PYTHON PROGRAMMING

FOR BEGINNERS

BASIC LANGUAGE FROM ABSOLUTE BEGINNERS TO INTERMEDIATE.
LEARN EASILY AND FAST DATA SCIENCE AND WEB DEVELOPMENT
IN A SIMPLE AND PRACTICAL WAY STEP-BY-STEP

MARK CODING

Python Programming for Beginners

Basic Language from Absolute Beginners to Intermediate. Learn Easily and Fast Data Science and Web Development In a Simple and Practical Way Step-by-Step

by
Mark Coding

© Copyright 2019 - All rights reserved.
The content contained within this book may not be reproduced, duplicated or transmitted without direct written permission from the author or the publisher.
Under no circumstances will any blame or legal responsibility be held against the publisher, or author, for any damages, reparation, or monetary loss due to the information contained within this book. Either directly or indirectly.
Legal Notice:
This book is copyright protected. This book is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part, or the content within this book, without the consent of the author or publisher.
Disclaimer Notice:
Please note the information contained within this document is for educational and entertainment purposes only. All effort has been executed to present accurate, up to date, and reliable, complete information. No warranties of any kind are declared or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice. The content within this book has been derived from various sources. Please consult a licensed professional before attempting any techniques outlined in this book.
By reading this document, the reader agrees that under no circumstances is the author responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to, — errors, omissions, or inaccuracies.

Table of Contents

Introduction
Chapter 1: Presentation of Python
- How is Python Used?
- The Benefits of Working with Python

Chapter 2: The Language of Python
Chapter 3: First Step with Python
- First Steps with Mac OS X
- First Steps with Windows
- First Steps with Linux

Chapter 4: The Basics of Python
- Running Your Program
- Parts of the Code to Know

Chapter 5: Python Programmable Calculator
Chapter 6: Dictionaries and Python Data Structuring
- What is a Python Dictionary?
- Creating Our Python Dictionary
- Methods for Python Dictionaries
- The Advantages of Using the Dictionary

Chapter 7: Strings and Their Handling
- Functions to Use with String

Chapter 8: Reading and Writing Files with Python
- Opening a File
- How to Write in a File

Chapter 9: Organizing Files
- Creating a new File for Python
- Handling Binary Files
- Opening Your File
- Seeking Your Files

Chapter 10: Python Tools for Debugging
- Debugging Your IDE
- Options for Debugging
- Preventing the Bugs

Chapter 11: Tuples
Chapter 12: Web Scraping
- Why Use Web Scraping?

What is the process of web scraping?
- Writing the Code

Chapter 13: File, Data, and Statistics Management with Python
- Why is Statistics Important in Python?
- Computational Problems in Statistics
- Managing Your Python Data

Chapter 14: Working with Excel Charts and Spreadsheets

The Terms You Should Know
Getting Started with openpyxl
Chapter 15: Graphics and Images
Chapter 16: To Relax
Python Sprites
Creating a Drawing
Creating a Hangman Game
Creating a Magic 8 Ball
Conclusion

Introduction

Congratulations on purchasing *Python Programming for Beginners*, and thank you for doing so.

The following chapters will discuss everything you need to know when it comes to handling the Python language. There are many coding languages out there, but none offer the ease of use like we will find with Python, while still having all of the power, libraries and extensions needed to get the work done. This guidebook is designed to show you how to work with Python and how to ensure you can begin coding like a professional in no time.

In this guidebook, we are going to spend some time learning more about how the Python language works. We will explore what the Python language can do, some of the ways we can apply this language, and even some simple steps that will ensure Python is downloaded on your chosen operating system. Then, we'll jump on in and learn a few of the basics of coding in this language.

There are a lot of different parts that go into a Python language project. We are going to take the time to explore how this works, and what we can do with some of our own codings as well. We will look at some of the basic parts of the code, and then move on to more complex things, such as how to work with strings, dictionaries, tuples, lists, and so much more.

Don't worry, though! We will go through these step by step, helping you to become a professional in programming without having to worry about it being too difficult or frustrating to handle.

Along the way, we are going to take the time to look over the Python codes that you will use. There are a lot of different codes that work well with Python and learning how to use them and seeing them in play can make a difference in how well we understand this kind of language overall. We will learn how to create some of our own projects as well, going through the codes that are there to ensure we can become master coders in no time at all.

When you are ready to learn more about the Python coding language and ready to make this your own, going from a beginner in programming to a

professional in no time, then make sure to check out this guidebook to help you get started with that goal.

There are plenty of books on this subject on the market, thanks again for choosing this one! Every effort was made to ensure it is full of as much useful information as possible. Please enjoy it!

Chapter 1: Presentation of Python

The Python programming language is one of many different types of coding languages. Some are suited best for helping out with websites. Some are better to help with gaming or with specific projects you want to handle. But when it comes to finding a great general-purpose language which can handle a lot of different tasks all at once, the Python coding language is the one for you.

There are a lot of different benefits to working with the Python language. Python is easy enough for a beginner to learn how to work with it. It has a lot of power behind it, and there is a community of programmers and developers who work with this language who can help you find the answers you are looking for. These are just some of the benefits we get to enjoy with the Python language, and part of the reason to get started with this language as soon as possible!

The Python programming language is a great general-purpose language that can take care of all your computing and programming needs. It is also freely available and can make solving some of the bigger computer programs you have as easy as writing out some of the thoughts you have about that solution. You can write out the code once, and then, it can run on almost any kind of program you like without needing to change up the program at all.

How is Python Used?

Python is one of the best general purpose programming languages and is can be used on any of the modern operating systems you may have on your system. Python has the capabilities of processing images, numbers, text, scientific data, and a lot of other things you would like to save and use on your computer.

Python may seem like a simple coding language to work with, but it has a lot of the power and more that you are looking for when it is time to start with programming. In fact, many major businesses, including YouTube, Google, and more, already use this coding language to help them complete complex

tasks.

Python is also known as a type of interpreted language. This means it is not going to be converted into code readable by the computer before the program is run. Instead, this is only going to happen at runtime. Python and other programming languages have changed the meaning of this kind of coding and have ensured it is an accepted and widely used coding method for many of the projects you would like to handle.

There are a lot of different tasks the Python language can help you complete. Some of the different options you can work with include:

1. Programming any of the CGI you need on your Web applications.
2. Learning how to build up your own RSS reader
3. Working with a variety of files.
4. Creating a calendar with the help of HTML
5. Being able to read from and write in MySQL
6. Being able to read from and write to PostgreSQL

The Benefits of Working with Python

When it comes to working with the Python language, you will find there are a lot of benefits with this kind of coding language. It can help you complete almost any kind of coding process you would like and yet still have some of the ease of use you are looking for. Let's take a quick look at some of the benefits that come with this kind of coding language below:

1. Beginners can learn it quickly. If you have always wanted to work with a coding language, but you have been worried about how much work it is going to take, or that it will be too hard for you to handle, then Python is your best option. It is simple to use and was designed with the beginner in mind.
2. It has a lot of power to enjoy. Even though Python is easy enough for a beginner to learn how to use, that doesn't mean you are going to be limited to the power you can get with some of your codings. You will find that the Python language has the all the power and more needed to get so many projects done.
3. It also works with other coding languages. When we get to work

on data science and machine learning, you will find this is really important. There are some projects where you will need to combine Python with another language, and it is easier to do than you may think!

4. It is perfect for simple projects all the way up to more complex options like machine learning and data analysis. This will help you complete any project you would like.
5. There are a lot of extensions and libraries that come with the Python language, which makes it the best option for you to choose for all your projects. There are a lot of libraries you can add to Python to make sure it has the capabilities you need.
6. There is a large community that comes with Python. This community can answer your questions, show you some of the different codes you can work with, and more. As a beginner, it is always a great idea to work with some of these community members to ensure you are learning as much as possible about Python.

When it comes to handling many of the codes and more that you would like in your business or on other projects, nothing is going to be better than working with the Python language. In this guidebook, we will spend some time exploring the different aspects of the Python language, and some of the different things you can do with this coding language as well.

Chapter 2: The Language of Python

Many companies work with Python and use it to complete a variety of tasks. You don't always hear about the uses of Python because many companies are going to be reserved about letting go of their secrets. But there is not a lot that Python can't help out with and your company can really benefit from using this for their needs.

As a general-purpose programming language, Python can be used for a wide variety of things. Python can be used for both large and small projects, as well as projects that are offline and online. The best options for utilizing Python will include things like data analysis, simple scripting, and Web development. Some of the examples of what Python can help you and your company out with include:

1. Web development: You can use Python to create a Web application, no matter how complex they are. There are a lot of excellent frameworks for Python, including Flask, Django, and Pyramid.
2. Data analysis: Python is one of the biggest choices for data scientists. It has grown in popularity in this field, often because of all the great libraries that come with it, including Pandas and NumPy and even some of the data visualization libraries like Seaborn and Matplotlib.
3. Machine Learning: Machine learning is growing strong for many businesses, and learning how to use will help companies get ahead. There are many great algorithms of machine learning that work with Python, and help us to learn more about our customers and thereby stand out from the competition.
4. Computer vision: You can do a lot of interesting things with computer vision, including color detection and face detection while using OpenCV and Python.
5. Game Development: The module Pygame, which is run with Python, can help us create a video game. These applications work well on an Android device.
6. Web scraping: If you would like to grab some data off a website, but the site does not have the API to expose this data, Python can

step up and do some of the scrapings you are looking for.

7. Writing scripts: If you are doing something manually and you want to make sure the repetitive stuff can be automated, such as emails, it is easier to do with Python.
8. Browser automation: There are some neat things Python can do in this part of the process. It can help you to automatically open your browser and then post the Facebook statuses you want without having to do it manually. Selenium with Python is the best option for doing this.
9. GUI Development: This can help you build your own GUI application, or desktop app, with several modules of Python to support it, including PyQT and Tkinter.
10. Rapid Prototyping: Python has a lot of libraries, and this allows it to handle pretty much every task you would like. You can use it to help you build up your own prototype, even though this is often with lower-performance and less powerful than the full model. Python can be a great tool to validate your ideas or products, whether you are a startup or an established company.

Chapter summary

Python is one of the best languages out there for many businesses to use. Learning how to make this coding language work for your needs and how to ensure you can utilize it to its full potential is going to be a big challenge, but so worth it in the long run. With the ease of use and the power that comes with Python, it is no wonder so many companies want to jump on board and use this coding language for all of their programming needs.

Chapter 3: First Step with Python

Now that we know a bit more about the Python language and what we can do with it, it is time for us to take a look at some of the steps you can take to install this language on your system. Python works well with all three of the major operating systems, including Linux, Mac OS X, and Windows. This makes it easier to use and work with, regardless of the kind of system you prefer in the process. Some of the steps you can use to get started with installing Python on your system include:

First Steps with Mac OS X

If you work with a computer with the Mac operating system on it, then you should find a version of Python 2 already on it. The version of Python 2 is going to vary based on how old the computer is, and which operating system you are working with in particular. But there should be some version of Python on there. To determine which Python 2 version is on your computer, you can open up your terminal app, and use the prompt below to help:

python – V

This will show you the version you get by showing you a number that will come up. You can also choose to install Python 3 on this system if you would like, and it isn't required to uninstall the 2.X version on the computer. To check for the 3.X installation, you just need to open up the terminal app and then type in the following prompt:

Python3 – V

For this one, remember the default for this kind of operating system is that Python 3 is not going to be on the computer at all. This means if you would like to work with a newer version of Python, such as Python 3, then you will need to visit www.python.org to download the version of Python you would like. There are a few other resources to download Python, but the website above can get it all done for you and will ensure you have the IDLE, the interpreter, and other tools needed, without any cost to you.

As soon as you start writing out some codes in this language, you will find

that having the Python Shell and IDLE setup is going to be important to getting that code writing down. You should double-check to make sure both of these are on your system, and the best way to check this is below:

- For Python 2.X just type in “Idle”
- For Python 3.X, just type in “idle3”

As we mentioned a bit before, you will find that when it is time to install your Python 3 version, you will need to install IDLE as part of the standard application. And this part of the code is then going to be found in the applications folder. In order to start this program up and get it to work on your desktop, just open up the folder, double-click on the application for the IDLE, and get to work.

First Steps with Windows

In addition to working with the Mac operating system when it is time to handle some of your Python needs, you can also download and install this language on your Windows computer. And since so many people enjoy working with this coding language and like the ease of use and all of the features, it is nice to know Python will work just fine on it as well.

One thing we need to keep in mind when working with the Windows system is that Python is not going to come pre-installed for you. This is because Windows already has its own coding system, so they would not add another one. The good news is that Python will still work just fine. We just don’t get the added benefit of having this already installed and ready to go on the system.

Setting up Python to work well on a Windows computer doesn’t have to be difficult. Some of the steps you can follow to ensure it is installed include:

1. To set this up, you need to visit the official Python download page and download the Windows installer. You can choose to get the latest version of Python 3 or go with another option. By default, the installer is going to provide you with the 32-bit version of Python, but you can choose to switch this to the 64-bit version if you wish. The 32-bit is often best to make sure there aren’t any compatibility issues with the older packages, but you can

experiment if you wish.

2. Now, right-click on the installer and select "Run as Administrator." There are going to be two options to choose from. You will want to pick "Customize Installation."
3. On the following screen, make sure all of the boxes under "Optional Features" are clicked and then click to move on.
4. While under Advanced Options," you should pick the location where you want Python to be installed. Click on Install. Give it some time to finish and then close the installer.
5. Next, set the PATH variable for the system so it includes directories that will include packages and other components you will need later. To do this, use the following instructions:

    a. Open up the Control Panel. Do this by clicking onto the taskbar and typing in Control Panel. Click on the icon.
    b. Inside the Control Panel, search for Environment. Then click on Edit the System Environment Variables. From here, you can click onto the button for Environment Variables.
    c. Go to the section for User Variables. You can either edit the PATH variable that is there, or you can create one.
    d. If there isn't a variable for PATH on the system, then create one by clicking onto New. Make a name for the PATH variable and add it to the directories you want. Click onto close all the control Panel dialogs and move on.

6. Now you can open up your command prompt. Do this by clicking on Start Menu, then Windows System, and then Command Prompt. Type in "python." This is going to load up the Python interpreter for you.

From here, the program is going to be ready for us to use on our Windows system. You can choose to open up all of the different parts and check them out. Look over the features that are present, and then get started on your own coding journey!

First Steps with Linux

We also have the benefit of working with the Python coding language on a

Linux operating system. While the Linux system may not be used quite as widely as we are going to see with Windows and Mac, there are still a lot of benefits that come with using Python on this system, and when it comes to programming and doing other tasks, Linux is a great choice to go with.

When you want to work with the Linux operating system, the first thing we need to check for is whether or not there is already a version of Python on your system. This is easy enough to accomplish, and you just need to work with the following code to help you get it done:

```
$ python3 - - version
```

If you are on Ubuntu 16.10 or newer, then it is a simple process to install Python 3.6. You just need to use the following commands:

```
$ sudo apt-get update
$ sudo apt-get install Python3.6
```

If you are relying on an older version of Ubuntu or another version, then you may want to work with the deadsnakes PPA, or another tool, to help you download the Python 3.6 version. The code you need to do this includes:

```
$ sudo apt-get install software-properties-common
$ sudo add-apt repository ppa:deadsnakes/ppa
# sudo apt-get update
$ sudo apt-get install python3.6
```

The good news here is if you are working with other distributions of Linux, it is likely you already have Python 3 installed on the system. If not, you can use the distribution's package manager. And if the package of Python 3 is not recent enough, or not the right one for you, you can go through and use these same steps to help install a more recent version of Python on your computer as needed.

Chapter Summary

Python is unique in that it works with every operating system you would like to use. This offers you a lot of freedom and will help you to get some programming done, without having to change up your computer or any of the

systems on it. This can be nice because you can use any operating system you are already comfortable with, whether it is Mac, Windows, or a Linux system.

Chapter 4: The Basics of Python

One of the best things you can to do get familiar with the Python language is to make sure you actually try it out a few times. The more you practice some of the coding you want to do in this language, the more familiar you can become with it, and the easier it is to work on some of the more complicated stuff later on. That is why we will spend some time in this chapter looking at the basics of programming in Python and how you can interact with your IDE and your compiler to get the work done.

In this chapter, we are going to focus on writing some of our first codes. We will keep it simple for this one, and learn one of the most basic codes, the Hello, World! The good news is there are a lot of different parts we can put together with this code to get the best results. We just need to learn the basics before we move on to something a bit more complicated as we move through this guidebook.

To work with the Hello World! program, you can open up your command line text editor. This should have come with the version of Python that you downloaded, so open this up and create a new file. Inside, write out the following line:

$ nano hello.py

Once you get this text file to open up in your terminal window, you can then type out the program by writing out the following line:

print("Hello, World!")

Now, let's break down the different parts of this code. The print() is a function responsible for telling the computer to perform a specific action. We know this is a function because it has parentheses. This function will tell the compiler to display as our output whatever we write inside those parentheses. By default, this is going to output to the current terminal window.

Some functions, such as the print() function, will be built into the Python language by default. You can always use them in any of the programs you decide to create. You can also go through and create some of your own

functions through some other elements.

Now, when you are working on the parentheses of this function, we did add in the term of “Hello, World!” to it. This is going to be inside of some quotation marks to make it work. Any characters in that quotation will be the string we work with. Once we have written this down for our program, it is time to push the X key to exit out of the code. When there is a prompt that shows up, you can press y and then exit back to the shell for the next steps.

Running Your Program

Now that you have written out this program, you can run it in your program. We are going to work with the python3 command along with the name we gave the program file. To run the program, just write out the following line in your command prompt:

$ python3 hello.py

When you type this out into the command line, the terminal is going to give you the following output:

Hello, World!

Let’s take a closer look at what happened in this program. Python went through and executed the line that said print(“Hello, World!”) by calling on the print() function like we talked about before. The string value was then passed over to the function.

For this example, the Hello, World part of the code is going to be called the argument because the value is going to be passed over to be with the function. The quotes we have put on either side of the statement will not show up on the screen here because they are just there to tell the Python compiler the function is going to contain a string. These quotation marks are just going to mark where the string starts and ends, rather than being an important part of the string.

Parts of the Code to Know

Now that we have had a chance to take a look at the Python code and how we can work with it, it is time to actually spend our time learning some of the

basic parts that come with this kind of code. Python coding is not meant to be hard, but knowing a few parts and how they come together can make a big difference in how well you can code overall.

As we get through this guidebook, you will be able to catch onto a lot of the different parts of the code that is important and see how Python works and how easy it is to use. But we are going to take a look at some of the basics that come with writing the codes you would like in Python, and will ensure you can get it all to fit together.

You will notice there are many parts in the Python code. First up is the keywords. These are the commands you give to the compiler so it knows how to behave and what you want it to do. These are reserved; if you put them in the wrong places in the code, the compiler will not work.

As you look through some of the codes, you will notice the # symbol shows up on a regular basis. This is how we write out comments in the Python language. If you want to name a part of the code, you want to give some information, or leave something else in the code, without it messing with the code, you just add the comment, or the # symbol before it.

We also need to know about the variables. These variables are basically just spots in the memory of our computer. We use the variables to reserve them. Then, once we assign a value to the variable in the code, that value will be placed in that part of the memory so the program can call it up.

You can call the variable anything you want, but keep in mind that capitalization matters. Python sees the words python, Python, and PYTHON as different variables, so keep that in mind as you work through this guidebook.

For the Python code, you may notice that many of the codes have spaces and indentations in them. Unlike some of the other coding languages, indentation is going to matter with this one. You need to watch where those are in the codes we provide, and in some of the ones you work on for yourself.

Chapter Summary

As we can see, writing out the codes we would like to use in the Python

language is pretty simple. We went through the simple Hello, World! code above, and it shows us how easy even some of the most basic codes can be, even though they do include some of the more advanced features along the way. Take some time to practice working with the basics of the code above so you are ready for some of the more advanced stuff we are going to progress through in this guidebook.

Chapter 5: Python Programmable Calculator

Now that we have a better understanding of how the Python code works and some of the basics that go with it, let's take some time to explore how to work with our Python code. The first thing we are going to look at is how to program a simple calculator. We are going to look at just making the calculator, without all of the additions you might find with some more advanced calculators. But this is still a great way for us to learn how to work with Python and how to try out some of your coding skills.

In the example we will look at, we are going to learn how to create a simple calculator, one that can add, subtract, multiply, and divide. The action the calculator is going to take will depend on what the user adds as their input. This calculator will be functional, and it will be able to handle a lot of the adding and subtracting, and so on, that you would like to do.

With that in mind, the calculator we are going to make is one that will be done when we work with functions. Functions are a basic idea that comes with Python that can help you organize your code and will ensure the right parts of your code show up at the right times.

The coding you need to use to get started with making your own Python programmable calculator includes:

```
# Python program for simple calculator

# Function to add two numbers
def add(num1, num2):
    return num1 + num2

# Function to subtract two numbers
def subtract(num1, num2):
    return num1 - num2

# Function to multiply two numbers
def multiply(num1, num2):
```

```
    return num1 * num2

# Function to divide two numbers
def divide(num1, num2):
    return num1 / num2

print("Please select operation -\n"
        "1. Add\n"
        "2. Subtract\n"
        "3. Multiply\n"
        "4. Divide\n"



# Take input from the user
# Python program for simple calculator

# Function to add two numbers
def add(num1, num2):
    return num1 + num2

# Function to subtract two numbers
def subtract(num1, num2):
   return num1 - num2

# Function to multiply two numbers
def multiply(num1, num2):
    return num1 * num2

# Function to divide two numbers
def divide(num1, num2):
    return num1 / num2

print("Please select operation -\n" \
        "1. Add\n" \
```

```
        "2. Subtract\n" \
        "3. Multiply\n" \
        "4. Divide\n")


# Take input from the user
select = input("Select operations form 1, 2, 3, 4 :")

number_1 = int(input("Enter first number: "))
number_2 = int(input("Enter second number: "))

if select == '1':
    print(number_1, "+", number_2, "=",
                    add(number_1, number_2))

elif select == '2':
    print(number_1, "-", number_2, "=",
                    subtract(number_1, number_2))

elif select == '3':
    print(number_1, "*", number_2, "=",
                    multiply(number_1, number_2))

elif select == '4':
    print(number_1, "/", number_2, "=",
                    divide(number_1, number_2))
else:
    print("Invalid input")


number_1 = int(input("Enter first number:"))
number_2 = int(input("Enter second number: "))

if select == '1':
```

```
    print(number_1, "+", number_2, "=",
                    add(number_1, number_2))

elif select == '2':
    print(number_1, "-", number_2, "=",
                    subtract(number_1, number_2))

elif select == '3':
   print(number_1, "*", number_2, "=",
                    multiply(number_1, number_2))

elif select == '4':
    print(number_1, "/", number_2, "=",
                    divide(number_1, number_2))
else:
    print("Invalid input")
```

Chapter Summary

This is a simple calculator you can work with that uses the Python language. This helps to ensure we can do some of the basic mathematical equations that we would like and is simple enough for even a beginner to get used to working with. If you are interested in using this for your own needs, you simply need to plug the code above into your compiler, and then give it a try!

Chapter 6: Dictionaries and Python Data Structuring

Another topic we need to take a look at is how to work with the dictionaries in our Python language and why these dictionaries are going to be so important to some of the work we try to do in this language. One thing you will quickly realize when you are working with the Python language is that the dictionaries are going to work similarly to a regular dictionary you can hold in your hands.

Python can offer us quite a few different structures of data that can hold onto our information, and out of these, the dictionary is going to be one of the simplest and most useful. While many things that show up in Python are going to be seen as iterables, not all of these are going to be sequences and the dictionary in Python is going to fall into this kind of category. In this part of the guidebook, we are going to spend some time looking more at the Python dictionary and how it works, along with some of the most common applications of working with this language.

What is a Python Dictionary?

The first thing we need to look at is what the Python dictionary is all about. Being able to get clean data you can act on is one of the key challenges that happen in some of the data analysis and many other common Python codes you try to accomplish. You can't build and fit in some of the models to the data if that data is not usable in the first place. A Python dictionary is going to make it easier to read and then change the data, which is going to make it more actionable for some of the work you would like to do with predictive modeling.

To make things simple, a Python dictionary is going to be an unordered collection of data values. While only some of the other types of data that shows up in this language can only hold onto one value as their element, the dictionary can hold a key or a value pair. This means the dictionary in Python is optimized in a manner that allows it to access values at any time.

While each of the keys in this process are separated out by a comma in this

kind of language, each of the key-value pairs are separated out by a colon when you use them. In addition, while the keys of the dictionary have to be immutable and unique, meaning there are parts like integers, strings, and tuples, you can repeat them more than once if you would like.

While there are several methods for using the Python dictionary when you need it, there are a few basic operations we need to master before we can do this. We have to walk through a few of those different parts in the sections to come, as we will look more at how to create some of our own Python dictionaries along the way.

Creating Our Python Dictionary

Now, it is time for us to look at some of the steps needed to create our own Python dictionaries. To create one of these dictionaries, you need to put the items, which are going to have a key and a corresponding value and will be expressed as the key value inside some curly brackets. And then each of the items in these brackets has to be separated from one another with a comma.

As we discussed earlier, values can be repeated and by any type that you like. Keys, on the other hand, are going to be immutable and unique. There is also a function that is built-in when you work with the Python language, called dict(), that you can work with to create some of your dictionaries when you are ready to go.

Another thing you can work on is accessing the items within your Python dictionary. Accessing these items is simple. You just need to take the time to put the key name of the item within some square brackets to tell the compiler what you would like to do. Make sure you are using the right brackets though because the keys are going to be unique and will not repeat for you.

To help you get the value of your model key, you would use the code “x = thisdict[“mode”]. It is also possible to use another method of Python dictionaries, which is the function of getting () to help you access the idea when you would like. You would just need to use the following code to get it done “x = thisdict.get(“model”). Either of the two is going to be correct; you just need to make sure you utilize them in the proper manner.

There are also times when you will need to change up the values found in one of your Python dictionaries. To change up the value of an item, you once again need to refer to the key name. So, if you have the year as 1964 and you would like to change it to 2015, you can type in a code like “thisdict[“year”] = 2015.

The loops are another fun part of working with some of our Python codings. You can use a loop function to make it easier for the program to loop through the dictionary in Python. The default for this one is to return the value while it is looping through the dictionary. And the trick here is that you will have the keys doing the work as well.

There are other methods you can choose from as well to help you return these values. You can use something simple to do all of this. To print the key names, for example, you would just type in “print(x). But to bring out the values of the dictionary, making sure they come out one by one, you would use the code “print(thisdict[x]).

Methods for Python Dictionaries

Now that we know a bit more about the dictionaries and how they work, it is important to take a look at some of the most common methods used for this kind of process. You will find that these are going to show us a lot of the different tasks we can do with the Python dictionaries to get the best results. Some of the different methods we can use with the Python dictionary includes:

1. Clear(): This is going to remove all of the items out of the dictionary you created.
2. Copy(): This method is going to provide us with a copy of our Python dictionary.
3. Fromkeys(): This one is going to help return a different directory that has only the pairs of key values we have been able to specify so far.
4. Get(): This is going to return the value of all the keys we mention.
5. Items(): This method is going to return to us a tuple for every key-value pair in that particular dictionary.
6. Keys(): This one is going to return a list of all the Python

dictionary keys in that dictionary right now.

7. Pop(): This is going to make sure that only the key you mentioned will be removed.
8. Popitm(): This is going to be the method that can delete just the items that were the most recently added.
9. Update(): This method is going to update the dictionary with certain key-value pairs you mention.
10. Value(): This method is going to work because it returns the values of all the items on your list.

Before we move on, it may be helpful to see how a dictionary in Python can be created. It is a relatively simple process, so let's dive right in with the code below to help us out:

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

The Advantages of Using the Dictionary

There are a lot of benefits we can see when it comes to working with the Python dictionary. First, it is going to ensure you get more readability in the code you write. Writing out the Python dictionary keys, along with some of the values that go with them, is going to add in some documentation to the code you are writing. If you can make the code more streamlined, it will be easier for you to debug this code. Ultimately, the analysis you are doing with the help of the dictionary will be done faster than before and the models are going to be fitted more efficiently.

Apart from some of the readability we discussed, there will be the question of speed. You can use the Python dictionary to look up a key very quickly. The speed of a task like looking the keys up will be measured because we can look up how many operations it will take to finish this. Looking up a key will be done in constant time vis-a-vis looking up an item in a large list will be

done with linear time in mind.

To look up an item on a very large list, the computer will take some time to look through each and every item found on your list. If every item has been assigned with a pair of key values, then you only need to look for the key. This helps to speed up the process and can give you the information you need in no time. A Python dictionary is basically an implementation of the hash table. This means it will provide you with all of the benefits of a hash table, which includes checks of memberships and speedy tasks like looking up keys when you need them.

Of course, while the Python dictionary is going to be a good one to work with, and it can be considered one of the most useful tools for you to use, especially with data analysis and data cleaning, there are some downsides. For example, these dictionaries are going to be unordered. In cases where the order of your data is very important to how well the code and program work, the dictionary would not be a good choice to go with.

And finally, the Python dictionary, while helpful, is going to take up more space on the memory of your computer than the data structure. The amount of space taken up by your dictionary will increase when there are a lot of keys for the dictionary. Depending on how much memory you have, this may or may not be a big deal to you.

Chapter Summary

There are a lot of advantages to working with the Python dictionary. It is one of the best ways for you to keep information in one place, and then pull them out with the right key values to get things done. We took a look at some of the different things we can do in order to get started with these dictionaries, how to change things up within the dictionary, and even some of the most common methods you are likely to use when handling some of these dictionaries as well.

Chapter 7: Strings and Their Handling

With some of the codes we write in Python, we will work with a topic known as strings. This is a type of data that can help us to learn more about how to get the code to work the way we would like. We are will spend some time in this chapter looking at what the strings are all about, the operators that come with these strings, and even the functions of strings.

To start, we need to know that a string is going to be just a series of text characters written out in your code. There are a number of operators you can focus on when it comes to strings in Python. An operator, in this case, is simply a symbol that can perform a specific operation in the code you are working with. In this scenario, we are talking about how the operator is going to be what we can do with the string. There are a number of things we can do with these strings to help us get the best results, but some of the options include:

1. No in operator: This is the operator we can use when to want to do the opposite of the operator that comes next. This one can search out a specified character in the string. But if the operator can't find that character in the string, then the return you get will be True. If the character is found in your string, then the return you will get will be False.
2. In operator: This is the operator responsible for searching out a specified character in your target string. If the character is found somewhere in the string, then the return will be True. But if the program can't find the answer for you, then False will be your return.
3. Concatenation Operator: This is going to be the operator helpful any time you want to do some work with the concatenate strings you have.
4. Repetition operator: This is the operator you can use when it is time to make more than one copy of the string you are working with. You get to choose the number of times this string needs to repeat for your code.
5. Slice operator: This is the operator that can take a look through your string and then will retrieve out the character you would like

from there. Any time you choose to use this one, always remember that zero is going to be the placeholder for the first character in that string.

6. Range slice operator: This is the operator we can use when we would like to retrieve a range of characters from your index, rather than working with just one character. If you just want to showcase one word or even just one part of that string, then this is the operator you are going to work with.

Functions to Use with String

The next thing we need to spend a bit of time focusing on here is all of the neat things that we can do with the help of the functions. There are a lot of functions you can utilize when it is time to work with a Python string. Some of the most common options that are going to be used for a lot of the coding you have includes:

- Capitalize(): This one is going to take the first letter of the string and capitalize it for you.
- Center(width, char): This is going to return to you a string that is at least the specified width, and then it will be created by padding the string with the character.
- Count(str): This is going to return the number of times a particular string is contained in another string.
- Find(str): This is going to return the index number of the substring in the string.
- Isalpha(): This is going to check if all the characters of a string are alphabetic characters.
- Isdigit(): This part is going to check whether the string just contains numbers or digits or if there is a mixture.
- Islower): This function is going to take a look to see if the string you are checking has all lower case characters.
- Len(): This is going to let you know the length of the string
- Isupper(): This one is going to check to see if all the characters in the string are upper case.
- Lower(): This will give you a return that has all the string in lower case letters.

- Replace(): This is going to take the string that you have and replace it with a new string
- Upper(): This is going to return the string in upper case.
- Split(): This is going to split up the string based on the split character.

As we can see, there is a lot you can do when it comes to working with the strings found in Python, and being able to put these to good use and see how they work together can make a big difference in some of the coding you can accomplish. Make sure to learn about some of the functions that show up in the string and how they all work, and you will be able to easily insert them into any type of code you would like.

Chapter Summary

Working with these strings can help add another level to some of the programmings you would like to accomplish. Learning the right functions that will help to get this done can be even more important as well. These are considered a bit more advanced than maybe some of the other stuff we have discussed in this guidebook, but taking the time to learn how to get it done can really add another level to the coding you are trying to accomplish.

Chapter 8: Reading and Writing Files with Python

Working with files is going to be an important part of working with a Python code. A file is a named location on a disk meant to store related information. It will be used in order to permanently store some of the data you want to work within the non-volatile memory or the hard disk. Since our RAM, or random access memory, can be pretty volatile, it is going to lose its data when you turn the computer off. The files are what we want to work with because it ensures we can find our information in the future.

When we want to either write to or read from a file, we first need to go through the process of opening that file to see how we can make it work for our needs. When we are all done, we then need to close this up, so the resources we use for this are tied up with the file we are freeing. In this chapter, we will take a look at how to open a file, how to write on the file, and how to read the file.

Opening a File

Python is set up so it has a built-in function to help you open up the file you would like. And this function is called open(). This function will return the file object or the handle we want, as it is used to help read or make modifications for the file in the way you would like. We can then specify the mode while opening up our file. In mode, we need to make a specification of whether we want to work with “r” for reading or “w” for writing, or even “a” to append the file. We also need to specify if we want to get that file to open up in the binary mode or the text mode.

The default for this one is that your file is going to show up in text mode. When we are in this mode, we will end up with some strings when we read from the file. But you also have some choices when it comes to how you open the file, and you have the option to open it in binary mode. Choosing binary mode is going to return bytes to us, and can help when you are dealing with a file that doesn’t contain text.

To help out with this one, we need to take a look at some of the different Python file modes available. Some of the options that can help you to open up the file you want in the correct order and method desired includes:

1. r: This will help us to open up a file to read.
2. w: This will open up a file so we can write in it. This can help us create a new file if one doesn't exist, or you can use it to truncate the file if it does exist.
3. x: This will help us to open up a file for exclusive creation. If the file is already existent, the operation is going to fail.
4. a: This is going to open up the file to append it. This allows you to add some new information at the end of the file without having to truncate it. This can also help you to create a new file if it does not exist at all.
5. t: This is going to open up the file in text mode.
6. b: This will open up the file in binary mode.
7. +: This opens up a file for updating, whether you are reading it or writing it.

How to Write in a File

The next task we need to look at is writing out a file with the help of Python. In order to write into one of our files, we have a few options, including write, exclusive creation, or append mode. Remember to be careful with the write mode, because if you use it, it will end up overwriting all of the work you have put into the file already. All of the previous data, with this option, is going to be erased, so make sure that is actually the thing you want to do.

Writing out a string, or even a sequence of bytes if you are working with your binary files, is done with the help of the write() method. This is a method that can provide us with the number of characters found in the file. The code you can use to see this includes:

```
with open("test.txt",'w',encoding = 'utf-8') as f:
    f.write("my first file\n")
    f.write("This file\n\n")
    f.write("contains three lines\n")
```

This program is going to create a new file named "testtxt" if you don't already have a file with this name on your computer. If you do have this file already, then the code above is going to overwrite it. We need to include some of the newline characters on our own to help us distinguish some of the

different lines.

Chapter Summary

Handling some of the files available with your code can be important. This ensures we can write a new file, make changes to a file, append to a file, and so much more. But you have to ensure you are picking the right method to work with, or you may end up with some issues along the way with your programming. Following the codes above, though, can help you get so much done with your files in Python, and you can see just how easy all of this can be when put together.

Chapter 9: Organizing Files

When you are writing out some of the different codes you want to get done in the Python language, there will be times when you need to organize some of your files. These files are important to what you are doing in your language, but if we don't take the time to organize them and give them the proper care and attention they need, we are going to end up with a big mess when it is all said and done.

Any of the data you use in Python is going to be saved as a file that goes on your disk. Sometimes, you will be able to reuse the code over and over, such as what we will do with an inheritance, but we have to take the right steps for this, and often, most people are happier when they can just turn that data into a file. There are a number of things we can do when it is time to work in the file mode with Python. To help us keep this as easy to understand as possible, we can look at an example.

Think about when you use a Word document. You will open it up, write something into it, and then save the information so it doesn't get lost. The files you are working with on Python are going to work in a manner that is similar. However, instead of saving the pages, you are going to save the code you are writing in a place where you can find it and where the compiler will be able to bring it out later. There are a few operations we can focus on to make this happen and that includes writing out some code in a file we already created, seeking out or moving a file, creating a brand-new file, and closing up and saving your file. Let's take a look at how each of these will work for the coding you want to do.

Creating a new File for Python

The first thing we are going to focus on when it is time to work with the Python files is how to create one of your own files. If you want to make up a new file, and then put some code inside of it, you first need to make sure the IDLE and compiler are up and running. You also have some decisions to make, such as figuring out which mode you want to use to write out the code. Keep in mind there are three main options for you to choose, including append(a), write(w), and mode(x).

Any time you want to make some changes to the current file that is opened, you can use the (w), or write, option because this is often the easiest one to work with. Any time you are trying to open up a file and then write a new string in that file, you will work with binary files and will need the write() method. This is going to work well because it ensures you will get the right characters returned at the end.

The write() function is really easy to use and allows you to make as many changes to the file as you wish. You can add some new information to the file or you can make some changes to one you opened up. If you are interested in doing the writing in the code, you can use this example to make things easier:

```
#file handling operations
#writing to a new file hello.txt
f = open('hello.txt', 'w', encoding = 'utf-8')
f.write("Hello Python Developers!")
f.write("Welcome to Python World")
f.flush()
f.close()
```

Before we move on, make sure to open up your compiler and write out this code. This code is basically helping you to create a file and get all the information into the right directory. The default directory is known as the current directory. It is possible to switch out which directory where you want to store the information, but make sure you go through and change that ahead of time or you may have trouble finding it later on.

We have to be careful with what we are doing here though. The specific directory we are in when we write the code is where the file is going to be saved. Sometimes, this may not be the right place where you would like to put the file, and you may decide it is best to move it. If you would like to have the file in a different directory, then you will need to make sure your current directory is the chosen one before you get started.

With the option above we worked on when you go to the right directory where the file is saved, and you open that file up, the message you are going to see is "Hello Python Developers! Welcome to Python World!"

Now that we are at this point, we can see that through here, we have been able to write out a new program. However, there may be times when you need to make some changes, or you will want to overwrite the program you are working with. This helps you to get a new statement to show up on that file. Say that you already wrote out the file above, but now you want to change up the message coming up with it. This is something we can do with this, but we have to make sure the syntax we are working with is changed up a bit. The coding needed to make this happen includes:

```
#file handling operations
#writing to a new file hello.txt
f = open('hello.txt', 'w', encoding = 'utf-8')
f.write("Hello Python Developers!")
f.write("Welcome to Python World")
mylist = ["Apple", "Orange", "Banana"]
#writelines() is used to write multiple lines into the file
f.write(mylist)
f.flush()
f.close()
```

The example above is a good one to use when you want to make a few changes to a file you worked on before because you just need to add in one new line. This example wouldn't need to use that third line because it just has some simple words, but you can add in anything that you want to the program, just use the syntax above and change it up for what you need.

Handling Binary Files

While we are on the topic of handling files, we are going to take a little detour and look at some of the steps you need to take in order to put your data into a binary file instead of the other files we were looking at before. This is easier to do than you may think. With this one, you can take all of the data you are working with and then can turn it into a sound or an image file, rather than having it written out as the text file from before.

You can take any of the text you are working on in Python and change it to be a binary file. It doesn't matter if the data was originally a text file, picture file, or a sound file when it started. The most important thing we need to consider here is you need to supply the data inside of the object to help make

it easier to expose as a byte later on in the code. To make the data you have into a binary file, you just need to use the following code:

```
# write binary data to a file
# writing the file hello.dat write binary mode
F = open('hello.dat', 'wb')
# writing as byte strings
f.write(b"I am writing data in binary file!/n")
f.write(b"Let's write another list/n")
f.close()
```

This will help you to get the data changed over to a binary file simply by turning it over to the image or sound file.

Opening Your File

After taking the time to create and save the file you want to work with, it is time to take it to the next step and actually open up the file. Maybe you created a file and got it all set up, but then you have to come back to it later. After saving, you want to make sure you can open up that file and get it to work for your needs.

In the two examples we discussed above, we talked about how you can write out words into the file you created and you can change up that text to make it a binary function so you can bring it up later. Now, we are going to move over to opening up that file. Once you open up that file, it is easier to use it again. The code you can use to open up a file you created and saved includes the following syntax:

```
# read binary data to a file
#writing the file hello.dat write append binary mode

with open("hello.dat", 'rb') as f:
        data = f.read()
        text = data.decode('utf-8'(
print(text)
```

The output you would get from putting this into the system would be like the

following:

Hello, world!
This is a demo using
This file contains three lines
Hello world
This is a demo using

This file contains three lines.

Seeking Your Files

The final thing we are going to take a look at here is how we can seek some of the files we are working with. Working with these files in Python is a great way to learn more about the Python coding language. And since we have already taken the time to write out a file, and even save or open it, it is time to take it a bit further and learn how to seek out and find the file we want to work with.

For this one, we want to make sure we can move the file to another location, such as when we decide a new directory is the best one to use, or we find out we chose the wrong directory in the first place. Perhaps, you are working on your file and you find out that something is not matching up the way you would like, such as with the misspelling of a name or you got the identifier to show up wrong. Sometimes, it is just because you placed the file in the wrong directory. This is where the to seek function is going to come in and help you to get the work fixed.

You will be able to go through and change up the position where the file is so it goes to the right spot, or even so it becomes a lot easier for it to find. You just need to tell your code where you would like to find the code and then make the needed changes.

Working with files in the Python language will help you out a lot when you are trying to get things in order inside your code when you want to make changes to what you wrote, and so much more. Try out a few of the codes inside this guidebook and see just how easy it can be to create a new file, make changes, and get the files to work the way you would like.

Chapter Summary

Working with files is a big part of what we do while coding in Python. But these sometimes seem confusing to handle. That is why this chapter took some time to explore how to do some of the various actions needed with these files, including creating a file, changing it to a binary file, working with seeking it or opening it, and so much more. This is meant to ensure you can properly handle all of the files that come up in your code writing journey.

Chapter 10: Python Tools for Debugging

When you are working on some of the codes you would like to write in Python, there are going to be some times when things go wrong. The code may not work the way you would like, and little issues and bugs can become a problem for you to handle. It is important to know more about which tools you can use to not only prevent these bugs along the way, but to make sure you can fix them if one does show up in the coding you try to do with Python.

The neat thing about working with Python is it comes with a lot of different debugging tools. This is going to make it easier to check up on the code you are doing and to ensure you can get it to work the way you would like. There are tools that allow you to work from the command line, as well as from the IDE, and even some analysis tools that are going to come in and help you prevent some of these bugs in the first place.

Debugging Your IDE

The IDE, or Integrated Development Environment, is a great place to start when it is time to start on your big project in Python. This is also an important place where we are going to work on the debugging process because it has direct interaction with some of the coding you do in Python.

Now, you will quickly notice that the debugging tools you can use for your IDE will vary depending on what kind of IDE you would like to use. The good news is they all are going to have the same kind of features to get the work done, including the steps of running the code, setting the breakpoints, and examining the variable.

There are a lot of options when it is time to work with the IDE you would like to use. Whether you choose to go with one that comes with Python to keep things easier and to just download everything at once or you plan to use the specialized version you can find, you will see that it is easier to find a debugging tool for your version just by doing a simple search. Many of them are going to come with a similar kind of workflow so you’re able to pick out the one that is the best for your needs.

Options for Debugging

There are a lot of different debugging tools we can use in order to get started with this process in our program. Learning which ones can handle some of the work we want to do and can ensure our programs work in a smooth manner as we want is going to be important. Some of the different options you can choose when it is time to work on the debugging process in Python include:

1. Pdb: This works on all of the major operating systems. This is going to be seen as the standard library debugger. The neat thing is that it comes with the installations of Python you already have.
2. Pudb: This one is going to work with Mac OS X and Unix. This is a visual console-based and even a full-screen debugger that has been designed as a more comfortable replacement that we can use when compared to the option above.
3. HAP Python Remote Debugger: This is a debugger that works well with the Windows system. It is going to provide us with the ability to debug our system from a remote location.
4. Windpdb and Rpdb2: This one is going to work on Windows, Linux, and Unix: We will see this one as a more advanced Python debugger, with support for some of the smarter breakpoints, embedded debugging, encrypted communication, changes to the namespaces, and multiple threads. It is also faster than some of the other options.
5. Rpdb: This one will work with Unix, Mac OS X, and Windows: This is the predecessor to the option we just talked about above and it is going to improve some of the usability we see with pdbs and adds in some of the support needed to handle the remote debugging. It is also going to come with the ability to debug embedded scripts, post mortem of unhandled exceptions, and the ability to debug more than one thread at a time
6. JpyDbg: This one is going to work with Windows, VMS, Unix, OS/2, and Mac OS X. It is going to be both a JPYTHON and CPYTHON debugging framework that has been changed up so it works as well inside of Jedit as a standard Jedit plugin.
7. DDD: This one will work with the Unix operating system. It is going to be a graphical command-line debugger that will help us to handle some of the debugging we would like to get done in our

code.

8. Pyclewn: This works well with Unix and Windows: This one is going to allow the programmer to use Vim as their front end to get the work done that they would like.
9. Pdb++: This one is going to work with Mac OS X, Windows, and Unix. This is going to be a kind of extension from the pdb module we saw with our standard library and it is meant to be compatible with the predecessor as much as possible. The good news is it is still going to introduce a number of new features to make the experience of debugging as easy as possible.
10. Python-pydebug: And finally, we are going to look at this debugger that works well with Unix, Mac OS X, and Windows. This is a set of debugging decorators that will respect the settings for Django within our coding. It will allow a user the ability to turn PDB into a function, inspect an object, work with a line profiler, and even to disassemble the function if needed.

Preventing the Bugs

Of course, one of the best things you can do for your own coding and your own program is to make sure none of those bugs get into the program in the first place. In the beginning, this is going to be hard and you may worry that your code is never going to work because you get a bug in it. But the more you practice and learn about these debugging tools, the easier it is going to be to avoid some of these bugs and to make sure the coding is going to work the way you would like.

Learning how to prevent some of these bugs, or at least keep them down to a minimum when you are first starting with your coding journey, can be important. So far, we have spent our time talking about some of the reactive tools out there for finding and fixing the bugs that do show up. But what if we were able to prevent these bugs from showing up in the first place?

This is where a number of analysis tools are going to come into play for us. Using these tools will give us a handle on where the code is going. One thing to keep in mind with this though is that your code may not always go to a good place like you want it to. Some of the tools you can use in order to pre-debug your code include:

1. Pylint: This is known as a linter because it can look over the code you have and then make some suggestions on the things you can improve on within that code.
2. Pycodestyle: This is a good tool to use because it will make some recommendations for you and about how you should format the code you want to work with. The idea with this one is that if your code can follow some of the standard conventions, then it is easier to have other Python programmers take a look at it and gain a better understanding in no time.
3. Flake8: This is going to be one of those tools you will want to use all of the time. It is going to come with both of the two features we talked about in one, along with a few other features as well. This makes it easier to have all of the tools you need to check the code and make sure it is working well.

Chapter Summary

Sometimes, there are bugs that can get into our programs and can cause a bit of a mess with getting the program to work the way we want. The good news is that Python will come with a lot of different options we can use to help get the bugs out and to make sure our codes will work the way we want. However, using some tools and being careful with some of the coding we work on can be the best way to handle this because it ensures we can avoid some of these bugs in the first place.

Chapter 11: Tuples

The next topic we need to spend some time looking at in this guidebook is Python Tuple. When we talk about the tuple, we are basically talking about a list in Python. There are a few differences we need to explore before we get started though. The big difference between a list and a tuple is that once we assign an element of the tuple, we are not able to change that element. But when we work with a list, it is possible to change up the elements as you work with them in the code.

When you take a look at a list and a tuple though, they are going to look pretty much the same. It is simply going to be a matter of whether you would like the ability to change up your elements when they are assigned. If you want to be able to make some changes, then you need to work with a list. If you want to keep the elements exactly the same after they have been assigned, then you will want to work with a Tuple.

The main task we are going to take a look at when we want to work with Tuples is how to create one of our own. This is a simple process to work with, and you can create it when you place all of your elements, or items, inside the parentheses that are there, and separate them out by commas. One thing to keep in mind is these parentheses are going to be optional, but it often is considered good practice and makes coding easier if you add them in.

As you work on creating your own tuple, you will find you can add in as many items as you would like. There will usually be at least two since you are working with a list kind of ideas, but you will not have a maximum or minimum you need to worry about. You can even make the items different types, meaning you can add in strings, list, float, and integers to the mix as much as you would like.

Now that we have talked about the tuples and how they fit into some of the codings you are doing with Python; it is time to take a look at some of the coding needed to create your tuple. The coding that helps you to create a tuple in Python will include the following.

```
# Empty tuple
my_tuple = ()
```

```
print(my_tuple)  # Output: ()

# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)  # Output: (1, 2, 3)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)  # Output: (1, "Hello", 3.4)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))

# Output: ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)
```

With this in mind, you will find here are a number of benefits that come with working on a tuple, rather than a list. Since these tuples are going to be similar to lists, and they are going to be used in a similar situation as one another, it is hard to know when one needs to be used, and why a tuple would have some advantages over working with a list. Some of the best advantages of working with the tuple, rather than putting the list into your code will include:

1. The tuple is going to be used for heterogeneous or different types of data, and for homogenous or similar types of data.
2. Since these tuples are going to be considered as immutable, iterating through the tuple is often going to be faster than what we will see with a list. This helps us to get a slight boost in the performance of our code when using tuples.
3. Tuples are going to contain some of the immutable elements that can be used as a key for a dictionary. When you work with a list, this is not something that is possible.
4. If you have some data that doesn't change, implementing it as a tuple is going to help ensure that you can keep it protected.

Chapter Summary

Tuples and lists are going to be very similar, and if you just look at them in

the code, they are going to look identical. But the main difference is whether you would like to be able to change up the list of items after they are assigned or not. With the tuples, once you assign a value there, the value is going to stay the same and you are not going to be able to change it at all. But with a list, you can make some of the changes you need after the assignment. Understanding how the two of these works together can make a difference in the kind of coding you are trying to handle.

Chapter 12: Web Scraping

Imagine that your business has decided to go through and do a data analysis to help them improve customer relations. A huge amount of data should also be pulled out as fast as possible from several websites. Ensure that this process is done immediately. However, how will you possibly do this in a short amount of time without having to do it manually, gathering and searching data one by one on every website?

If the process above sounded tedious, then web scraping may be for you. With this process, we can go through and extract the information we want out of that website. There is a lot of benefit to working with the process of web scraping, and the Python coding language can help make this process as easy and effortless as possible.

Why Use Web Scraping?

For the most part, web scraping is used in order to collect a large amount of information from a website. But the first question you may have about this is why someone would need to collect all of this data from a website to start with. Below are some examples of web scraping applications. Let's all know about them.

1. Email addresses: Almost all companies nowadays use email as a way for them to do their marketing activities. Web scraping can then be used to help collect the emails they need to send out some bulk marketing emails to many potential customers.
2. Price comparison: There are a number of services that work with web scraping, including ParseHub, to gather information from shopping websites found online. This information will be used in comparing the prices of several products.
3. Social media scraping: This is going to be somewhat similar to what we see with email gathering from above. Using this, web scraping will be used to aid us in gathering information from different social media platforms like Facebook and Instagram. This will greatly help in knowing what the next best steps are for the business and figuring out what is currently trending.
4. Research and development: You can use web scraping in order to

help collect a large set of data from websites. Then you can analyze that information and use it to carry out some of the work you need to get done for research and development and for surveys.

5. Job listings: Sometimes, details about various job openings are going to be gathered from a large number of various websites. They can be listed in a single area, making it easier for the user to access them when needed.

Web scraping is simply a method used to automatically extract a huge amount of information from a specific website. The data on the website is going to be unstructured in nature. The unstructured data options can be collected by using web scraping and then transform it into a structured form, thus allowing the information to now be stored.

There will be different ways to scrape a website, including writing out your own code, APIs, and online services. We will take a look in a moment at how you can implement some of the web scrapings you need with Python and some of the code needed to make this happen.

Before you jump in and try to do this process though, realize that some websites are fine with this, and some are not. The robots.txt file for that website can also be looked into to figure out whether a particular website allows for this scraping or not. If the website does not allow scraping, then it is best to find another resource for the work you want to do.

What is the process of web scraping?

When a code is run for web scraping, you will find that a request will be sent over to your mentioned URL. The server, as a reply to the request, is going to send the data and then will allow you a method to read that XML or HTML page. The code then will parse through that page, find the data you need, and then can extract it for you to use as needed.

There are a number of steps we can use when it comes to web scraping. Since the first few are pretty simple, we are going to focus most of our attention on the steps you can use in order to help us to write our codes. The other steps we need to use to work with web scraping include:

1. Find the URL of the website you are interested in scraping.
2. Inspect that webpage to make sure it is the right one for your needs.
3. Find the data you are most interested in scraping and extracting.
4. Write the codes.
5. Run the code so you can actually draw out the information.
6. The data can then be stored in your preferred format until you are ready to complete your data analysis later on.

With these in mind, let's take a look at some of the coding you can do in order to get started with web scraping and making this process work for your needs.

Writing the Code

Now that we know a bit more about the idea of web scraping, it is time for us to look at some of the codings that need to be done. We will start out by creating our own Python file. To do this, we are going to name our file "web-s." Then, we need to open up our terminal, and type in the following code:

gedit web-s .py

Now, we want to go through and write out our code in the file. Before you can do that though, we need to import all of the libraries that are necessary. These can include the following:

```
from selenium import webdriver
from BeautifulSoup import BeautifulSoup
import pandas as pd
```

To configure our web driver so that it works with the Chrome browser (which is the one that we are going to focus on here), we need to make sure the path is set to this. We can make changes based on the browser we choose to use, or just go to this browser and find the website you are interested in. The code we can use for this includes:

```
driver = web driver.Chrome("/usr/lib/chromium-browser/chromedriver")
```

We can then refer to the code below to help us get this URL open and ready

to go for our needs:

```
products=[] #List to store name of the product
prices=[] #List to store price of the product
ratings=[] #List to store rating of the product
driver.get("<a
href="https://www.flipkart.com/laptops/">https://www.flipkart.com/laptops
/</a>~buyback-guarantee-on-laptops-/pr?sid=6bo%2Cb5g&uniq")
```

With the code we have above, we can open up the URL we want to use here. Now, it is time for us to go through the steps needed in order to extract out the data needed from that particular website. As we took some time to mention a bit ago, the data to be extracted will be placed in the tags. In order to make this work, we want to find the tags that have the right class names we need and gather the information from that. The data can then be stored in the variable you have chosen. To get this done, we are going to rely on the following code to help:

```
content = driver.page source

soup = BeautifulSoup(content)

for a in soup.findAll('a attrs={'class': '_31qSD5'}):

name=a.find('div', attrs={'class': '_3wU53n'})

', attrs={'class': '_1vC40E _2rQ-NK'})

', attrs={'class': 'hGSR34 _2beYZw '})

products.append(name.text)

prices. append(price.text)

ratings. append(rating .text)
```

We have been able to get everything set up, so it is time for us to actually run

the code we have been writing and then extract all of the data. The command we will need to use to make this happen is:

python web-s .py

After you have been able to run the previous code and have gotten it to extract the data for you, it is important to take the time to store this data so you can look at it later. Often, businesses will perform data analysis on the data they have and will want to store it all together until they can create the model or the algorithm that will sort through the data and provide them with some of the insights they need.

The format you want to use is going to depend on what your requirements are all about. For this example, we are going to store the data we were able to extract in a CSV format. To make sure this happens, we want to add in the lines below in our code:

```
df = pd.DataFrame({'Product Name':products,'Price':prices,'Rating':ratings})
df.to_csv('products.csv', index=False, encoding='utf-8')
```

To finish this up, we will need to run the whole code again. You have created your own file, which is named products.csv. And you will be able to find all of the data you extracted out of your website on this page to use as you see fit.

Chapter Summary

When you are using Python to handle a data analysis on your business, web scraping can be a great option to focus on. It will help you to gather up a lot of the information you need out of that resource without having to manually read through it and write down the data that you need. With the codes above and a knowledge of what websites you would like to use, you can have the Python language do it all for you.

Chapter 13: File, Data, and Statistics Management with Python

In this chapter, we will spend some time taking a look at the idea of statistics and how we can use this to handle some of the different data we gather and handle. This can be important whether you want to learn about your customers, learn more about your industry and the competition, figure out which products to release to the market that will actually do well, or do some of the other options that come with data analysis.

Statistics will make it easier for us to see some of the basics that come with the data we are working with and will ensure we are better able to handle some of that as well. We are going to take some time to look at how to figure out the mean, median, mode, and standard deviation of any kind of data set we are working with, which can really help us to learn more about our data and figure out how we can use it for our benefits.

Why is Statistics Important in Python?

There is usually a lot of misunderstandings when it comes to the field of statistics, but it is going to play an important factor in our day-to-day lives. Knowledge from the real world will be extracted when the statistics are correct. However, it can cause misleading and harm when it is used in the wrong way. In order to help us know this information, we should fully understand what statistics is all about and what the various measures actually mean.

To help us see a bit about how statistics work, we are going to use a data set from Kaggle's Wine Review set of data. For the help of this part of the guidebook, we are going to pretend we are a newly-hired wine taster and have discovered a lot of information about wines. And with this information, you want to figure out some of the comparisons and the contrasts between different types of wine. To do this, we are going to use some statistics to help us in identifying the wines in the data set and give us some knowledge out of that as well.

Before we get going into statistics and some of the different parts that come

with it, we need to take a look at a definition of statistics and why it is so important in working with the Python language. Statistics are many things, so trying to come up with some definition to make it work can be hard. As a field, statistics is the framework used to handle data. This definition is a good one because it will handle all of the tasks for collecting, analyzing, and interpreting the data. Statistics will also refer us to the individual measures that will help us to represent aspects of the data itself.

All of the world's observations, once it is collected as a whole, can be referred to as data, and it can vary quite a bit in nature, from being quantitative or qualitative. Researchers and companies are going to gather the data they need from many locations, including from experiments, from their users, surveys, and more.

These examples are important because they will help us to see another important part of data as well. Observations that are done, no matter how they are done, will relate to the interest's population. According to our last example, a researcher wanted to see some patients suffering from a unique kind of illness. For the data to help with the example we are using now, the population we want to view is a set of wine reviews. When we can clearly define our population, it is a lot easier to get some statistics out of the data and then get the knowledge we want.

One thing we need to spend some time looking at is the measures of central tendency. This is an important thing in statistics because it is going to be a metric that can show a solution to the question of "What does the average of our data looks like?" The word average, of course, is pretty vague because there are potentially a few different definitions we can use to help represent this average. Let's take a look at each of these and some of the coding we need to use to handle statistics in Python.

The first option on the list is the mean. This is a kind of descriptive statistic that will look at the average value in our set of data. While the mean is going to be more of a technical word, people usually will see it just as the average of your group. How can we calculate out the mean we want to work with? We will take a look at an example in a minute.

When we are talking about the mean of our data, we are going to refer to a

typical value. The mean is going to basically be the typical observation of our set of data. If we were then able to pick out one of the observations in a random manner, then we're likely to get a value somewhere near the mean. The calculation of this kind of mean is actually easier to handle in Python. Going back to the example we talked about earlier with wine, let's figure out what is the average score for a glass of wine in our set of data.

What this code is going to tell us, based on the set of data we are using, is that the typical score of our set of data will be near 87.8. This means most wines in our set of data are going to be rated highly, with the assumption that the scale is between 0 and 100. However, we must take note that the Wine Enthusiast site chooses not to post many reviews where the score is less than 90.

Now that the mean is handled in our statistics, it is time for us to take a look at the median. The median is important because it is also going to attempt to define a typical value in the set of data you want to work with. Unlike the mean though, the median is not going to need a calculation at all.

To help us find this median, our first goal is to go through and do some reorganization of our data so we can set it up in ascending order. Then the median is going to be the value that coincides with the middle of this set of data. If you end up with an even amount of items, then we can take the average found of the two values that are around our middle.

You will find the standard library of Python can't support a median function, but there is still a way to find it using the method we just talked about above.

While we are here, we need to take some time to watch out for our outliers. Remember that when we calculate out the mean, we are going to sum up all of the values we have in the set of data and then divide this by the number of items. But the median, on the other hand, is going to be found when we can rearrange the items.

There are times, though, where we will end up with outliers. Outliers are items in the set of data that will have a disadvantage on what we get in the mean, but doesn't matter as much with the median. This means the mean is going to have some trouble with being robust to the outliers. The median

though, since it doesn’t need to worry about the outliers, is going to be more robust to them.

To help us see whether there are any outliers present in our data, we need to first take a look at some of the prices found there. We want to focus on the minimum and maximum prices seen in the data and figure out what they are all about.

As we can see by the output, there are already some outliers present in the data we are working with. These outliers are going to represent some interesting errors or events that show up in the data, so we need to know when they are there and determine how they are going to affect the work we are doing.

And the last part that we need to take a look at when we come to a measure of central tendency is all about the mode. This is the value that appears the most in our data.

A value that often shows up in data is going to really influence the average we see towards the modal value. The more we see this value in our set of data, the more it can make an influence on our mean. Thus, the model is going to be important because it comes in and represents the highest weighted contributing factor for the mean.

Similar to what we are going to find with the median, you will see that Python doesn’t have a built-in mode function. It is possible for us to figure out what the mode is, though, if you count out the prices' appearance and look for the maximum.

The mode and median will be very close to each other, so we are going to have some kind of confidence that both of these will help us to find out the middle values and prices of our wines. The measure of central tendency will be useful when it is time to do a summary of our data’s average. Knowing how much the data is spread out will not always be given to us. These measures of spread will help us learn more about the data, and figure out some of the different things about our data before we proceed.

Another thing we may want to focus our attention on along with the mean, median, and mode is the standard deviation. This is a good way to see what

the spread of our observations is like and can even tell us how much any of the data is going to deviate from our typical points in the data. It can let us know more about some of the outliers present in that set of data as well.

The standard deviation is going to be used and calculated because it can give us a bit more information about the prices and the scores that come with our wines. Calculating the cumulative sum of numbers is going to be long and tedious when we are doing it with our hands, but for loops are going to help this to be a lot easier. To make this happen, we need to create some of our own functions because it can help make sure that Python helps us get the statistics done. It is also a good thing to know that the NumPy library is going to implement the standard deviation under std.

The results we get with this is thanks to some of the other options we worked with. The scores here are going to range somewhere between 80 to 100, so we knew ahead of time that the standard deviation would not be that large. In contrast, we have some outliers in our prices and these are going to give us a higher value with the standard deviation. We will find that the variance is going to be related pretty closely to the standard deviation we want to work with.

Working with statistics can give us a lot of information on what we are doing with some of our coding, and what we should work to accomplish as well. Many programmers will work with the Python language to help them do a data analysis, and find that working through a process similar to what we just did above, although really much bigger than what we would use for this example, can make all of the difference.

Computational Problems in Statistics

The next thing we need to work with is some of the computational problems that can come up in the statistics you work with. Starting with some of your data, which can come from a simulation or an experiment depending on what you would like to do, we are going to use the process of statistics. The statistics can answer a few questions for us including:

1. How well is the data going to match some of our assumed distribution?
2. If the data doesn’t match all that well, but we think it is still going

to belong with a known family of distribution, can we estimate the point estimate or the parameters?
3. How accurate are our interval estimates or parameter estimates?
4. Can we estimate the entire distribution, including the approximation or the function estimation?

In most cases, the computational approaches used to address these questions are going to have a few different points in mind to make it easier. Some of these will involve:

1. Minimization of our residuals or the least squares.
    a. Numerical optimization
2. Monte Carlo methods
    a. This can include the simulation of a null distribution, which can be like the permutation and the bootstrap.
    b. Estimation of a posterior density which would include something like EM, MCMC, and Monte Carlo integration.

Managing Your Python Data

The final thing we are going to take a look at in this guidebook is how this language can help us to handle some of the data we are dealing with. Python is a great option for managing our data. Data scientists are already going to know that these databases are going to come in a variety of forms. For example, when we work with AskSam, we will find it is a type of free-form database that is more textural.

We have to remember with this one that the majority of the data our company is going to use will rely on these relational databases because these will provide them with some simple means of organizing massive amounts of data that is more complex, doing so in a manner that is more organized, and it can help us to manipulate the data we are working with.

There is one main goal that comes with this database manager. And this goal is to help us to make the data as easy as possible to manipulate for our needs. Whether this is so we can create our own models or algorithms, or we want to make sure we can manipulate the data we are using. The focus of most of the

storage we want to use for our data is to ensure the data is easier to retrieve overall.

You can have the choice to work with a relational database. These will accomplish the retrieval of data objectives and the manipulation of data with ease. However, because the needs of your data storage will vary all of the time, there will be a lot of products you can choose from when it comes to the relational database. In fact, for a data scientist, the proliferation of the different types of DBMSs using different layouts of data is going to be one of the biggest problems a professional is going to experience when it is time to create a set of data that is comprehensive and good enough for your analysis.

The one common denominator between the variety of databases you can work with is that they will rely on the same kind of language to do this. Often, they are going to work with SQL but we can make some changes to ensure we can use this along with some of our needs with Python. For example, you may use SQL in order to pick out the items you want from the charts or to gather up the information, and then Python will help to manage the data through their models and algorithms to make things easier for you to work with overall.

The SQL will be helpful because it allows you the ability to perform all of the management tasks needed in a relational database. The SQL will help us to pull up the data we would like, and they can even take that data and shape it in a certain way so the need to go through additional shaping at a different times won't be necessary at all.

Creating a connection to a database can turn into a more complex undertaking. For one thing, you have to know the right steps needed to connect to a particular database. However, this is a big process and you do have the ability to divide it up into some smaller pieces of it if needed.

The first step you need to work with when it is time to begin this process is to gain some access to the engine of the database. You will need to work with two lines of code that we will have written below:

```
from sqlalchemy import create_engne
engine = create_engine(‘sqlight:/// :memory:’)
```

After you have been able to access your engine like you wanted to, it is time to use that engine to perform some of the tasks specifically needed for DBMS. The output of our read method, in this case, is going to be the DataFrame object that can hold onto the required data you want. You can also write out the data. But to do this process, you need to either use an object from the DataFrame you already have, or you need to go through and create your own.

There are a few methods the programmer can use to get things started with the data they want to handle. Some of the most common ones to use with the data will include:

1. Read_sql_table(): This will help us to read the data from an SQL table to a DataFrame object.
2. Read_sql_query(): This will read the data from either an SQL table or a query over to a DataFrame object
3. DataFrame.to_sql: This one will write the content we need for a Dataframe object to the tables we specify to use in that database.

The sqlalchemy library will provide us with some support for a broad range of SQL databases. This can include options like SQL Server, PostgreSQL, MySQL, SQLite, and other relational databases like those you can connect to using ODBC or Open Database Connectivity.

Handling your data will be an important step to the process when you are trying to handle a number of tasks for your business. Many companies will spend their time gathering up the data they have in the hopes of getting it organized and ready to use. With the help of this data, they can handle their customer issues, learn about which products to release, and figure out the best course of action to take to beat out the competition. When we can add in some Python to help us get the work done, it can make things a whole lot easier and ensures we will get the results we want in the process.

Chapter Summary

Many businesses will choose to learn about the Python coding language because it helps them to deal with the statistical and data problems they encounter on a regular basis. Learning how to handle these issues, and why

they are so important to a business can make a world of difference in the amount of success you are going to see with this language and with your business as a whole.

In this chapter, we took some time to describe a few of the different parts, and how we can bring them together to make things easier. Whether you handle data and statistics on a regular basis, or this is something that is pretty new for you, it won’t take long before you see all of the ways Python can help you see some success with it.

Chapter 14: Working with Excel Charts and Spreadsheets

When it comes to handling our data and getting things organized or data analysis, one of the best tools we can work with is a spreadsheet. These spreadsheets are set up to help us handle a lot of data and get it put into a usable format that works for our needs. Getting started with these charts and spreadsheets can seem hard, but it is going to be one of the best things to help us organize our data and ensure it is ready for the analysis and any of the models and algorithms we would like to use with it.

With this in mind, the Python coding language is a good option to work with when it is time to handle some of our data overall. You will find that when we rely on this language, we can create our own spreadsheets with all of the right parts in place right from the beginning in no time.

If you are ever working on a project where you need to extract some data out of a database or log a file into an Excel spreadsheet, or if you often have to convert one of these spreadsheets over into a format that is more usable for your program, then this is going to be the best chapter for you to use.

We can handle all of our spreadsheet needs in one place and get the needed benefits along the way. To help us get started with this, we need to first jump into the idea of the openpyxl library and what this can do for us when it comes to creating some of our own Excel spreadsheets.

The first thing we need to take a look at here, though, is the practical use cases. This will include information on when we would really need to use a package like openpyxl for our work. We are going to take a look at some of the different situations when this is needed, but if you plan to work on Excel sheets or data science, then this is definitely one of those extensions you should consider adding to the mix as well.

One way you can use this extension is when you want to import new products into a database. Let’s say you are responsible for tech in an online store company, but your boss wants to save money and doesn’t feel like using an expensive, although cool, CMS system to get the work done. Every time they want to have you add in a new product to their store online, they are going to

come over to you with a nice spreadsheet with a few hundred rows in hand. And for each of these, you have a lot of information, including the product name, the price, the description, and more that you will need to work with here.

Now, to make sure you can import the data, you will need to take the time to iterate over each row of the spreadsheet, and then add each product to the online store. This can be tedious and take a long time if you are doing it the traditional way, especially if you are working with hundreds of products along the way. The good news is there are a few available options, especially when we are talking about the Python language for coding, to make this a bit easier.

And that brings us into our second part. Here we are going to work on exporting the database over to our own spreadsheet. Let's say we have a table for our database where we are going to record all of the information from our users, including their names, a phone number, an address, an email address and so forth.

This is valuable information your marketing team would love to have because they can use it for a lot of different purposes. For the purpose of what we are doing here, the marketing wants the information in order to contact the customers and give them a discounted offer or some other kind of promotion. Even though this is a great idea that can help your business, the marketing team doesn't have access to the database, and they don't know how to work with the SQL language in order to get the information out of that database in the way they want.

This brings up the question of what you can do to help them out? Well, you can work with some of the scriptings available with openpyxl and use that to iterate over all of the user records. And when the script is done doing that kind of work, it can put all of the essential information into an Excel spreadsheet for you to use.

And finally, we need to take a look at how we can use this extension to help us append information to a spreadsheet that has already been created. You may also have to open up one of your spreadsheets and read some of the information to it. Then, after you look it over and see what is already inside

of that document, you may decide it is important to go through and append more data to that spreadsheet.

There are many times we would want to do this to make sure the information in that spreadsheet is as accurate as possible. Maybe a name changed or someone moved. Maybe there is more information or are more products you need to add. Or maybe we have the issue of needing to fix something that was not done right.

For example, when we go back to the example of the online store from above, say that you have an Excel sheet with a list of all the users, and your job is to go through and append each of the rows to see how much these customers have been able to spend in the store. This data is already in your database, so for you to get it done, you have to read through that spreadsheet, iterate through each of the rows, fetch the total amount you are going to spend from the database, and then write it all back into the spreadsheet as needed.

As you can see, there are a lot of different things we can do in order to get started with creating our own spreadsheet with the help of the Python language. It is a simple process we can handle and get done, and learning how to put the parts together and ensure that they are working can be critical to keeping your database up and running. No matter what your business decides to do with that information, having an up to date and accurate spreadsheet is very important.

The Terms You Should Know

Before we get into some of the codings we can do with this language, let's take some time to learn a few of the basic codes and parts that are come with this. There are many different terms we can work with, but some of the best ones to know ahead of time include:

1. Workbook or spreadsheet: This is the main file we are either in the process of creating or we are working on.
2. Worksheet or a sheet: The sheet will be there to help us split up different kinds of content within the same spreadsheet. It is possible to have one or more sheets in our spreadsheet.
3. Column: This is known as a vertical line, and it is represented by

an uppercase letter: A

4. Row: This is a horizontal line, and it will be represented by a number: 1.
5. Cell: This is a combination of the row and column that we just talked about. It will be represented by both a number and an uppercase letter: AI.

Getting Started with openpyxl

Now that we know a bit about these spreadsheets and how they work for the programs we want to write; it is time for us to get started with the Python library that will help us to handle our Excel spreadsheets in this language. For this part, we need to use Python37 and open up the version of openpyxl that is 2.6.2. To install the package, you will need to work with the following code:

```
$ pip install openpyxl
```

After you have installed the package, it is time for us to create our own super simple spreadsheet along the way as well. The code we can use to make this happen includes:

```
from openpyxl import Workbook

workbook = Workbook()
sheet = workbook.active

sheet[“A1”] = “hello”
sheet[“B1”] = “world!”

workbook.sove(filename=”hello_world.xlsx”)
```

Now, if you used the code in the proper manner, the code above is going to help us create a file known as hello_worldxlsx in the folder you would like to use to run the code. If you open up the file with the help of Excel, you should get a spreadsheet like you are used to, and then the first two places on top will say hello and world.

Of course, we can go through this and add in as much complexity and more

to the process as possible. But we have to be careful about how we add in the different rows and columns. If you are careful about doing this, you can fill up the whole table using some of the codings listed above to help you get it done.

Chapter Summary

When you are handling data and other important parts of your Python code, there are times when you will need to create your own spreadsheets. This will help us to take care of a lot of the issues that may show up with some of the codings you will do and can make it easier to form our own data set, perfect for data analysis.

As you can see with the information above, it is easier to get this started than it may seem, and with a few simple steps, you will be able to get the right library, and all of the other information you need, set up to handle creating your own Excel spreadsheet and ensure all of the relevant information is inside.

Chapter 15: Graphics and Images

We also need to take some time to look at how the graphics and images in Python can work with graphics and images a bit. We can choose to have Python bring up an image when it is needed and that will ensure the webpage or another product we are working with will be able to contain some of the graphics we are looking for.

When we talk about computer graphics, it will teach us how a pixel that is on our screen can then be manipulated in a manner that helps us to draw some beautiful shapes, typography, and any of the illustrations we want to look for. Many of the devices we like to use today, including all of our smart devices, were developed with the use of computer graphics.

Despite the idea that the world of computer graphics has evolved so fast and there is a lot of great software that has been developed to handle this as well, we can't generate images on the fly with any of these. In order to do this, we need to be able to get to a level of programming where there is no drag and drop, none of the fancy select all makes bold kind of shortcuts on the computer, no cropping, and none of the copying and pasting that we used to in the past.

Even though we can't generate some of the graphics we want on the fly, at least with some of the different options out there for graphics right now, this doesn't mean we are at a complete loss for what we need to get done. In fact, we can make this happen with our coding.

For us to work on this project, we are going to start by learning how to draw one of our own dynamic text data on an image. You first need to open up your favorite text editor (most of them will work so just go with the one that is already installed on your computer) and make sure you have pip and Python installed on your computer. If you don't have this present yet, then it is important to get it all set up so your process is going to work.

After the pip and Python are ready to go on your system, it is time for us to open up the shell you would like to use, and then we need to work to install the Pillow library and all of the dependencies that come with it. Pillow is going to be a part of the Python Image Library or PIL that was started and

maintained by a variety of contributors, including Alex Clark. This is based on the code that comes with PIL and has evolved to be a better version for some of this work than PIL originally was.

When you are working with Pillow, you will find it is going to add on some support for opening, manipulating, and saving many of the different file formats for images. A lot of the things you work with on Pillow are going to be the same as what you can do with the original PIL.

Downloading Pillow on your computer is going to be easy. It is available to use with all three of the main operating systems, including Linux, Mac OS X, and Windows. The most recent version as of the time of writing this guidebook is 2.2.1 and you can use it with Python 2.6 or above, so it should work with most of the installations of Python you already have. To install this on your Windows computer, you would use the code below to help:

```
easy_install Pillow
```

You can also install this on the Linux machine, and the code is really not much different than what we see with the Windows installation option. To help you to get Pillow up and running on a Linux operating system so you can work on your images and graphics, you would use the following code:

```
sudo pip install Pillow
```

And finally, we are also able to spend some time adding the Pillow extension onto our Mac OS X system as well. There are a few more steps to take in order to make this happen. You first need to install the XCode and then install all of the prerequisites through Homebrew. After you have installed this on your system, you will need to run the following codes:

```
$ brew install libtiff libjpeg webplittlecms
$ sudo pip install Pillow
```

Now, we have the Pillow extension on our computer and ready to use. We need to know how to take this in order to write some code. You will find that Pillow is actually a really extensive library when it comes to what we can do with graphics and images. But we are going to keep it simple for now and only work with the following classes to help us get done:

1. Image: This is used in order to create an image object, such as what we would like to do with our greeting background.
2. ImageFont: This is the font of the text we are going to draw on the greeting.
3. ImageDraw: This will create a drawing context.

With this in mind, we are going to work on creating a background that we would like to work with. This is going to be a simple background image that we can work with and add in some different colors and words along the way as well. We can initialize it with the following code:

```
# import required classes

from PIL import Image, ImageDraw, ImageFont

# create an Image object with the input image

image = Image.open('background.png')

# initialize the drawing context with
# the image object as background

draw = ImageDraw.Draw(image)
```

For creating the imageFont we talked about earlier, we need to make sure we have the right files to make this happen. You can use any of the fonts that work the best for your needs. For our goals here, we are going to use the Roboto font, which is something that we can download when we visit the Google Fonts GitHub rep. The code we are going use to make this happen is:

```
# create font object with the font file and specify
# desired size

font = ImageFont.truetype('Roboto-Bold.ttf', size=45)

The # starting position of the message

(x, y) = (50, 50)
message = "Happy Birthday!"
color = 'rgb(0, 0, 0)' # black color
```

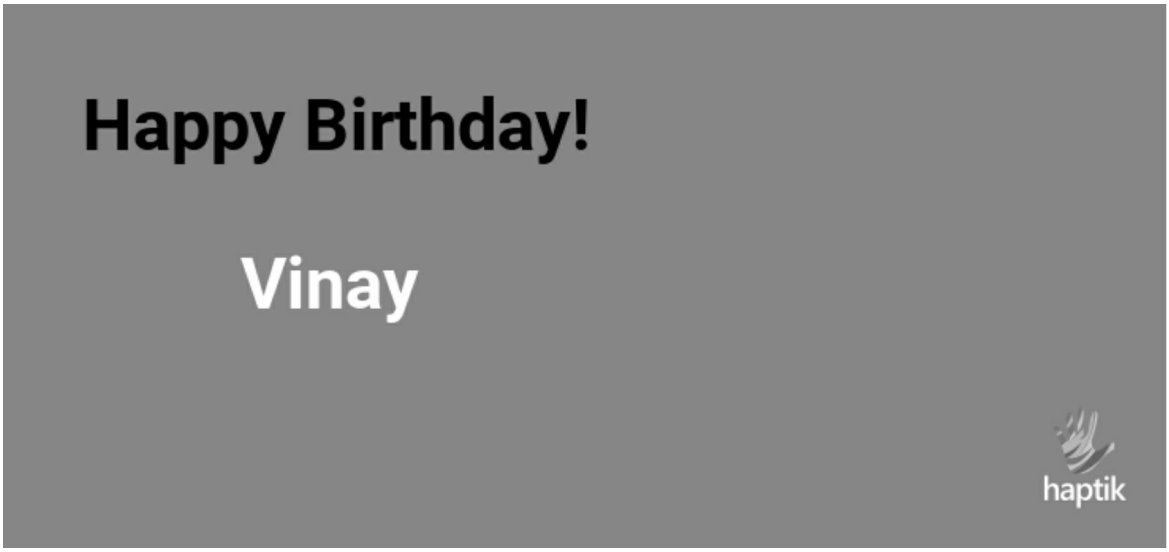
 # draw the message on the background

draw.text((x, y), message, fill=color, font=font)
(x, y) = (150, 150)
name = 'Vinay'
color = 'rgb(255, 255, 255)' # white color
draw.text((x, y), name, fill=color, font=font)

 # save the edited image

image.save('greeting_card.png')

If you have been able to go through this code and write it out the proper manner, the output you will get when it is all done will be in the image below:



With some of the fonts you want to work with, you might have to pass an optional parameter that will encode the font. This means it is going to work to tell the ImageFont module while encoding to use the file that contains the font you want to work with.

Computer graphics are going to work on a system that is inverted with the coordinates. The origin(0, 0) that lies at the top left of the image is going to represent the distance of the text box going from the left, and then they are going to help to show us the distance we need to go to the top.

While you take the time to save the image, you can pass some of the optional parameters like quality and optimize in order to control what size the image shows up as when you get started. When you do this, it will generate an output image optimized png. This one will be in a smaller size, but the quality is going to be a bit reduced in the process.

This is a great way for you to learn a bit more about how to work with the idea of creating the images you would like and can make it easier to handle some of the different fonts and colors you are looking for. We just took some time to do this in one example, but you can easily work with Python to handle a lot of the work you want to complete when it comes to text and images in Python.

Chapter Summary

There are many times when you will need to add an image or the text you would like to create and use the Python language to help make this happen. You will find that working with the codes above, and more, will help you to create some of the images and the text you need for websites, and other parts of the code you would like.

Chapter 16: To Relax

Before we end this guidebook, we are going to take a look at some of the codes you can use in order to have some fun with the Python language, and see exactly how this language works. We're going to have a bit of fun with Python, and move off some of the more complex tasks we have talked about so far. This allows us some time to relax, have fun, and put our new skills to the test.

Python is a great language to use when you are learning how to program, and it is the perfect solution for those who would like to get things done while not having to spend a lot of time just learning how to use the code. And that is why we are going to make sure that we can create a few games that will help us to enjoy coding, rather than seeing it as a chore all of the time.

Python Sprites

When we come to work on some of our codes in Python, and we want to create some of our games, then we are going to come across a term known as Sprites. Sprites are the various objects inside of a game. These include building, characters, and anything else you need to have interaction with each other when you are working on a game.

The reason you will not be able to find a subclass for these Sprites is that it is seen as more convenient not to have these. When an object can inherit from the sprite class, it is going to be added right into the sprite group you want. We can't add in the sprites to the sprite group unless we take the time to inherit from our sprit class. This is going to be useful because the programmer cam then do things like update all of the sprites that are in the same group and even draw them all with just one function. It is also possible to work with these for collision detection if we would like.

Creating a Drawing

The first project we are going to work with will help us create a simple drawing. We are going to use the Arcade library from Python, so make sure to visit that website and get it installed on your computer as well. We are going to teach the computer how to create a simple drawing, and you will be surprised at how few lines of code you will need to use to make this happen.

For this example, our goal is to draw a smiley face. The code you will need to use to make that happen includes:

```
import arcade

# Set constants for the screen size
SCREEN_WIDTH = 600
SCREEN_HEIGHT = 600

# Open the window. Set the window title and dimensions (width and height)
arcade.open_window(SCREEN_WIDTH, SCREEN_HEIGHT, "Drawing Example")

# Set the background color to white.
# For a list of named colors see:
# http://arcade.academy/arcade.color.html
# Colors can also be specified in (red, green, blue) format and
# (red, green, blue, alpha) format.
arcade.set_background_color(arcade.color.WHITE)

# Start the render process. This must be done before any drawing commands.
arcade.start_render()

# Draw the face
x = 300
y = 300
radius = 200
arcade.draw_circle_filled(x, y, radius, arcade.color.YELLOW)

# Draw the right eye
x = 370
y = 350
radius = 20
arcade.draw_circle_filled(x, y, radius, arcade.color.BLACK)

# Draw the left eye
x = 230
y = 350
radius = 20
```

```
arcade.draw_circle_filled(x, y, radius, arcade.color.BLACK)

# Draw the smile
x = 300
y = 280
width = 120
height = 100
start_angle = 190
end_angle = 350
arcade.draw_arc_outline(x, y, width, height, arcade.color.BLACK,
start_angle, end_angle, 10)

# Finish drawing and display the result
arcade.finish_render()

# Keep the window open until the user hits the 'close' button
arcade.run()
```

Creating a Hangman Game

Now that we have gotten some practice with working in Python and having some fun with our own smiley face, it is time to take this to the next level and learn how to create one of our own games. We are going to work with the classic Hangman Game. Once it is done, you will be able to play a game of Hangman by yourself or with a friend. You will be amazed at the amount of power and the amount of fun you can have with this one just by working with this code.

You will find there are a lot of the basic parts you need with Python coding even in this game. There will be a loop present in there because it allows us a few chances to make guesses before the game considers itself over. This is important because you do not want the game to stop after you guess just one letter, and you don’t want the game to get stuck and freeze up on you.

You may also notice there are a few conditional statements that show up in this Hangman game. The conditional statements are important to this kind of game because they are going to let us know whether the letter we chose was the right one or not for that word. And for the code that we are writing here,

you are not responsible for picking out the secret word because we are going to set Python up to pick out the word. This allows you the chance to actually play the game as well. We can work with the conditional statements to put up the correct letters when you guess them, or tell you that you are wrong when you choose incorrectly.

Even though there are a lot of different parts that come with this kind of code, it is actually a simple one to work with. You will enjoy all of the basics that are in it though, and the fact that, even though it is longer and is creating a game for you, it is still able to use some of the basics you already know. The code you need to use to handle your Hangman game includes:

```
#importing the time module
import time

#welcoming the user

print "Hello, " "Time to play hangman!"


#wait for 1 second
time.sleep(1)

print "Start guessing..."
time.sleep(0.5)

#here we set the secret
word = "secret"

#creates an variable with an empty value
guesses = ''

#determine the number of turns
turns = 10

# Create a while loop

#check if the turns are more than zero
while turns > 0:
```

```
# make a counter that starts with zero
failed = 0

# for every character in secret_word
for char in word:

# see if the character is in the players guess
    if char in guesses:


    # print then out the character
        print char,

    else:


  # if not found, print a dash
        print "_",


    # and increase the failed counter with one
        failed += 1

# if failed is equal to zero

# print You Won
if failed == 0:
    print "You won!"


# exit the script
    break

print

# ask the user go guess a character
guess = raw_input("guess a character:")

# set the players guess to guesses
guesses += guess
```

```
# if the guess is not found in the secret word
if guess not in word:


 # turns counter decreases with 1 (now 9)
    turns -= 1


# print wrong
    print "Wrong"


# how many turns are left
    print "You have", + turns, 'more guesses'


# if the turns are equal to zero
    if turns == 0:


    # print "You Lose"
        print "You Lose"

        #if the turns are equal to zero
        if turns == 0

        #print “You Loose”
```

Creating a Magic 8 Ball

And the final project we are going to spend some time looking over and learning how to work with is the Magic 8 Ball. We are going to create a program that is just like that little Magic 8 Ball you enjoyed as a child, but this time, it is all going to happen on your computer. With this one, we can ask the computer questions, click on it to start, and we can get an answer that is randomly picked by the computer.

The code we will write out will help the computer to determine which answers to give you to based on the questions that you ask. You do have some freedom here because you get the benefit of picking out the different answers that happen as well. We are going to keep this simple, but you will

notice there are options like the if statement and the elif statement that shows up in this kind of code and makes it easier to use for your needs.

Writing out the code for this may take a few minutes since we want to add in a few different parts to the mix, but this doesn't make the process any harder overall. The code we need to create our own Magic 8 Ball will include:

```
# Import the modules
import sys
import random

ans = True

while ans:
        question = raw_input("Ask the magic 8 ball a question: (press enter to quit)")

answers = random.randint(1,8)

if question == ""
        sys.exit()

elif answers ==1:
        print("It is certain")

elif answers == 2:
        print("Outlook good")

elif answers == 3:
print("You may rely on it")

elif answers == 4:
        print("Ask again later")

elif answers == 5:
        print("Concentrate and ask again")

elif answers == 6:
        print("Reply hazy, try again.")
```

```
elif answers == 7:
        print("My reply is no")

elif answers == 8:
        print("My sources say no")
```

With this one, we decided to pick out eight options since it is true that we are working with a Magic 8 Ball. You can change up how many answers you would like to have, and whether or not you would like to have more or less to make this fun. You are just using the elif statements here so you can simply add in more of these to get the results you would like.

Chapter Summary

When many programmers take a look at the Python language and all that it can do for them, they are focusing more on how they can use this language to help them create an application, work on data analysis, or some other technical use of this coding language. But this is cutting out some of the fun we can have when it is time to code. When you want to have a bit of fun with the Python language while still learning more about coding and how it all works, make sure to try out a few of the options above to help you out.

Conclusion

Thank you for making it through to the end of *Python Programming for Beginners*. Let's hope it was informative and able to provide you with all of the tools you need to achieve your goals, whatever they may be.

The next step is to get started with writing out some of the codes that you would like to use with Python. Even as a very beginner in the process, you will find that working with the Python language is going to be as simple as possible. While there are some complex things you can focus your time and attention in order to get this work done, you may find that, as a beginner, you can handle and understand a lot of the codes in this guidebook as well.

In this guidebook, our aim was to take you from being a complete beginner in the world of coding and of Python, in particular, and change that up a bit so you have more experience and knowledge about how this coding language works. From some of the basics about what Python can do for you, all the way to more specifics that include some of the codings, you will be able to handle many of the challenges that come with the Python language.

There are a lot of things we can do when we rely on the Python language to get our work done. And often, the reasons for working with the Python language in the first place are going to be pretty vast. For example, you may want to use this to create your own game, create your own calculator, and work to handle data science and your own data analysis at the same time.

But before we can jump into all of that complicated stuff, we need to make sure that we have some of the basics of working with the Python language and how to write some of our own codes. With that in mind, make sure to check out this guidebook and learn exactly the steps you need to take to help you get started with your goal of writing codes in Python!

Finally, if you found this book useful in any way, a review on Amazon is always appreciated!