Python Programming for Intermediates
A Complete Crash Course on Python Programming

Adam Stewart

Table of Contents

Introduction

If you are looking for a great program that will allow you to write a great code without all the hassle or the messy code to read through, going with Python is one of the best options to choose. When you look at the code, you will see that it is really easy to read, even when you have no experience with writing code. When you get started with using it, you will see that it is really easy to understand and learn even though it still gives you the power that you are used to with some of the other language types.

For those who have had some experience with using programming languages, this is the guidebook for you. It is a comprehensive look at Python, providing some more examples and in-depth information on what you are able to do with the codes that you are learning. Not only are you going to get some information and a few syntaxes that you are going to have to figure out on your own, but you also get the chance to see some of these in work and learn how they will place on your computer when working in Python.

All of this will come together to give you some experience as well as the confidence that is needed to do really well with the Python language. While you may have some of the background information in programming or with Python in particular, getting started with this kind of code is sometimes scary and a bit confusion. But with the help of this guidebook, you are going to get some real life experience to work with and it will be easier than ever to get the work done for you.

So when you are ready to put some of your practical learning to use and you want to have some actual choices that will help you to see results when you want to put your codes to work, make sure to check out this guidebook and see how it can all work out for you without all the hassle or headaches.

Chapter 1: Getting Started with Python

Getting started with Python is a surprisingly easy process. You will love how simple this programming can be and how it will work on a lot of different programs. It is simple enough for the beginner to read through, once they learn some of the tricks of the trade, but it still has the power that you want when working on a programming language. It has the best of both worlds, which is why this programming language is one of the best options out on the market to choose from.

What is Python?

Python is a high level programming language that will use Object Oriented

Programming, or OOP. It is often used as a glue language to help connect other components together and can be used as a general purpose programming language. Because it is so versatile, it can be used with some of the more powerful programming languages, and easy to read, it gets rid of some of the brackets and other messy stuff that is in other languages, it has been ranked as one of the most popular languages for programming in the whole world.

If you are looking to create things like scripting interpreters, web applications, and even applications on the desktop, Python is the best program for you. It is considered open source right now, meaning that no one owns the rights to using it exclusively. This provides you with many different choices with the language as anyone can take the code and make it better all throughout the world. The best part is, it is powerful, can be placed on many different computers and computer systems, and it is free.

Versions of Python

There are a few different versions of Python that have come out over the years and they all have some different benefits as to why you should choose them. The most updated one is Python 3, although there are a few versions of this out right now. Most professionals are still using Python 2 because it is able to go back and work with older versions if needed while Python 3 is not able to do this. There are no newer versions of Python 2 that have come out in the past few years so it is likely that it is going to be phased out soon and Python users will have to stick with their older version or choose Python 3.

The original create of Python, Guido van Rossum choose to develop this language because it was simple, based on the English language, so that programmers new and old would be able to use the program without having too much complication. The language has undergone many changes throughout the years, especially since it has become open sourced for other companies and individuals to work with, but it has still maintained its simplicity and ability to work well with people who are new to programming.

Downloading Python

Let's have a little refresher course on how to download and install the Python program for those who may not have this program on their computers. You can download this program by visiting www.python.org/downloads or you can pick one of your favorite distribution sites and download the version that you want. You will need to select the python windows installer and then follow the simple steps that follow.

You should notice that the setup wizard will come up; it is easiest to just click on "Next" for all the options during the following steps in order to get everything to show up in default. If you would like to customize some things or make some changes, you are able to select these as well when going through the setup.

After Python has been successfully installed on the computer, you should select IDLE in order to get started. This is basically going to be the part that allows you to work on

Python so you need to make sure that you have it opened so that you can start writing your code. You will also need a text editor to work with so that the program is able to go through and read the work that you are doing. For a Windows computer, working with Notepad is a great option or you can pick out another option online.

How does Python Execute a Program

Each programming language that you work with is going to execute a program a bit differently. This is why it is so important to learn how to organize the words and the different statements that the program requires so that you avoid errors and other issues in your code. In this chapter, we will take some time to learn how Python will execute the commands that you give and basically how the whole program runs.

When you are working with Python, you are working with an interpreted programming language. You will have a text interpreter that will execute each of the programs going line by line and then will convert it into a code for the process to understand the words and carry them out for you.

Python is also a scripting language, so you can write out the script and then save it using the extension .py or you can directly write it and then execute each statement into the Python shell.

Internally, Python is going to work to compile your program, basically the source code, into a byte code that has the .pyc extension, just like the Java byte code. This makes it easier for the code to be executed without the delays and you will be able to see it come up in just a few seconds rather than waiting around.

You will be able to save your byte code files into a subdirectory that is called __pycache__ which is located in the directory where the source file resides. For example, if you wrote out helloworld.py it is going to then be converted into one of these byte codes and renamed helloworld.pyc.

You can go in and manually compile this code if something does go wrong, but for the most part, Python is going to do the compilation for you so it won't be an issues. As a beginner you may wonder where some of these .pyc suffixes come from, but Python is going to store them with that specific suffix so don't worry if it shows up.

Of course, this is only going to happen if Python has the write access, but even if the Python has no write access, it may not be saved that way, but the program is still going to work.

Whenever you call up a Python program, Python is going to check if there is already a compiled version with this .pyc suffix. This file should be newer than the .py suffix and if it exists, the Python will load in the byte code to speed up how fast the script is able to go. If the byte code doesn't exist on your computer, Python will work to create your byte code before it executes the program.

So basically, each time that you execute a script in Python, you will have a byte code created by the program as well. If the script in Python is imported like a module, your byte code is going to be stored in the proper .pyc file.

Python Implementations

When you hear about implementation of Python, it means that the environment or the program that is providing support for executing your programs inside of the Python language, will be represented with the CPython reference implementation. This means that it is going to help you to work on executing the different codes and statements that you are working on within the program. There are also some variants of the CPython that you can work on and will make a big difference in the way that the program works. Some of the features that are available with the variants include:

- CrossTwine Linker—this is going to be a combination between CPython and an add-on library of your choice. It is going to offer some better performance when it comes to the code that you are working on.
- Stackless Python—this is CPython that has an emphasis on concurrency while using channels and tasklets. This is often the kind that is used for the dspython on programs like the Nintendo DS.
- Wypthon—this is considered a re-implementation of some of the parts of Python, which will drop the supports of using bytecode in order to use the wordcode

based omdel. It is going to use the stack register in the implementation and adds in lots of other types of optimization.

Implementation is everything when it comes to how you are able to work on your programs and can help you to get more done with Python compared to some of the other programming languages. What is so unique about Python is that it is able to work with a lot of other programming languages in order to still be simple to work on the code plus has all the power that you need to really get things done. Some of the other implementations that you may want to consider if you want to do something specific with the Python programming language include:

- Brython—this is one of the implementations that you can use in order to run Python in your browser using a translation to JavaScript so you can use these two together.
- CLPython—this is an implementation of Python in common lisp.
- HotPy—this is considered a virtual machine for Python that will support translation and bytecode optimization.
- IronPython—this is Python in C#. C# is a great programming language to use inside of the Windows platforms and is often a competitor to Python based on how popular and easy it is to use. This implementation allows you to translate your work from Python over to C# if you choose.
- Jython—this is the version of Python available for the Java platform.
- PyMite—this is Python that you can use for embedded devices
- PyPy—this is Python within Python so that you are able to target a few different environments at the same time.

- RapydScript—this is a language that is similar to Python that will compile into JavaScript so that you can use it in the Java platform without having to go with all of the difficult language issues.

Working on Python can be a great experience. If you are just a beginner with the idea of programming and are unsure about how to get all of this started, some of the other programming languages can be a bit confusing. Python is easy to use but has all the power that you want from some of the bigger names in programming language and you get the benefit of getting to use this program along with some of the other popular languages that you may want to work with!

Review of a Simple Program

As we mentioned, using Python is one of the simplest programming languages that you can choose. It isn't going to have a lot of excess around it like some of the other languages, which can save you a lot of time and effort.

It also makes it easier for you and for someone else to take a look at the information and be able to read through it. So let's take a look at some of the things that you can do with Python and how to get started with writing your first program.

The first program that we are going to write out is the "Hi World" program. This one is going to need a Python shell to make it easier and you will be able to test it out on your editors if you do it properly. This makes it easier to have a good idea of what you are doing and to catch any errors right in the beginning. If you are using the Python Shell, which works well on most of the computer types and programs that you may be using with Python, you will simply need to type in the following program to get the information to show up:

Print(“Hi World!)

You should be able to go and execute this information and find that it will show up with the words Hi World! On the screen. This is a simple process to do, but it is going to help you to get things started and provides a good review of some of the simple steps that you need in order to start writing your own program on Python.

Remember that Python is a really easy programming language that won’t have a lot of different brackets and other information that is in the way and will slow down what you are doing. It is also really easy to read.

As you can see, you only needed a few things in place in order to write out the phrase, rather than needing to type out lines of code to get the same result like you would need to do with other programming languages. Let’s take a look at some of the other things that you are able to do with Python programming and how you can even write some good code to get your programs started.

Chapter 2: Some of the Basic Commands, Variables, Statements, and Other Things That You Can Do with Python

There are so many things that you are able to do in order to get a code up and running on Python. Many people may avoid using Python because they think that it is too simple or it just isn't going to get the job done. But in reality, it is simple just for the fact that even a beginner is able to learn how to use it, but that doesn't mean that you aren't able to do a lot with it. This chapter is going to take some time to look at the different commands that you can do with Python programming in order to make your programs and codes come to life.

Variables

Variables may sound like something that is too complicated to learn, but they are basically locations in the memory that are reserved in order to store the values of your code. When you work on creating a variable, you are reserving this spot in the memory. In some cases, the data type that is in the variable will tell the interpreter to save the memory space and can even decide what you are able to store on your reserved memory.

Assigning values to your variables

The value is going to be one of the basic things that your program will need to work with. it can be a string, such as Hi World, 3.14, which is considered a type float, or a whole number like 1, 2, 3 which is considered an integer. Python variables will not need an explicit declaration in order to reserve the space in the memory that you need. This is something that is going to happen automatically whenever you place a value with the variable. For this to work, simply place the (=) so that the value knows where it is supposed to go.

Some examples of this include:

```
X = 10              #an integer assignment
Pi = 3.14    #a floating point assignment
Y= 200        #an integer assignment
Empname = “Arun Baruah”     #a string assignment
```

Keep in mind that when you are working on codes, you are able to leave a comment with your wok by using the # sign. This allows you to explain what is going on in the code, leave some notes, or do something else within the program. It is not going to be read by the interpreter since it is just a little note that you are leaving behind for yourself or for someone else.

The next part is going to depend on which version of Python you are using. Python 2 is fine with you writing out print and then the information you want to talk about but Python 3 is going to require you to place the parenthesis in to make it work. An example would be:

```
Print("y = %d" %y)
Print("x = %d" %x)
Pring("Employee Name is %s" %empname)
```

These would then be put through the interpreter and the outputs that you would get should be

```
X = 10
Y = 200
Employee Name is Arun Baruah
```

Now go through and put in this information to your program and see what comes up. If you didn't get the right answers like listed above, you should go and check that the work is done. This is a simple way to show what you are able to do with Python and get the answers that you need.

Multiple Assignments

In addition to working with the single variables that were listed above, you will also be able to work on multiple assignments. This means that you are going to be able to assign one value to several different variables at the same time. To do this, you would just need to place the equal sign between all of them to keep things organized and to tell the computer that the value is going to be with all of the variables together. You can keep them separated out if that is easier for you, but using this method is going to help you to send everything to the same memory location on the computer and will give the code a clearer look on your screen.

A good example of how to give more than one variable the same value includes:

```
a = b = c = 1
```

This is telling the code that you want all of them to be tied with the value of 1 and that all of these variables should have the same value and that you want to assign them all to the same location within your memory.

Standard Data Types

Another thing that you are able to work on when doing Python is the various data types. These are going to be used in your code in order to define the operations that you can do on each data type as well as explain to others the storage method that will be used for this kind of data. Python has five data types that are considered standard including:

- Numbers
- Dictionary
- Tuple
- List
- String
- Numbers

Number data types are the ones that will store the numeric values. They are going to be created as objects once you assign a value to them. There are also four different types of numericals that Python will support including

- Complex (such as complex numbers)
- Float (floating point real values
- Long (long integers that can also be shown as hexadecimal and octal.)
- Int (signed integers)

One thing to note is that while Python will allow you to use the lowercase l when doing the long form of a number, it is best to go with an uppercase L whenever you are using the letter. This is going to help you avoid confusion in reading the program between the l and the 1 as they look really similar. Any time that Python is displaying a long integer that has the l in it, you will see the uppercase L.

Strings

Strings are identified in Python as a contiguous set of characters that will be shown by the use of quotations marks. Python is going to allow for either double quotes or single quotes, but you do need to keep things organized. This means that if you use a double quote at the beginning of your string, you need to end that same string with the double quote. The same goes when you are using a single quote. Both of these will mean the same thing, you just need to make sure that you are using the proper quote marks to make the code look good and to avoid confusing the Python program.

In addition to being able to print off the string that you would like, you are also able to tell the program to print just part of the string using some special characters. Let's look at some of the examples of what you are able to do with the strings, and the corresponding signs that you will use at well, to help illustrate this point.

```
str = 'Hi Python!'
print(str) #prints complete string
print(str[0])     #prints the first character of the string

print(str[2:5])  #prints characters starting from the 3rd to the 5th

print(str[2:])   #prints string starting from the 3rd character

print(str*2)       #prints the string two times

print(str+”Guys”)    #prints concatenated string
```

For the most part you are probably going to want to print out the whole string to leave a message up on your program so the first print that you do is going to be enough. But if you just want to print out Hi or some other variation of the words above, you may find that the other options are really useful. You can do any combination of these, they are just examples to help you get started!

Lists

Lists are one of the most versatile data types that you can work on in Python. In this language, the list is going to contain different items that are either enclosed with the square brackets or separated out with commas.

They are similar to the arrays that you would see in C if you’ve worked with that program. The one difference that comes up with these is that the items that are in a list can be from different data types.

The values that are stored inside the list can be accessed with a slice operator as well as the [:} symbol with the indexes starting at 0 at the beginning of the list and then working down until you get to -1. The plus sign will be the concatenation operator while you can use the asterisk as the repetition operator. For some examples of what all this means and how you can use the different signs within your programming, consider some of these examples:

list = ['mainu', 'shainu', 86, 3.14, 50.2]

tinylist = [123, 'arun']

print)list)#prints complete list

print(list[0])   #prints the first element of the list

print(list[1:3]- #prints elements starting from the second element and going to the third

print(list [2:]) #prints all of the elements of the list starting with the 3rd element.

*Print(tinylist*2)   #prints the list twice.*

Print(list + tinylist)   #prints the concatenated lists.

Tuples

The next thing that we need to learn about for the Python language is about tuple. This one is pretty similar to what you are going to find with a list, but it is going to use some different signs. The main difference though is that lists will use brackets and the elements, as well as the size, can be changed through the program.

On the other hand, the tuples are going to use parentheses and you will not be able to update them. A good way to think about tuples is that they are going to be like a read only page.

As long as you don't try to make changes to the tuple in the program, you are going to be able to use it in the same way as you did the list examples above. This makes it a great option to use if you're looking for something that is simple but won't let anyone make changes to the program after you are done.

Dictionary

Dictionaries are another kind of tool that you can use when you are working in Python. They are similar to a hash table type and they are going to work similar to the hashes or the arrays that you can find on other programming languages like C# and Perl.

They will also consist of key value pairs and while the key can be almost any type on Python, you will notice that they are usually going to be strings or numbers. For the most part, when it comes to values, you will find that they are an arbitrary object in python.

Some examples of how this will work include the following codes:

```
#dictionary stores key-value pair, later to be retrieved by the values with the keys

dict = {}

dict['mainu'] = "This is mainu"

dict[10] = 'This is number 10"

empdict = {'name': 'arun', 'code':23, 'dept': 'IT'}


print(dict['mainu']) #this will print the value for the 'mainu' key

print(dict[10])          #this will print the value for the 10 key

print(empdict)           #this will print the complete dictionary

print(empdict.keys())  #this will print out all of the keys

print(empdict.values())       #this will print all the values
```

One thing to keep in mind is that these dictionary values are not going to be stored in an order that is sorted. They aren't going to have the concept of ordering among the elements. This does not mean that you can say that the elements are out of order, they are just going to be unordered.

Keywords

Most of the types of programming languages that you will deal with will have some keywords or words that are reserved as part of the language. These are words that you really shouldn't use in your code unless you absolutely can't help it.

There are 33 keywords found in the most recent version of Python and you will need to spell them properly if you want them to do the job that you lay out. The 33 keywords that you should watch out for include:

False

Class

Finally

Is

Return

None

Continue

For

Lambda

Try

True

Def

From

Nonlocal

While

And

Del

Global

Not

Yield

As

Elif

If

Or

Assert

Else

Import

Pass

Break

Except

In

Raise

Keep this list on hand if you are worried about learning the language. It will be able to help you out any time that you have issues with the interpreter about the names that you are giving the variable. You may be confused about why it is giving you some issues with the words you chose, you can go through with this list and see if you used one of the keywords inappropriately within your code.

Statements

When you are writing your code in the Python language, you are going to be making expressions and statements to get it done. Expressions are going to be able to process the objects and you will find them embedded within your statements.

A statement is basically a unit of code that will be sent to the interpreter so that it can be executed. There are two types of statements that you can use; assignment so far and print.

You will be able to write out the statement, or multiple statements, using the Python Shell to do so interactively or with the Python script using the .py extension that we talked about later.

When you type these statements into the interactive mode, the interpreter will work to execute it, as long as everything is properly in place, and then you can see the results displayed on the screen. When there are quite a few lines that you need to write in code, it is best to use a script that has a sequence of statements. A good example of this is:

```
#All of these are statements
X = 56
Name = “Mainu”
```

```
Z = float(X)

Print(X)

Print(Name)

Print(Z)
```

Operands and operators

There are a lot of great symbols that are going to show up when you make a code in your Python program. It is important to understand what parts you are able to work with and what they are all going to mean. Operators are often used to mean subtraction, addition, division, and multiplication. The values of the operator will be called operands. You can use many different signs for these in order to get the values that you would like to see.

While you are using the operators and operands, you need to remember that there is going to be an order of evaluation that is followed. Think about going back to math class and how this all worked. You had to look for specific signs in order to figure out which tool you were supposed to use in order to come up with the right answer. This is the same when using these operands within your code.

When you have more than one of these operators in the expression, you will need to do the order of evaluation based on the rules of precedence. For anything that is arithmetic, Python is going to use the acronym PEMDAS which is parenthesis, exponentiation, multiplication, division, addition, and subtraction. If there are a number of these that are the same, such as two sets of numbers that need to be multiplied together, you will need to work from left to right to get the correct number.

Another important operator that you should look for is the modulus operator. This one is going to work with integers and is going to yield the remainder once the first operand has been divided by the second one.

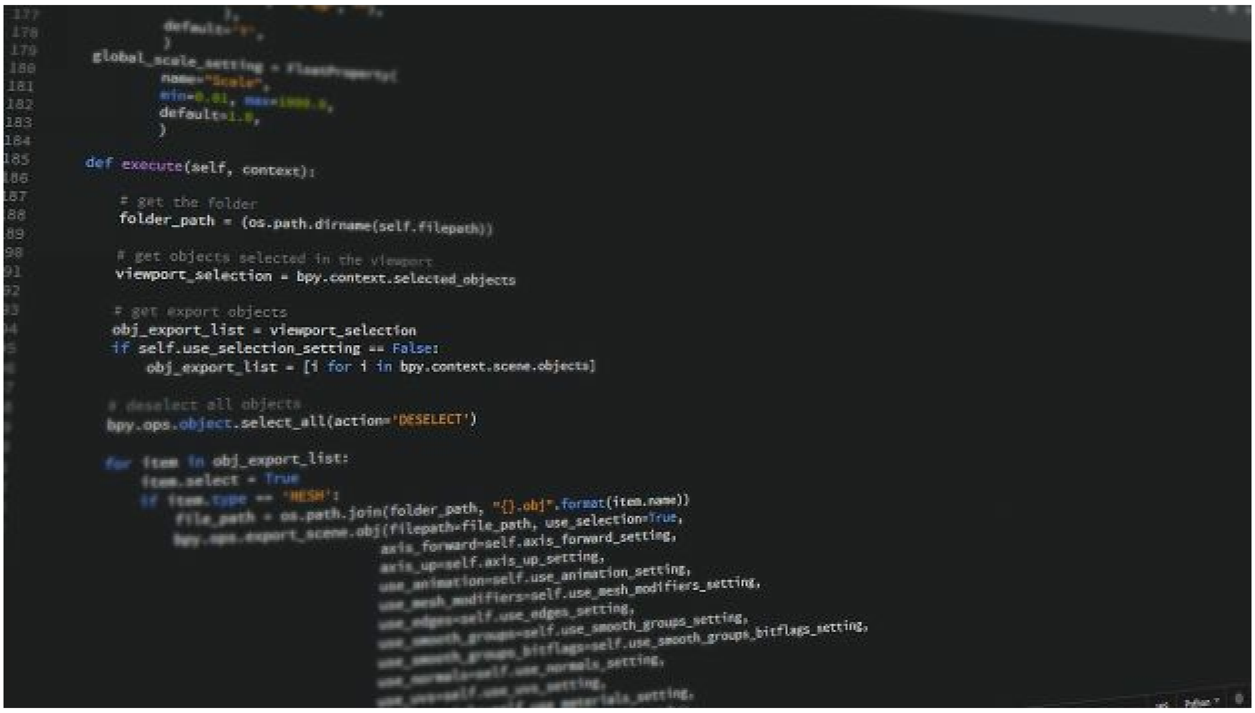
. for Python

One thing that you will find useful with Python is that you are able to send out . in the program. This allows you to add in something to explain what you are working on rather than just leaving it up to the other programmer to figure out. You will simply need to bring in the # sign in order to denote that you are going to leave a little bit of a comment or statement about the code.

Any time that you use the # symbol, the program interpreter is going to ignore what you say rather than trying to write it. While the computer program is ignoring it, it is still there for the programmer to look at if needed.

Using the Python language does not need to be difficult, but you do need to be able to understand what is going on in each part and how all of them are going to help you to get the results that you need. Each one will work slightly differently so that you are able to get the right codes done in Python.

Chapter 3: Understanding the Decision Control Structure



```
                default='Y',
                )
    global_scale_setting = FloatProperty(
                name="Scale",
                min=0.01, max=1000.0,
                default=1.0,
                )

    def execute(self, context):

        # get the folder
        folder_path = (os.path.dirname(self.filepath))

        # get objects selected in the viewport
        viewport_selection = bpy.context.selected_objects

        # get export objects
        obj_export_list = viewport_selection
        if self.use_selection_setting == False:
            obj_export_list = [i for i in bpy.context.scene.objects]

        # deselect all objects
        bpy.ops.object.select_all(action='DESELECT')

        for item in obj_export_list:
            item.select = True
            if item.type == 'MESH':
                file_path = os.path.join(folder_path, "{}.obj".format(item.name))
                bpy.ops.export_scene.obj(filepath=file_path, use_selection=True,
                                        axis_forward=self.axis_forward_setting,
                                        axis_up=self.axis_up_setting,
                                        use_animation=self.use_animation_setting,
                                        use_mesh_modifiers=self.use_mesh_modifiers_setting,
                                        use_edges=self.use_edges_setting,
                                        use_smooth_groups=self.use_smooth_groups_setting,
                                        use_smooth_groups_bitflags=self.use_smooth_groups_bitflags_setting,
                                        use_normals=self.use_normals_setting,
                                        use_uvs=self.use_uvs_setting,
```

There are times in life that you are going to have to make decisions based on the situation around you. Perhaps you woke up one morning and wanted to go for a walk, but it started to rain out. Did you just sit there blankly without another option to help you out? No, you may have woken up and decided to grab an umbrella and go on the walk anyway or you may decide to stay home and read a book. You can make other decisions, even if the first one doesn't work out, based on the circumstances around you.

Now this is kind of the same idea when it comes to working with Python. So far we have just told the program to do one thing at a time. If the circumstances don't line up exactly with the program, you are not going to get any results. This just won't work for some programs, especially if the other people are allowed to pick from a couple of different answers. The decision control structure is the part that allows you to pick a couple different options for Python in case the first choice doesn't work out.

For the most part, these are going to work on a true of false kind of outcome. You will need to figure out which action you want to take and what statements the program should execute if the outcome is either true or false. In Python, any answer that is non-null or non-zero is going to be considered true and the ones that are either null or zero will be considered false. To understand some of these consider the following:

- If statements—the if statement is going to consist of a Boolean expression that is then followed by one or more statements that will be executed if the answer matches up.
- If…else statements—this option will have a statement that will show up if the "if" statement is correct, but there is also another statement that is told to come up if the Boolean expression ends up being false.
- Nested if statements—you can use one if or if else statement inside another one when needed.

Some examples of how this works includes:

```
age = 23
if (age == 23):
        print("The age is 23")
```

print(“Have a good day!”)

when this code comes through it is going to state:

The age is 23

Have a good day!

Using the “if” keyword is going to tell the compiler that what you are writing is a decision control instruction. The condition that is behind the keyword if it is inside the parentheses. The conditions will need to be true if you want the code to come out, but if it is not true, this statement is just going to be ignored in this situation and it is going to move on to the next command that you give.

On the other hand, you could set up an “if…else” clause. This one would show the original message if the conditions were true. But, if the conditions ended up being false, it would end up putting out a second statement instead. This can help to ensure that you are getting the right message across no matter what someone else is sending over and prevent the interpreter from completely ignoring this step.

So the next question that you may be asking is how we are able to tell whether the statement is true or false? You will need to use some of the relational operators to make this happen because it is going to be able to compare the two values to see if they are equal, unequal, or another choice. Some of the options that you can use to see if a statement is right include:

This expression	Is true if
X == y	X is equal to y
X != y	X is not equal to y
X < y	X is less than y
X > y	X is greater than y
X <= y	X is less than or equal to y
X >= y	X is greater than or equal to y

A good example of this would be:

```
age = int(input("Enter your age:"))
if (age <=18):
        print("You are not eligible for voting, try next election!")
print("Program ends")
```

This code will print you out a different example based of the age that is put in. Since it is an if statement only, you are going to get an answer just when the number is 18 or under. Here we will look at the output based on the age of 18 and that of 35.

```
Enter your age :18

You are not eligible for voting, try next election!

Program ends


Enter your age :35

Program ends
```

Now we should take a look at adding in multiple statements within your “if” statement. It is not uncommon to find two or more statements being placed inside an expression and working just fine if everything is satisfied. If these statements are executed, you will need to make sure that you indent them properly. Let’s take a look at how this could work. Make sure to give this a try on your interpreter to get some experience with typing code and how the “if” statements are going to work.

```
bonus = 0.0

currentyear = int(input(“Enter current year:”))

yearofjoining = int(input(“Enter year of joining:”))

yearofservice = currentyear – yearofjoining

if(yearofservice >2):

        bonus = 1500

        print(“Bonus - %d” %bonus)

        print(“Congratulations! We are able to provide you a bonus o %d” %bonus).
```

Now if the “if” statement does end up being higher than two, you will find that the congratulations statement is going to come out. But if the amount is lower than two, you are going to end up with no statements because all of the statements were not met. You can place whatever numbers that you want inside it in order to get it to work for each employee in this option.

The “if-else” Statement

So far we have just been talking about the “if” statements. These are ones that need to be true before you are able to see any of your statements come out. If things come out to be false, there will be no statement at all. Now this does work at times, but in other instances, you may want to have a few different statements come up.

With the “if else” statement, you can pick two, and sometimes more, statements that are going to come up based on the results. If your results end up being true, you will have the first statement come up, but if the results end up being false, you can pick another statement that you would like to show up as well. This ensures that you are getting an answer no matter the result so that the program shows something new.

You just need to make sure that after writing out your “if” statement, you add in the “else” and then put in the statements that you would like to have come up. This can take a bit of time to work, but it opens up so many great options with your code to make sure that it all evens out and looks nice with your code.

The elif Statement

Another option that you can do with your statements is the elif statement. This one is going to give you the option of checking out a few expressions as true, rather than just one expression as true, so that you can execute the whole block of code once just one of the conditions turns up to be true. Of course, doing this is optional, but there is the benefit of being able to have any number of elif statements after the if.

Let’s take a look at what all of this is going to look like in when you write it out in the syntax:

```
if expression1:
statement(s)
elif expression2:
statement(s)
elif expression3:
statement(s)
else:
statement(s)
```

You will then be able to place your information into the parts and get the answer that is listed with each of the parts. Let’s take a look at this syntax expanded out a bit so that you are able to get an idea of how this works:

```
Print(“Let’s enjoy a Pizza! Ok, let’s go inside Pizzahut!”)
print(“Waiter, Please select Pizza of your choice from the menu”)
pizzachoice = int(input(“Please enter your choice of Pizza:”))
if pizzachoice == 1:
        print(‘I want to enjoy a pizza napoletana’)
elif pizzachoice == 2:
        print(‘I want to enjoy a pizza rustica’)
elif pizzachoice == 3:
        print(‘I want to enjoy a pizza capricciosa’)
else:
        print(“Sorry, I do not want any of the listed pizza’s, please bring a Coca Cola
for me.”)
```

you will see the information that is listed after print show up for the first three lines. After the third line, the program should ask you to put in one of four options. Depending on which option you choose, you will be able to see the right answer, either the pizza of your choice or the option to just get a drink. It may look like a bit of a mess at first, but it helps you to get options that the code will be able to understand while returning the right “pizza type” to you in the process.

The ”if” statements are going to provide you with a lot of help when you are looking to make some great things happen in your code. You can also use nested if statements that allow you to add a few of these within each other, adding some more power to your code. Of course, it may take some experience to get this all down, but you are going to be amazed at all that you are able to do when you are working on your codes.

Chapter 4: Loop Control Statements

So far, we have discussed a lot of programs and what you are able to do when you are working in Python, but these are all going to be either decision or sequential control instruction. For the first ones, we were doing calculations that will be carried out in a fixed order while with the second one, the right set of instructions were executed based on the outcome of the conditions that were tested.

There were some limitations just because of the way they are executed and they are only able to perform the exact same series of actions, always in the same way, and they are only able to do it one time.

There are times when you will want to write out a code that can be a bit more complicated. One of these options is for the loop statement. This kind of statement will allow the programmer to execute a statement, or even a group of statements, several times. If you have a statement that you would like to keep coming back in the program, you will want to create your own loop statement to make this happen.

It is possible to use Python in order to hand these loop statements and there are three methods that you can choose in order to make the loop statement happen. These three methods include:

- Nesting loops
- Using a for loop
- Using a while loop

Let's take some time to discuss all of these and figure out when and why you would use each of the methods when creating your new code.

The While Loop

The first type of loop that we will look at is the while loop. This is a good one when you want the code to do something for a fixed number of times. You don't want to have it go on indefinitely, but you do want to have it go for a certain amount of times, such as ten times, before stopping. Let’s take a look at a good example of calculation of interest in the following example to help show how this works.

#calculation of simple interest. Ask user to input principal, rate of interest, number of years.

```
counter = 1
while(counter <= 3):
        principal = int(input(“Enter the principal amount:”))
        numberofyeras = int(input(“Enter the number of years:”))
        rateofinterest = float(input(“Enter the rate of interest:”))
        simpleinterest = principal * numberofyears * rateofinterest/100
        print(“Simple interest = %.2f” %simpleinterest)
        #increase the counter by 1
        counter = counter + 1
        print(“You have calculated simple interest for 3 time!”)
```

The output is going to come out to allow the user to put in the information that they want to compute. It will figure out the interest rates and the amount that is going to be calculated based on the numbers that you provide. It will go on a loop so that they are able to do this three times for use of this. You can set it up to take on more if you would like, but for this one we are just using it three times for simplicity.

The “For” Loop

This is for loops that you will use when you want a piece of code to repeat a certain amount of times. It is a bit different than the one above because it is more of the traditional way to do things, but it can still be useful. This option is going to be a bit different than what you will find in C++ and C.

Rather than this loop giving the user the chance to define the halting condition or the iteration step, Python is going to have the statement iterate over the items in the order that they show up within the statement. An example of this is below:

```
# Measure some strings:
words = [‘apple’, ‘mango’, ‘banana’, ‘orange’]
for w in words:
print(w, len(w))
```

When it goes through the loop with this, you are going to get the four fruit words above come out in the order that they are written. If you want them to be done in a different order, you will need to place them in a different order when you are putting them into the syntax in order to avoid confusion. You will not be able to make changes to the words once they are already in the syntax like above.

If for some reason you want to iterate just over a specific sequence of numbers, using the function range() can really come in handy with this. It is going to generate a whole list containing some of the arithmetic progressions that you are looking to use.

It is also possible to do a nested loop. This is basically just one loop that is inside of another one and it will keep going until both of the programs are done. This can be useful for a number of things that you want to do with your program, but we will show an example that will give out the multiplication table going from 1 to 10:

```
#write a multiplication table from 1 to 10
For x in xrange(1, 11):
        For y in xrange(1, 11):
        Print '%d = %d' % (x, y, x*x)
```

When you got the output of this program, it is going to look similar to this:

```
1*1 = 1

1*2 = 2

1*3 = 3

1*4 = 4
```

All the way up to 1*10 = 2

Then it would move on to do the table by twos such as this:

```
2*1 =2
```

2*2 = 4

And so on until you end up with 10*10 = 100 as your final spot in the sequence.

These loops can help you to get a number of things to show up on your computer, sometimes indefinitely, but for the most part they are going to keep going through the loop only for the amounts of times that you would like it to. You will be able to put these loops in with a bit of practice and there are so many things that you can do.

For example, the simple formula above will give you the whole multiplication table started with 1*1 and ending with 10*10. For such a simple statement, you are getting a great amount of information from it and many of the loops that you work with will be the same.

Chapter 5: Functions

Functions are another important part of learning the Python language. These are basically blocks of code that are self-contained and will be able to perform some kind of coherent task within your code. When you take the time to define a function, you will be able to specific the name of the function as well as the sequence of the statements. You will then be able to call up the function using its name. There are two types of functions that we are going to use here and we will discuss them below.

User defined functions

With this one, you will be able to define the functions that you are using. They are going to have pretty much the same rules that you found with variable names. This means that using underscore characters, numbers, and letters will work great, but you should never use a number for the first character in the function. You can't use your keywords as the name of your function and you should be careful about having a function and a variable that have the same name.

The empty parentheses that you will see after the name will show that the function isn't going to take on any arguments. The first line is going to be known as the header while the rest of the function is known as the body. You need to make sure that the header ends with a colon and that you indent in the body so that the interpreter knows what you are doing. A good example of the syntax or a function includes:

```
def functionname(arg1, arg2, arg3):
        '"docstring of the function i.e., a brief introduction about the function"
        #code inside the function
        Return[value]
        #a function may not have any arguments or parameters like
        def functionname():
        #code inside the function
```

Using the parameters of a function can be valid in any of the data types that you are using with Python, whether you are dealing with user-defined classes, dictionary, tuple, list, float, and int. Let's take a look at some of the different parts of this function so you can understand the importance of each one and how they work.

Docstring function

Now let's take a look at the docstring. The first string that you see right after the header in a function is going to be called the docstring, something that is short for documentation string. It is going to be used in the code in order to explain what the function does. This is an optional part, but it is a good practice to get into. If you are planning on having this go across a few lines, you should consider doing the triple quotes in order to tell the computer what you are doing.

The return statement

The function is always going to give back a value. The return statement is going to be used as an exit function and will go back to the place where it was called from. This statement is allowed to contain expressions that it can evaluate before giving out a value. If there are no expressions within the statement, or the return statement is not even present inside of the function, you will find that the return you get is the None object.

Lifetime and scope variables

Another thing that you can learn about the Python language is about lifetime and scope variables. The scope variable is the part that is going to be recognized and visible. The variables and the parameters that are inside the function are not going to be visible from the outside eye (those who are looking at the code when it is being executed) but they will have a local scope that can show up.

On the other hand, a lifetime is the period of time that the variable is going to exist in the memory. The lifetime of most variables inside functions will be as long as your function executes. After the function returns the value, these are going to be destroyed so that you can put in no inputs and get different results. For example, if you input certain numbers and your return ends up being 5, the next time you are able to put in different information and you may get a 6. These will delete after each function has run through so that you can get new results if new information is put inside.

Pass by references

In Python, all of the parameters that you put in are going to be passed by reference. This means that the address of the location in the memory is going to be referenced to the memory location. So if you are going to change what your parameter is able to refer to inside the function, this same change is then going to reflect back when you are doing the calling function.

Flow of execution

The execution of the statement is always going to start right at the first statement that is inside of the programs. Your statements are going to be done just one at a time to avoid confusion and make sure that the program is running as smoothly as it should. The execution is also going to happen going from top to bottom so you need to make sure that you are putting them in properly. So you must make sure that you are defining the functions before you decide to call them up to get the right order.

Anonymous functions

Python allows the programmer to create anonymous functions. These are functions that are not going to be bound to a name at run time and you are going to need to use a construct that is known as "lambda." This operator is a way to create these functions that don't have a name. basically you will want to create these when the functions are considered throw away, or they are just needed right where they have been created. There are a few functions that Lambda functions will work with including reduce(), map(), and filter().

The syntax that you will want to use with the lambda function is:

lambda argument_list: expression.

The reduce, filter, and map function

We have talked about a number of functions so far in this book, but it time to look at a few that are going to help you to get more out of the code that you are writing. Particularly, we are going to take a look at the map, filter, and reduce functions, as well as list comprehension to help you get started.

Map function

The map() function is the one that will apply to ever member inside of an iterable, or those inside a list. For the most part, you would use the anonymous inline function to make this work, though this is one o them that you are able to use inside of any of your functions. So i you are trying to work on a list, this would be a good one for you to use in your computer code.

Filter function

Next on the list is the filter function. Just like you would guess from the name, the filter is able to extract the elements in the sequence for which the function returns a true. The rest would be ignored. This means that you will want to do a sequence that would be either true or false in order to get this one to work. When the sequence has a few options that are true, the filter function will pick out those ones. On the other hand, if all of them, or at least some of the sequence points, are false, you will find that these are not filtered out.

The reduce function

This one is a bit unique. It is going to take several values to the sequence and will work it until you end up with just one value, rather than the large list that you have. It is going to work from let to right in order to get the answers that you are looking for. Depending on the length of your sequence, you may have to do some work with this to get it all done.

Let's look at the numbers 1, 2, 3, and 4. The reduce function is going to take these four numbers and turn them into one. It will do this by adding the 1 and the 2 together to get 3, then adding the 3 and the 3 together to get six, and then adding the 6 and the 4 together to get 10. In this answer you will get the single value of 10.

A good syntax to show how this works is

```
from functools import reduce
results = reduce( (lambda x, y: x +y), [1, 2, 3, 4])
print(result)
```

List comprehension

This is a great way to create some lists inside of Python. Some of the common applications will use your elements to make the new list and others will create a subsequence of these elements when they satisfy certain conditions. The list comprehension feature can actually be a substitute to using the lambda function as well as the other functions that we just talked about. This is because the list function is easier to work with and understand. The syntax of using the list comprehension is

[expression-involving-loop-variable ***for*** *loop-variable* ***in*** *sequence]*

This is going to step right over all the elements of the sequence so that you are able to set up the loop variable for ever element one at a time and then it completely eliminates what you needed the lambda function for in the first place.

Conclusion

Working in Python can be one of the best programming languages for you to choose. It is simple to use for even the beginner, but it has the right power behind it to make it a great programming language even if you are more of an advanced programmer in the process. There are just so many things that you are able to do with the Python program, and since you are able to mix it in with some of the other programming languages, there is almost nothing that you can't do with Python on your side.

It is not a problem if you are really limited on what you are able to do when using a programming language. Python is a great way for you to use in order to get familiar and to do some really amazing things without having to get scared at how all the code will look. For some people, half the fear of using a programming language is the fact that it is hard to take a look at with all the brackets and the other issues. But this is not an issue when it comes to using Python because the language has been cleaned up to help everyone read and look at it together.

This guidebook is going to have all the tools that you need to hit the more advanced parts of Python. Whether you are looking at this book because you have a bit of experience using Python and you want to do a few things that are more advanced, or you are starting out as a beginner, you are sure to find the answers that you need in no time. So take a look through this guidebook and find out everything that you need to know to get some great codes while using the Python programming.