> pip install virtualenv
```

Installation in Mac OS

To use Python on Mac, you will need a few tools:

Install XCode

XCode is Apple's IDE. You will need to have it to use Python smoothly. You can get XCode from the Apple Store if you do not already have it. Once you have installed it, install the Apple command line tools from the XCode Menu > Preferences > Downloads > Command Line tools, and click Install.

Install Homebrew

Homebrew is the package manager for Mac OS X. Homebrew finds and installs dependences that will be needed. Use the following code to install Homebrew.

```
$ ruby -e “$(curl -fsSL https://raw.github.com/mxcl/homebrew/go)”
```

Python will come with the latest versions of OS X. You can confirm this by typing *Python —version* in the terminal. This will also inform you of the Python version installed. If an error message pops up, Python is not installed, and you will need to do it. You can do this easily with Homebrew.

```
$ brew install Python
```

You will need to install pip. It is a package manager specific to Python. It has a single dependency that Homebrew will not be able to install. You have to do this manually.

```
$ curl -O http://Python-distribute.org/distribute_setup.py
$ Python distribute_setup.py
$ curl -O https://raw.github.com/pypa/pip/master/contrib/get-pip.py
$ Python get-pip.py
```

After installing pip, you can now install Virtual Environment, as we described in Windows installation. This will help in projects that have conflicting dependencies.

```
$ pip install virtualenv
```

Installing Python on Unix Systems

Installing Python on Unix systems is easy. For one, Ubuntu and Fedora will come with Python 2.7 out of the box. Some other versions, such as Redhat Enterprise and CentOS, will have Python 2. You will not need to install anything else other than the tools and libraries we discussed in the previous Windows section. Setup Tools and Virtual Environment are very important, as they enable use of third-party software. The installation of these in Unix is the same as Windows from the command line.

In the event that Python does not come pre-installed in your Linux distribution, you can easily get it from the repositories using the following commands:

```
$ sudo apt-get install Python2.7
```

To install in Redhat and CentOS, type the following command:

```
$ sudo yum install Python
OR
# yum install Python
```

To check which version of Python is installed in your system, type the following command:

```
$ Python —version
```

output:

```
Python 2.7.1
```

Chapter 3
Choosing a Text Editor/IDE

Before you move deep into programming, you have to choose a text editor of IDE for use as a source file. You will use this to write your code rather than on the interpreter.

IDE stands for Integrated Development Environment. It is almost the same as a text editor, but it has a lot more functionality. There are many fancy text editors and IDEs out there, and choosing which to use might be very confusing. However, as you will soon come to appreciate, an IDE will be very useful in your programming journey.

We cannot type a program at the interpreter prompt every time we want to run something. This is where the text editors come in handy. We have to type and save our codes in a file, then run the code later. These are what are known as source files. Python supports running a program from the interpreter or from the source file. A good text editor should be able to make you write the code faster and easier.

The text editors and IDEs will save your file without any formatting so that it is readable by the program, unlike word processors, which include formatting, mostly for visual purposes. That is why you cannot write code on any word processor and run it. However, text editors and IDEs will still include some formatting for visualization as well and to help you when writing; the only difference will come when saving. They will not save the formatting, but rather the plain text that will then be readable for computation.

Before we look at the various text editors available for use, especially to a beginner, let us first look at what factors you should consider before settling for any editor.

The Operating System You Will Be Using

This is the first consideration to make. Are you using Windows, Linux, or Mac? Some editors will work only on one platform and not on the others. Others will work better on one platform than on the others. In fact, most editors have been restricted to one OS. On Windows, the notepad comes in the installation as a default program. However, there some better alternatives such as Notepad++ and TextPad. On Linux, most beginners will be comfortable using GEdit or Kate. On Mac, BBEdit and TextWrangler are the most popular editors.

Editor Features

Most of the editors that come installed on the OS are barebones editors. This means that they do not support many features, and that is why they are easy to use. However, this is not a good thing, as you will need these additional features to help you when writing code. It may be difficult to learn using a more advanced editor, but it is definitely necessary, as some of the features will make your life simpler. For instance, some editors such as Vi and Emacs are very popular due to their features. However, the learning curve for these editors may be stiff for a beginner. I would suggest using the simpler editors when starting out, but still have another editor installed which you should be learning about in a small way each day.

Syntax Highlighting

This is very important, as the different parts of the code are colorized so that you can see the program when choosing and be able to visualize it. That is why Notepad on Windows is not a good text editor.

Network Capabilities

Depending on how you work, you might want an editor that can retrieve files over a network or even edit remote files. This is very possible, and will come in handy since most programmers work as a team. You may need a file that a colleague has, which your editor may retrieve over the shared network. This is very fascinating. That is why I said you might be tempted to use a simple editor, but the advanced editors are much better in terms of functionality.

Let us now look at some popular editors that you may want to consider. These are the editors/IDEs that have been known to work well with Python

PyCharm

This can be said to the most popular Python editor. It has a many features that will ease your programming experience. Some of these features are: good code completion, neat debugger, code navigation, code analysis, and support of other languages. These features are very important. For one, code completion will be a nice way to save your time. Support of other languages will also come in handy. These are languages such as HTML, CSS, and Java. Remember when we said that you could hide part of your code in another language within a program when using Python? This will be very important. This IDE has great plugin support and rich Python tools integration.

The disadvantage of Pycharm is that it is a bit slow compared to other editors. However, there are updates that continue improving it.

Wing IDE

Wing IDE is a Python only editor. It has many great features that suit professional development. Code completion and a good debugger are some great features. This IDE can also be scripted and extended in Python.

The debugger is particularly strong point for this IDE. It has breakpoints, steps, framework debug, and various inspection points. For programmers who are in math-related development and web development, this is the editor to use, as it supports matplotlib with automatic plot updates and various web frameworks such as Django, Pyramid, and Plone.

Sublime Text

This is a very powerful and popular text editor. It is known for its speed and the Goto Definition, which makes it easy to run definitions of functions and variables. In itself, the Sublime editor is a barebones editor, but has great plugins that make it work quite well.

Vi editor

This editor is found in almost all Linux distributions. It is also now available for the other platforms. It is simple to use and highly configurable (though beginners might disagree), but very powerful. In fact, if you just want to write or edit plain text, you may try using another editor. Experienced Linux users will use the Vi editor extensively in their coding or writing markup language. Due to its availability in many of distributions, learning Vi becomes very important. You may want to edit files in a Linux system or a remote server with a minimal install; it will definitely have Vi text editor.

Vim is the newer version of Vi; it means Vi improved. Vim inherits features of Vi but also adds some functionality.

Vi and Vim are the most widely used text editors in Linux. Once you get to work with this editor, you can comfortably use any other editor without any trouble. This editor does not require you to use the mouse – just the keyboard and the screen.

These editors are the most popular with Python users. You cannot go wrong with them.

Chapter 4
The Python Library

The Python library is an extensive reference index that describes the syntax and semantics of the Python language. The library is very extensive, offering a wide range of functions. It is important to learn more about this library to understand this language. In fact, the library is a major reason why Python is considered a great programming language.

The library provides built-in modules that provide access to system functionality. Some of these modules are written in C to provide Python programmers with these functionalities. Other modules will be written in the Python language to provide solution to everyday programming problems. Other modules are created to remove the platform specifics to make programs independent of any platform.

If you are using Python on Windows, you will have access to the entire library as well as other additional components. The library will be included in the Windows installer. For Unix systems, you will find the Python library as a package or a collection of packages. You will use the packaging tool provided with the operating system to get access to the library just as you would in other programs.

In the Python library, there are data types which can be said to be the core of any language. We shall be looking extensively at these data types to gain a good understanding of them. The language defines the literal and places constraints on their semantics in these data types.

Other than the standard library, there are other many components developed by individual programmers who are part of the Python community. These components, which may be modules or programs, are meant to ease the programming challenges. However, let us first look at some of the functions in the standard library.

Comments

Any line in the interpreter or source file that starts with # is a comment. The interpreter will skip it. These comments are meant to explain something or note something. In Python, unlike other languages you might have used, there are no multiline comments. Each comment line should start with #. Comments are important in code, but cannot improve your code. Your main focus should be making your code readable.

Variables

The variable is the most basic concept of any programming language. Variables are used to process values in programming. For instance, a variable called "x" can be associated with value 3. In Python, we can say that `x = 3.` This means that this statement assigns the value 3 to x. In other terms, it binds the name x to the number three.

A variable in Python can be anything provided the following rules are met:

- Must only have ASCII digits, characters, and underscores. This means no spaces or any other symbol.
- A variable cannot start with a digit.
- Do not use words reserved from a list of keywords.

Numbers

Numbers in Python are either integers or floating-point numbers. This generally means whole numbers and numbers with a decimal point.

Integers are whole numbers ranging from minus infinity to plus infinity. This may be a little bit confusing, but in general, it means you can use any number if your computer will be able to type it and keep memory of it. Of course, you may not have such a number you want to use. Python will use two types of integers: the integer and the long integer. This means that for very large numbers, you might want to use them as long integers. However, in normal circumstances, you will not need these long integers.

A floating-point number is a number with a decimal point, meaning it is not a whole number. 2.3 is a floating point number. Even with presumably whole numbers such as 5.0, the language will interpret this as a floating point number. Computers "like" dealing with integers rather than floating-point numbers, so unless you know you will have the floating-point number, just use the integers. It will be much faster for you and the computer. The standard library will provide built-in capabilities to deal with both integers and floating-point numbers. However, if your program will deal with a lot of floating point numbers, you may look for an extension such as jumpy or Multiprecision Project that will help in speeding things up.

Chapter 5
Data Structures

When coding, you will need to group information together and work on it in a group. Python allows you to do this with some powerful concepts. Data structures help you group data and store values together. We shall look at four data structures in Python: the list, the tuple, the set, and the dictionary. These are very powerful tools that will come in handy in your programming. There is a wide range of operations which you can perform on them. Let us now take a closer look at each of these four data structures.

Lists

Lists are versatile data types that are written as a list of comma separated values between square brackets. It is that simple; creating a list just involves writing items separated by a comma, then putting the square brackets. The items need not be of the same type. They can be anything. This makes the list the most versatile data type in Python.

list1 = [‘Biology’, ‘English’, 1997, 2000];

list2 = [1, 2, 3, 4, 5];

list3 = [“a”, “b”, “c”, “d”];

You can access values in lists by slicing along the index or indexes and obtaining the value available at that index. To update the list, slice the left side of the assignment operator. To delete or remove an item from a list, use the del statement if you are sure of the item to be deleted, or remove if not sure.

You can access the individual members of a list by indexing into it. This simply means that you use an integer representing the position of the member on the list. The first member of the list in in position 0. You then count up to the desired member and assign the corresponding number. If you attempt to access a position that does not, exist you get an index error. You can also access the list by using negative indexes. Position -1 is the last item.

Tuples

Tuples are similar to lists, but more rigid. The major difference is that in tuples, items cannot be deleted, removed, or altered in any way. This makes tuples faster than lists. A programmer should therefore use tuples when they are sure that items used will not change. In tuples, you can use the square brackets or the parentheses. Use a comma to separate the items just like in lists.

Sets

Sets provide classes, which help in constructing and manipulating unordered collections of unique items. Put simply, they are lists with no duplicate entries. Since sets are unordered, they do not record items' position or the order of insertion, so slicing and indexing is not possible. Sets are mostly used for removing duplicates in a sequence, membership testing, and computing standard math operations.

To create a set, you either use the built in function set( ) or place items inside curly braces {}, separating each item with a comma. These items can be of different types, the only exception being it cannot have mutable items as its element.

Dictionaries

The Python Dictionaries are an extremely useful and flexible data type. They store values with keys. When looking up the values, you use the key. Besides storing values, you can use the dictionaries to build complex nested data structures.

Chapter 6
Learning to Code in Python

You have probably learnt another language before Python, or you have decided to learn Python as your first language. There are some concepts that will make you learn better and be a good programmer. There are three essential skills that any beginner in Python has to know. They are attention to detail, reading and writing, and spotting differences. Python is an incredibly easy language to learn if you take time to apply these three tips.

Reading and Writing

You have to know how to read and write to learn how to code in Python. As we observed earlier, Python is almost like English, and you will read a lot, as well as doing some writing. You will also need to type the sometimes odd characters in source code. Another thing is that when learning about the different code in Python in this book and elsewhere, do not just read the code. Type it in your computer. Type it and make it run. If you find an error, read again and type again. This is the only way that you will learn fast. Just reading and seeing might add theory to you but you need to practice and practice a lot. Set aside some hours each day to code a few things. Once you type a code and it runs and do it again, you will rarely forget. Avoid the temptation of being lazy and just copy pasting the code into the interpreter. Type all the code yourself. This is the way to learn. Train your hands and brain to write code, and soon enough you will be good at it.

Attention to Detail

This is another thing you should have. It is what that separates mediocre programmers from good programmers. Never assume anything, and be attentive even to the slightest of details. When it comes to code, everything counts. Pay attention to all the details and do not miss any element. It is easier to learn just one thing at a time and learn it well than rushing through many things and, at the end of the day, failing to grasp anything. If you do not pay attention, your programs will always have bugs that will need more time to fix.

Spotting Differences

As a programmer, you need to have a keen eye to spot differences between things, and even more so with code. Some codes might appear to be the same, but are not, and serve different functions. Most developers develop these skills over time, but it would be useful if you start it earlier. It will come in handy. Even the slightest of differences in programming can make all the difference. When practicing code, you will often type code samples in your computer to run; sometimes, you will get errors, meaning you have done something wrong. You should be able to compare your code with the source code and spot the difference. This way, you learn to spot mistakes in your own work and fix bugs well in advance.

Chapter 7
Reasons Why You Should Use Python

Python is becoming more popular as a programming language due to its versatility. It is simple to use, yet very powerful. The quality of Python’s features also makes it quite a favorite for many programmers and organizations. Big organizations such as Google are using Python as one of their official programming languages. YouTube is almost entirely coded using Python. In addition to the features, we have the numerous data structures, the nested functions, the classes, the extensive standard library, and the outstanding documentation, among many other features that will be discussing in this chapter. When using Python, you will also have access to numerous third-party open source libraries that will come in handy in specific situations. A few examples here are Cython for low-level optimization, Numpy for numerical operations, IPython for interactive work, and MatPlot Lib for plotting. We had already seen some of these in earlier chapters. Let us now look at the reasons why you should be using Python.

Holistic Language Design

Python finds application in wide area of programming. It is not limited to certain areas like many languages. Some may argue that this generalization is not good, but it is actually the opposite. The extensive use of Python, from web design, to system administration, to numerical operations, shows the versatility of this language in problem solving. When you can program well in Python, you concentrate more on providing a solution rather than the language itself. You develop a real programming skill. Different programming problems call for different solutions. A language that provided a linear procedural style might solve a problem, but an object-oriented language will provide for more functional programming. In Python, you can pass everything as an object. This makes Python a useful language for problem solving, thus its popularity.

Readability

Readability is a primary consideration in Python syntax, unlike some other languages.

Being readable makes sharing the code and explaining it to others very easy – not just for the open source community, but also people working on a certain project. This leads to better codes when people get to understand what others are thinking and coding.

Balance of High-Level and Low-Level Programming

Python balances high-level programming with low-level optimization. The Python code is very high-level in some cases; a code that takes up one line in Python might take up to six lines in C and C++. However, this brings with it the disadvantage of sacrificing code speed with programming speed, as any other high-level language would behave. In Python, this problem is overcome by dealing with high-level objects. This speeds up the program. Matrices and arrays are good at doing this.

Language Interoperability

As a universal language, Python needs to glue other languages together, and it does well here. You can use many functions from other languages in Python. You will just need to find the relevant library that enables you do so. For instance, if you want to call a function from MATLAB, you can use MLabWrap to ease the transition between these two languages. If you have C or C++ libraries you want to call, Ctypes, Cython, or SWIG are three ways to easily interface to it.

Documentation System

Python incorporates class, function, and module and method documentation into the language. Here you will have two levels of comments. The programming level comments are those you start with # and will be ignored by the compiler. The documentation comments are specified by a doc string immediately after the function or method string. The doc strings are tags to the methods that anyone can access when using an interactive Python shell.

Data Structures

For good programming, you have to use the correct data structure for your code. Often, coders will ignore or under-emphasize the need for good use of the data structures. This then leads to scalability issues in the code. Coders need to learn and use good design patterns. In Python, we have the lists, tuples, sets, and dictionaries which I dedicated a whole chapter to in this eBook. There are many others such as strings, queues, and treads.

Conclusion

There are thousands of reasons why you should learn Python in this age. Even if you are not yet ready to make the switch from your current programming language to Python, learning how to code in it is a useful skill.

The Python community is also another thing which should attract you. Many Python community programmers are always eager to help you. In most of the cases, any issue you encounter will most likely have been resolved in the forums. So take your time, sign up on a forum, and go through the conversations to learn more about what is happening in the Python world. When you finally become good at it, give back to the community by helping others. This is what makes the Python community and other open source communities great and such a big force.

We can list many reasons why you should be coding using Python, but the most important are the fact that you have the liberty to experiment and try out new things, you have a language that will mean you focus on the problem rather than the code, and its extensive library. These are some of the reasons you should learn Python.

My Other books

Arduino in Under 60 Minutes: The Ultimate Disaster to Master Beginners Guide

[Beginning of Excerpt]

The Arduino is an open source platform (microcontroller) for building electronic projects. Our computers might be good for programming, but they generally do not have the capability to sense the outside environment. If you are running a project that will require gathering data to feed into a program, you need a microcontroller such as the Arduino to collect and feed this data into the computer, or even work on it without needing the computer. It can work as a standalone component in many projects. One quick example would be a project where you need to collect the temperatures in a room, which will then advise a certain function in a program, like a switch on the air conditioner or heater. The computer in itself can do this. The Arduino will collect the temperatures of the room easily.

Unlike other microcontrollers, the Arduino comes in handy as you can easily load a program to it without needing any additional hardware. Previous circuit boards required that you had some hardware to load code into them. For Arduino, you just need to use a USB cable to load new code into the board. The Arduino IDE uses a simplified version of C++, which makes it easier for beginners and even experienced coders to write code.

According to the makers of Arduino, they made this circuit board so that designers, hackers, hobbyists, students, and artists could easily express their ideas and implement their projects. This is true since this circuit board will interact with a wide range of input devices such as buttons, motors, cameras, speakers, LEDs, GPS units, a smartphone, TV, the Internet, and many more. Actually, this list is endless; it will depend on the requirements of individual projects. This flexibility is mainly brought about by the fact that the Arduino software is free and the hardware is very cheap. You can experiment with them a lot. Besides this, Arduino has a large community of users, just like many other open source programs, who are always ready to help.

Arduino is designed to be used by even a beginner in electronics. It consists of the hardware you purchase and the software or the source code editor, which is downloaded for free. However, it is still popular with experienced users and has been used all over the world in different kinds of projects. Robots, home automation motors, games, toys, sensors, and controlling lights are some of the areas in which it has found lots of usage. For a beginner, there are many examples of projects and their source code on the Internet that you can see. These will help in advising your current idea and how you can use the Arduino to implement it.

To summarize, the Arduino comes as a circuit board with a microcontroller in it. The board helps the user attach various input and output devices, connect to the power supply, and connect to the PC or to other circuit boards for programming tasks. You can use the microcontroller without the board, but in most instances, you will find the board useful for the aforementioned uses. There is something called a shield which will further help you interface with other devices such as an ethernet connection, LCD, or a joystick. We shall be talking more about these later. I hope that this is a good introduction to Arduino before we get into the deeper workings of it.

[End of Excerpt]

Arduino

IN

UNDER 60 MINUTES

The Ultimate Disaster to Master Beginners Guide

MATT ELECK