

PYTHON PROGRAMMING LANGUAGE FOR BEGINNERS

The First Real Guide For Beginners Towards Machine Learning
And Artificial Intelligence. Learn How To Develop Your First
Web App In Just 7 Days With



OLIVER R. SIMPSON

Python Programming Language For Beginners

*The First Real Guide For Beginners Towards
Machine Learning And Artificial Intelligence.
Learn How To Develop Your First Web App In
Just 7 Days With Django!*

By

Oliver R. Simpson

□ Copyright 2019 by _____ - All rights reserved.

This eBook is provided with the sole purpose of providing relevant information on a specific topic for which every reasonable effort has been made to ensure that it is both accurate and reasonable. Nevertheless, by purchasing this eBook you consent to the fact that the author, as well as the publisher, are in no way experts on the topics contained herein, regardless of any claims as such that may be made within. As such, any suggestions or recommendations that are made within are done so purely for entertainment value. It is recommended that you always consult a professional prior to undertaking any of the advice or techniques discussed within.

This is a legally binding declaration that is considered both valid and fair by both the Committee of Publishers Association and the American Bar Association and should be considered as legally binding within the United States.

The reproduction, transmission, and duplication of any of the content found herein, including any specific or extended information will be done as an illegal act regardless of the end form the information ultimately takes. This includes copied versions of the work both physical, digital and audio unless express consent of the Publisher is provided beforehand. Any additional rights reserved.

Furthermore, the information that can be found within the pages described forthwith shall be considered both accurate and truthful when it comes to the recounting of facts. As such, any use, correct or incorrect, of the provided information will render the Publisher free of responsibility as to the actions taken outside of their direct purview. Regardless, there are zero scenarios where the original author or the Publisher can be deemed liable in any fashion for any damages or hardships that may result from any of the information discussed herein.

Additionally, the information in the following pages is intended only for informational purposes and should thus be thought of as universal. As befitting its nature, it is presented without assurance regarding its prolonged validity or interim quality. Trademarks that are mentioned are done without written consent and can in no way be considered an endorsement from the trademark holder

Table of Contents

[Introduction](#)

[Chapter 1: History of Artificial Intelligence](#)

[Chapter 2: Introduction to Python Programming](#)

[Chapter 3: Variables and Data Types](#)

[Chapter 4: Conditional Execution](#)

[Chapter 5: Python Functions](#)

[Chapter 6: Python Operators](#)

[Chapter 7: File Handling](#)

[Chapter 8: Dictionaries](#)

[Chapter 9: Object-Oriented Programming](#)

[Chapter 10: Inheritance](#)

[Chapter 11: Introduction to Django Framework](#)

[Chapter 12: Django Installation](#)

[Chapter 13: MVC Pattern](#)

[Chapter 14: Create, Install, Deploy First Django App](#)

[Chapter 15: Tips for Django and Python Beginners](#)

[Conclusion](#)

Introduction

Whether you're getting ready to build your development skills or get a new start as a software programmer, mastering Python is critical to your success.

As you know, there are many computer programming languages. You'll perhaps need more than one life to master them all. So why do we suggest you go with Python?

- It's future proof. This language isn't vanishing soon, especially with the increasing demand for Data Scientists.
- It's versatile.
- It's universal. All modern technologies can support Python code.
- It is quite easy to learn. In fact, an experienced programmer in any language can master Python quickly. It is so easy for beginners to use and learn.

The syntax of Python is clear; the language is high level and boasts of higher readability than most other languages. Plus, it's straightforward to pinpoint and correct Python errors, which means a lot to beginners.

By reading this book, you're off to a great start. This book is designed to ease your way into Python programming world. So, let's jump in.



Chapter 1: History of Artificial Intelligence

We start by exploring early efforts to define artificial intelligence.

The history of artificial intelligence may sound like a deep and impenetrable subject for individuals who are not well-acquainted in computer science and its related fields.

Regardless of how mysterious and untouchable artificial intelligence may look, when it's broken down, it becomes easy to understand than you might imagine.

So, what is artificial intelligence, otherwise referred to as "AI"?

AI is a branch of computer science that concentrates on non-human intelligence or machine-driven intelligence.

AI relies on the concept that human thought functionality can be replicated.

Early thinkers pushed the concept of artificial intelligence throughout the 1700s and beyond. During this period, it became more tangible. Philosophers imagined how the idea of human thinking could be artificially computerized and transformed by intelligent machines.

The human thinking that generated interest in AI eventually resulted in the invention of the programmable digital computer in the 1940s. This particular discovery finally pushed scientists to proceed with the concept of building an "electronic brain," or an artificially intelligent being.

Besides that, mathematician Alan Turing, had developed a test that ascertained the potential of a machine to emulate human actions to the degree that was indifferent from human actions.

From the 1950s, many theorists, logicians, programmers broadened the modern mastery of artificial intelligence as a whole.

During this period, intelligence was viewed as a product of "logical" and "symbolic" reasoning. The reasoning was executed by computers using search algorithms. The focus during this time was to replicate human intelligence by solving simple games and proving theorems. Soon, it became obvious that these algorithms could not be used to find solutions to problems such as the movement of a robot in an unknown

room. Extensive knowledge of the real world would have been required to avoid a “combinatorial” explosion of the problem to solve.

Come in the 80s; it was pragmatically accepted to restrict the AI scope to specific tasks like the replication of intelligent decision making for the medical diagnosis of particular pathologies. This was the time of “expert systems”, capable of successfully replicating the intelligence of a human specialist in small and well-defined industries.

Similarly, it became apparent that some intelligent actions such as recognition of written text, could not be achieved with an algorithm designed with a sequence of instructions previously set. Instead, it was possible to gather numerous examples of the objects to be identified and using algorithms that could master the essential characteristics of these objects.

It was the beginning of what we now refer to as “machine learning.” The computer learning steps could be defined as a mathematical optimization problem and described with probabilistic and statistic models. Some of the learning algorithms that were used to replicate the human brain were referred to as “artificial neural networks.” During the first four decades, AI has moved through moments of Euphoria, followed by times of unfulfilled expectations. In the early 2000s, the increasing emphasis on specific problems and the increase of investments resulted in the first historical accomplishments. In some functions, AI systems attained higher performances than humans.

What is an Intelligent Machine?

After the discovery of computers, the debate on the nature of the intelligence that had involved philosophers for thousands of years took the form of the title of this section. Already some years before the Dartmouth workshop, Alan Turing had asked himself this question, and while seeking a solution, he had suggested a “test”, known as “Turing Test” to determine the machine intelligence. Suppose a human and a computer that claims to be intelligent are put together in one room. Another human, a “judge,” can speak with them in written and spoken form, but without seeing them. The judge interrogates the two interlocutors and decides who is human and who is not. The mistake of the judge is proof of the machine’s intelligence. It is proof that the machine is inseparable from an intelligent human.

This definition of intelligence addresses many of the ambiguities that can be experienced in defining what intelligence is. We don't pretend that the computer acts like us, just like we did not ask that planes fly like birds. We're satisfied that the computer cannot be differentiated from a human for a sequence of tasks that demand we call for intelligence. The complexity and broadness of the tasks required to distinguish what is known as "Narrow AI" from the "General AI" of future systems that should indicate an intelligence on a human level, or higher, for a wide variety of tasks.

The Current Nature of Artificial Intelligence

First, you need to understand that the "real" AI doesn't exist at the moment. A real AI refers to AI systems that can solve all human-level tasks with the same proficiency.

So far, the current systems are referred to as "narrow" AI. This means they can only generate useful results within a small domain. Even with that, the past few years have seen important ingredients compiling to push AI beyond early adopters to an expansive marketplace. Today, newspapers and magazines are full of articles about the current progress in AI and machine learning technologies.

Two reports predicted that in the next few decades, AI will rise to become the largest commercial opportunity for firms and nations. Progress in AI has the potential to boost global GDP to 14% between now and 2030. This is equivalent to an additional \$14-15% trillion contributions to development in productivity.

As time goes, AI may become a transformative technology such as the steam engine, electricity, and the internet. Acquisition of AI in the market place will take a general S curve pattern with a slow start in the early period.

While still in the early stages, AI is already delivering significant value for those who've embraced the technology. Seventy-eight percent accept to attain significant value, while only 1% agree that they've seen none. Across business processes, the highest value was in risk management and manufacturing.

Some of the Recent Progress in AI

Text Generation

When it comes to text generation, there is the OpenAI's GPT-2 Model which can produce realistic text. While there is a lot of hype and ethical discussion around this model, but below is a list of commercial applications associated with text generation:

Fan Fiction Generation: Provide some context to a model and let it create a whole fiction around that, or you can apply ideas and continuously guide it to establish a story you prefer within no time.

Automated News Generation: Normally, the news is created around something that takes place on social networks. For example, Elon Musk sends a tweet, and there is a good chance it will become news, and numerous articles will be written about it. You can adopt real-time social media data to create automated news generation.

Content Generation: This model can be used to collect an extensive amount of content across the internet on your blog. A single article is normally released across multiple sites, applies that context, and allows the model to rewrite the blog for you.

Personalized Articles: You read news articles every day but what if the article you read is personalized to your reading style. This has a huge problem.

Automated Storylines for Games: This may appear futuristic, but it could be possible if you train the model on a big dataset of a universe. For example, the MCU model may release custom story-lines, and it might be applied for a separate game-ending for everyone who plays it!

Image Generation: Images showcase a large portion of the internet. From graphics designers to concept artists, many people use images to communicate and generate their livelihood.

Face Generation: This application may already be affecting the game industry because you can use it to build realistic faces of non-existent people. You can still use it to create faces of children and the entire family trees very quickly or in real-time based on the game store-line.

Anime Face Generation: Anime is an enormous industry in Japan but it consumes a lot of time to design the face of a character. By using GANs (Generated Adversarial Networks), it is possible to generate a large number of them with a click of a button.

Photography: Recovering broken image, superzoom, noise removal, and night time realistic photos are some of the developments that are already being embedded in

modern smartphones. This AI allows you to become an expert in photo editing in no time.

Audio Generation

Audio is another form of content we use online daily. Whether it's speaking or music from your beloved playlist. Here's a shortlist of advancements:

- **Music Composition & Synthesis:** If you have a music sheet, you can create music with the help of numerous instruments, but what if AI can help play those musical instruments. You can go further by letting AI compose music as long as there are certain constraints.
- **Speech Synthesis:** Realistic speech synthesis has been an issue but with recent breakthroughs, you can synthesis realistic speech. There is also research on changing the voice of one individual to another or even editing what one individual said.

Animation Generation

Video is the most popular type of content used in today's internet. YouTube is a popular platform, and many YouTubers earn online by generating great content.

- **Video Style Transfer:** You can change one type of video to a new kind of video, or if you're an artist or an animator, you create simple annotations and let AI generate some ideas.
- **Realistic Animation:** AI is also powerful at animating objects. This can be directly used in the 3D animation sector to animate complex objects without breaking a sweat. It's also relevant in-game animations.
- **3D Modelling:** GANs can be used to build a massive variety of 3d models, and it is essential to create a large number of synthetic characters and buildings.

From self-driving cars to healthcare diagnosis, stock market analysis, and small things like editing files, there's no question that artificial intelligence will play a significant role in day-to-day life as the future opens.

It may be a gradual change as the opportunities of AI grow to the challenges that humans present, but gradual, the change is coming. As that change happens, it will likely prepare a totally new era of technology and human growth in the process.

Machine Learning

In the past few years, Machine Learning (ML) has been a buzzword.

Machine Learning has a range application from automating mundane tasks to generating intelligent insights, industries in every field strive to gain from it. You might already be using a device that relies on Machine Learning. For example, a wearable fitness tracker such as Fitbit, or an intelligent home assistant. However, there are many more examples of ML in use.

Overview of the history of ML

ML is a vital element of modern business and research. It has algorithms and neural network models to help computer systems progressively changing their performance. Algorithms of ML develop a mathematical model using sample data to make training decisions without being programmed to make those decisions.

In 1952, Arthur Samuel working at IBM, built a chess program. The program could observe positions and learn an implicit model that provides better moves for the latter cases. Samuel played different games with the program and noted that the program could become better as time goes.

With that program, Samuel coined the general providence defining machines cannot extend passed the written codes and learn patterns the way human beings do. Therefore, he coined the term "machine learning," which he described as:

A field of study that provides a computer with the ability without being explicitly controlled.



Chapter 2: Introduction to Python Programming

Python programming language refers to an open-source, advanced language created by Van Rossum in the 1980s. Currently, the language is maintained by Python Software Foundation.

Python is a powerful language that you can use to build games, develop web applications, and create GUIs.

This is a high-level language. To read and write codes in the Python language is similar to reading and writing regular English statements. Since they're not written in machine-readable form, Python programs must be processed before machines can run them.

Python is a language that's interpreted. Python is an object-oriented language that lets users control data structures to build and run programs. Nearly everything in the Python language is a class. All objects, functions, classes, data types, and methods assume an equal position in the Python language.

Programming languages are built to fulfill the desires of programmers and users for an effective tool to create applications that affect lives, economy, lifestyles, and society. They make lives enjoyable by boosting productivity, improving communication, and enhancing efficiency. Languages die and become extinct when they fail to meet expectations and are replaced and replaced by more forceful language. Python is a programming language that has passed the test of time and has remained significant across industries, and businesses and among programmers. It is a thriving, and critical language that is recommended as a first programming language for those who want to jump into and experience programming.

The Ups of Using Python Language

Below are reasons why you would prefer to learn and use Python over other languages:

Readability

Programs written in Python use simple, clear, and short instructions that are easy to read even by those who have no significant programming knowledge. Therefore, Python programs are more comfortable to debug, improve and maintain.

The Least Learning Time

It's easy to learn Python. Most people find Python a great first language for mastering programming because it has shorter codes and simple syntax.

Expands Different Platforms

Python can work on Mac OS X, Windows, and other operating systems including small devices. It also can work on microcontrollers used in toys, remote controls, appliances, and other similar devices.

Installing Python

How to Install Python on Windows

To install Python on a Windows laptop, you need to first download the installation package of your preferred choice from the Python main website.

On the Python main website, you will be requested to download the latest version of Python.

Also, if you're searching for a particular release, you can scroll down the page to select download links for earlier versions.

However, it is better to download the latest version which, at the time of this writing, is Python 3.7.4. But your preference should always depend on what would be most usable for your project.

How to Install Python on Mac

If you're using Mac, you can still download the installation package from the Python main website.

How to Run the Installation File

Once you're done with the download, you can move on to installation by clicking on the downloaded .exe file. The standard installation will comprise of IDLE, documentation, and pip.

Working with Python

Python is a dynamic and flexible language that you can use in different methods. You can use it interactively when you want to test code or a statement on a line-by-line basis or when you're searching its features. You can use it in script mode or when you want to interpret a whole file of statements or application program.

Command-line Interaction

The command line is the most common way to deal with Python. It is easy to visualize how Python works because it responds to every command entered on the >>> prompt. It might not be preferred interaction with Python, but it's the easiest way to discover how Python operates.

Starting Python

There are different styles to access Python's command line depending on the type of operating system on your machine.

- If you are going to use Windows operating system, then you can navigate to the start menu and click the Python command line.
- For those working on Linux, Mac OS, and Unix, you will run the Terminal Tool.

Computer programming languages has commands to direct the computer on how to execute instructions. So, when you want to perform something in Python language,

then you will need to write down several commands. After that, Python translates the commands so that it can be executed by the computer.

How to exit Python

You exit the Python language by typing the following commands:

`quit ()`

`exit ()`

Control-Z, and then press enter

Integrated Development Environment (IDLE)

This tool is found in the Python's installation package. However, if you wish you can search for complex third-party IDLEs.

The IDLE has an excellent platform to build your code and work well with Python. The IDLE is found in the same folder as the command line icon. Click the IDLE icon to take you to the Python Shell window.

The Python Shell Window

This window has several dropdown menus and a `>>>` prompt. Once you sign in the Python Shell window, you can begin to enter your statements for execution. However, the IDLE's editing menu allows you to go back to the previous commands. The Python Shell Window features different items on the menu like Edit, File, and Help.

The Shell and Debug menus have different functions which you would find important when creating complex programs.

The Shell menu will allow you to access the most recent reset or even restart the shell.

The Debug Menu option has relevant menu items for exploring the source file of an exception and selecting the error line. The debugger option will provide an interactive debugger window that will support the programming running. The stack viewer option describes the current Python stack through a new window.

The Option will help you to configure the IDLE to complete Python working preferences. The Help option features documentation and Python Help window.

The File Window

The items found on the File menu enables you to build a new file, create a module, create an old file, and save our session. When you click on the 'New File' option, you will be directed to a new window, a simple and normal text editor where you can type or edit your code. Initially, this file window has the name 'untitled,' but on saving your code, its name changes.

The File window's menu bar differs slightly from the Shell Window. It doesn't have the 'Shell' and 'Debug' menu located in the Shell Window, but it provides two new lists: the Run and the Format menu. When you decide to run your code on the file window, you can see the output on the Shell Window.

The Script Mode

When you work in script mode, you don't automatically see the results the way you would in an interactive mood. To see an outcome from a script, you'll have to run the script and trigger the print () function inside your code.

Python Syntax

When it comes to Python Syntax, it revolves around how human users and the system should write and interpret a Python program. If you want to write and run your application in Python, you have to familiarize yourself with the syntax.

Keywords

Keywords in Python language refers to reserved words that need to be used as variable, function name, constant, or identifiers in your code. Be keen on the following keywords if you don't want to experience errors when you implement your program:

break, def, lambda, is, not, tray, with, yield, exec, del, else, and, continue, except, from, finally, print, while, return, try, pass, global, import, assert, class

Python Identifiers

A Python identifier refers to a name assigned to a function, variable, module, or other objects that you'll be using in your Python program. Any entity you'll use in Python should be correctly named or identified as it will form part of your program.

Below are naming conventions in Python that you must know:

- An identifier can be a mix of uppercase letters, underscores, lowercase letters, and digits between (0-9). Therefore, the following are proper identifiers: `my_variable`, `myClass`, `var_3`, and `print_hello_world`.
- Unique characters like `@`, `%`, and `$` are not approved within identifiers.
- An identifier should not start with a number. As a result, `2variable`, is not correct, but `variable2` is acceptable.
- Python is a very case-sensitive language and this should extend to identifiers.
- You cannot use Python keywords as identifiers.
- You can apply underscores to separate multiple words in your identifier.
- The class identifiers start with an uppercase letter, but the whole identifiers start in lowercase.

You should always select identifiers that are realistic even after a long gap. Therefore, although it's easy to set your variable to `c = 2`, you may find it more useful for future reference if you go for a longer and proper variable name such as `num = 2`.

Use Quotations

Python supports the use of quotation marks to display string literals. You can use triple quotes, double, single, but you need to start and end the string with the same kind. You may use the triple quotes when your chain runs across different lines.

Python Statements

Statements contain instructions that a Python interpreter can run. When you allocate value to a variable, for instance, `my_variable = "dog"`, you'll be creating an assignment statement. An assignment statement can be as short as `c = 3`. There are other types of statements in Python such as while statements, if statements, and for statements.

Indentation

In most programming languages like Java, there are braces to define blocks of code. However, Python programs are different because they are formatted using indentation. Python blocks of code are indented. As a result, codes written in Python are easy to understand and read.

While you indent blocks of code, make sure that the indent space is constant. If you use IDEs to type your code, Python has a standard rule that guides on the application of indentation. In general, indentation should be around 4 spaces to the right.

Comments

When building a program, you will come to realize that it's a good practice to include some description to your code. This is the point where comments become important. A comment is a way to ensure that your code is easily understood by other programmers. Also, when you revisit your code after some few months, you will not be stranded on what exactly a given line of code was doing. Comments in Python are written by using the hash symbol. The hash symbol enables the Python interpreter to ignore the comment when executing the code.



Chapter 3: Variables and Data Types

Python programs are saved with a file extension of .py. In other words, for the hello program, the file name should be hello_world.py. This automatically shows the file contains Python programs. Your editor then executes the file using the Python interpreter, which reads through the program and determines the meaning of every word in the program. For example, when the interpreter comes across the word print, it outputs to the screen whatever is contained in the parentheses.

While you write your programs, your editor highlights different sections of your program in different ways. For example, it understands that print is the name of a function and outputs that word in blue. It realizes that “Hello, World.” is not Python code and outputs that phrase in orange. This property is known as syntax highlighting and is crucial as you begin to write your programs.

Variables

A variable is like a container that holds values that you can access. It is a way of pointing a memory location that a given program uses. You use variables to help the computer save or retrieve data to and from the memory location.

Python differs from other languages when it comes to variables. Languages such as Java, C++, or C link a variable to a specific data type. In other words, it can only store a given data type. For that reason, when a variable is of type integer, you can only save integers inside that variable when executing the program.

Python is very flexible when handling variables. As a result, Python lets you change the value and data type during program execution.

Now, let us run a hello world program using a variable. First, you will need to add a new line at the start of the file and change the second line. Below is how the program will appear:

```
text = "Hello world!"
```

```
print (text)
```


If you run the following code snippet in your Python editor, you need to see an output as: Hello world!

In this program, the variable text has been used to store the value, which is the information related to that variable. In the following example, the value is the text "Hello world!"

Adding a variable increase, the work for the Python interpreter. Once it processes the first line, it connects the text "Hello, World!" with the variable text. When it goes to the second line, it prints the value connected with a message to the screen.

Let's expand on the following program by modifying it to display a second message. Include the following lines of code:

```
text = "Hello, World!"  
  
print(text)  
  
text= "Hello Python beginner!"  
  
print(text)
```

Now when you enter the above code in your Python editor and proceed to run, you will see two lines of output:

Hello, World!

Hello Python beginner!

Still, if you want, you can modify the value of a variable in your program at any given time, and Python will always remember its current value.

How to Name and Use Variables

When you use variables in Python, you need to respect a few rules and guidelines. If you break the rules, you will encounter an error. Other instructions enable you to write a code that is simple to read and understand.

Some of these rules include:

- Variable names can only contain underscores, letters, and numbers. A variable name can start with a letter or an underscore, but not with a number. For example, it's possible to call a variable message_1.

- Spaces aren't allowed within variable names, but underscores can be used to separate words inside variable names. For example, `num_year` but `num year` will lead to errors.
- Don't use Python keywords and function names as variable names. In other words, don't use words that Python has reserved for a given programmatic reason such as the word `print`.
- Variable names need to be short but descriptive. For instance, the name `is` is better than `n`, `num_year` is better than `n_year`.
- Be keen when you use lowercase letter `l` and uppercase letter `O` because it's easy to confuse with the numbers `1` and `0`.

It might take some practice to master how to build proper variable names, particularly as you become more exciting and advanced. While you write more programs and begin to read through other people's code, you'll become better at creating meaningful names.

Last but not least, computers are firm, but they don't adhere to good and bad spelling. This means you don't need to follow English and grammar rules when you try to create variable names.

Most errors in programming include single-character typos in a single line. If you spend a long time searching for one of these errors, note that you're in a good company. Most experienced and talented programmers spend countless hours searching for these types of errors.

The best way to learn new programming ideas is to try them in your programs. If you get stuck while doing an exercise, try to do something different for a while.

Strings

Python deals with different data types to support the needs of programmers and application developers for useful data. These comprise of Booleans, time, numbers,

date, and time.

A string comprises of a sequence of Unicode characters that could be a combination of letters, special symbols, and numbers. To define a Python string, you can include a string in a single or double-quotes.

```
>>> stringone = 'I am enclosed in single quotes.'
>>> stringtwo = "I am enclosed in double quotes."
```

A string is a Python data type.

You can define a string as a “sequence of characters”.

Characters are organized in a certain position inside a String. For example:

Greeting = “hello”

In the above string, ‘h’ is the first character inside the string, followed by ‘e’ etc.

The following string has same characters but they are placed in a separate position:

Greetings = “loleh”

How to Access the Individual Characters Inside a String

Certain programming tasks will require that you access individual characters stored inside a string.

You can separate individual characters inside a string using a for loop. The featured variable inside your loop will assume every character in your string one at time. For instance: for c in “Maria”:

```
print (c)
```

```
>> M
```

```
>> a
```

```
>> r
```

```
>> i
```

```
>> a
```

Making use of individual characters in a string for calculations

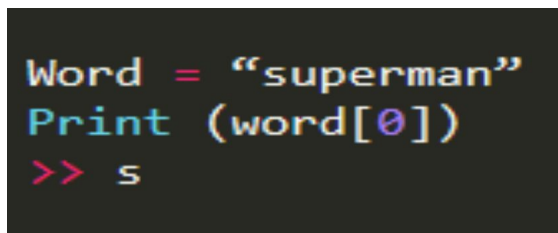
Now that you can iterate over a string it is easy to imagine how you can analyze the details of a string.

String Indexing

Indexing notation is another method to analyze a string.

Indexing refers to a means of referencing individual elements inside a string depending on their position.

You can determine an index by applying an integer value within a pair of square brackets just after the variable:



```
Word = "superman"  
Print (word[0])  
>> s
```

String indexes start with zero.

Iterating a String Using Indexing

You can apply a “for” loop to go over the characters in a string. For example:

```
word = "hello"
```

```
for c in word:
```

```
    print (c)
```

But, if you want to go over the string by using indexing, then you will need to generate a range of numbers that correlate to the indexes you want to visit. For example:

```
word = "hello"
```

```
for i in range(0, 5):
```

```
print (word[i])
```

String Immutability

Strings are an example of immutable data type. In other words, they cannot be modified once they are created.

What really goes on behind the scene is that Python defines a different string inside the computer memory and references that string rather than the previous one.

```
name = 'superman'
```



```
name = 'wonder woman'
```



In other words, it's impossible to modify single characters within a string by applying notation. You may trigger an exception.

```
word = "Superman"
```

```
word[0] = "X"
```

```
# exception!
```

Changing a String

To make changes to a string you will require to create a new String variable.

This new variable can then store the modified version of the string.

For instance, let's take the following programming challenge:

“Write a program that replaces all vowels in a String with the underscore character.”

At first, you may assume that you could simply apply String indexing to modify the characters in question. For instance:

```
word = "hello"
```

```
word[1] = "_"
```

```
word[4] = "_"
```

But since strings are immutable, this will not work!

Instead, you need to define a new string variable to store the changes and then apply a loop to check every character within a sequence to “create” the new String. For example:

```
word = "hello"
```

```
newword = ""
```

```
for c in word:
```

```
    if c == 'e' or c == 'o':
```

```
        newword += "_"
```

```
    else:
```

```
        newword += c
```

String Functions

So far you know several functions that you can use with strings.

For instance, the `len()` function can count the number of characters in a string.

Besides that, there are other functions that you can apply when you work with Strings to simplify things for you as a developer.

Determining the Largest and Smallest Character within a String

You can use Python built in functions to calculate the largest and smallest characters inside a string. For example:

```
a = max("python")
b = min("python")
print ("max:", a)
print ("min:", b)
>> y
>> h
```

String Slicing Notation

For Python to cut a string you must use a bracket symbol to define the index range of the characters you want.

`substring = bigstring[start:end:step]` The standard notation for the following syntax includes:

You need to include at least an index value.

Substring properties comprises all the characters at the start value and continue to the ending value.

Eliminating a starting index value will trigger Python to assume that you want to begin at the start of the string or you want to proceed slicing the last section of the string. This should resemble the function range.

String Operators

Now that you know the "+" and "*" operators can be applied together with a String.

The "+" operator can be applied to "concatenate" two strings together.

The "*" operator can be applied to repeat a String a specific number of times.

Testing String Using "in" and "not in"

The “in” operator represents a Boolean operator that you can apply to determine whether a substring exists within another string. For example:

```
word = "Jackson James John Chris Tom"
if "Chris" in word:
    print ("found him!")
else:
    print ("can't find Chris")
```

When you define an expression using the “in” operator the result will compute to a Boolean.

String Method

A “method” refers to a function that is member to an “object”, and completes operation on that object. For example:

We haven’t discussed objects much in class, but strings can be a great introduction to methods and objects.

The standard syntax includes:

stringvariable.method(arguments)

String Testing

String testing methods enable you to determine whether a specific pattern is available within a given string variable. For example:

```
word = '1234'
if word.isdigit() == True:
    print ("All chars in word are digits!")
else:
    print ("Not all chars in word are digits!")
```

In this example, the “isdigit” method is used to call “word”. The following method will evaluate to True in case all characters inside a string are numeric digits, and False if not.

String Modification Methods

Keep in mind that strings are immutable and cannot be directly changed.

That said, they do have a number of “modification” methods that return a new copy of the string that reflects a given change.

```
word = "Craig"  
newword = word.lower()  
print (word)  
>> craig
```

Searching and Replacing

Programs usually need to search and substitute functions on data, same as “find and replace” functionality that is found in your word processor.

Strings can be searched for strings and substrings with the help of find () or index () methods.

The find method returns -1 in case the substring isn’t found.

The index () returns an error in case the substring isn’t found.

Searching a Specific Section of a String

Both the find () and index () methods search from the start of the string, and determine the occurrence of the substring.

You can also define a start and ending index to only check a portion of the string.

How to Search Backwards Through a String

Both the rindex () and rfind() methods work the same as find() and index(), except the string is searched backwards from the end of the string.

Centering Text

s.center(w, [pad]) centers the text within a field of width w.

The optional pad character will be included to the ends of the string.

Concatenation

You can concatenate strings using the + operator.

```
>>> 'welcome' + 'home'
```

```
'welcomehome'
```

Join Method

Strings too have a join () method.

Multiplications

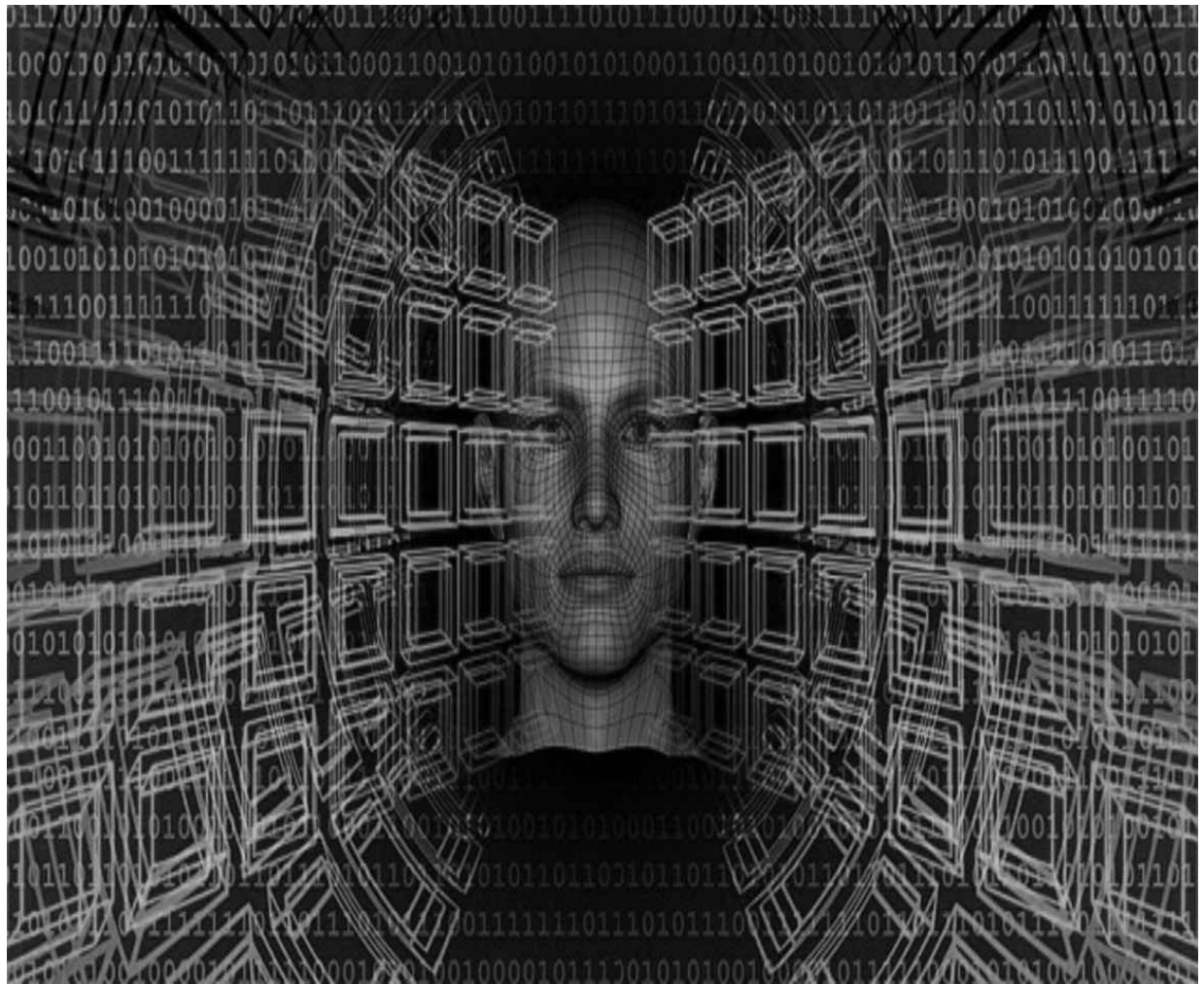
Strings can even be multiplied by integers.

```
>>> 'ship' * 5
```

Methods of Extracting Information About a String

s.count(ss)- this counts the number of times of substring ss

s.endswith(sfx)- this confirms whether a string ends with the substring sfx.



Chapter 4: Conditional Execution

There are certain conditions in life when decisions become inevitable. This is not different when it comes to programming. It is the same for every program, which has to evaluate relevant problem. There is no way to program without applying branches in the program flow.

When it comes to programming and scripting language, conditional statements are used to compute various calculations depending on whether a given condition evaluates to true or false.

The condition normally has arithmetic and comparison expressions. The expressions are computed to the Boolean Values False or True. The statements responsible for decision are referred to as conditional statements, alternatively they are also known as conditional constructs.

The if-then syntax is sometimes referred to as the if-then-else. This is common in most programming languages, but the syntax differs from language to language.

Conditional statements are found in every programming language. Conditional statements allow you to have a code that may run and at other times not run, depending on the input of the program.

Once you fully run each statement of a program, moving from top to bottom with every line executed, you aren't asking the program to execute certain conditions. By applying conditional statements, programs can tell whether a given condition is being met and then be informed what to do next.

Here are some examples where conditional statements are important:

- If a student attains over 65% on her test, print that her grade passes, if not, print that her grade fails.
- If they purchase 10 mangoes or more, determine the discount of 5%. If they purchase less, then don't.
- If he has money in his account, compute the interest, if he doesn't, charge a certain fee.

While evaluating conditions and allocating code to run regardless of whether those conditions are met or not, then you are writing conditional code.

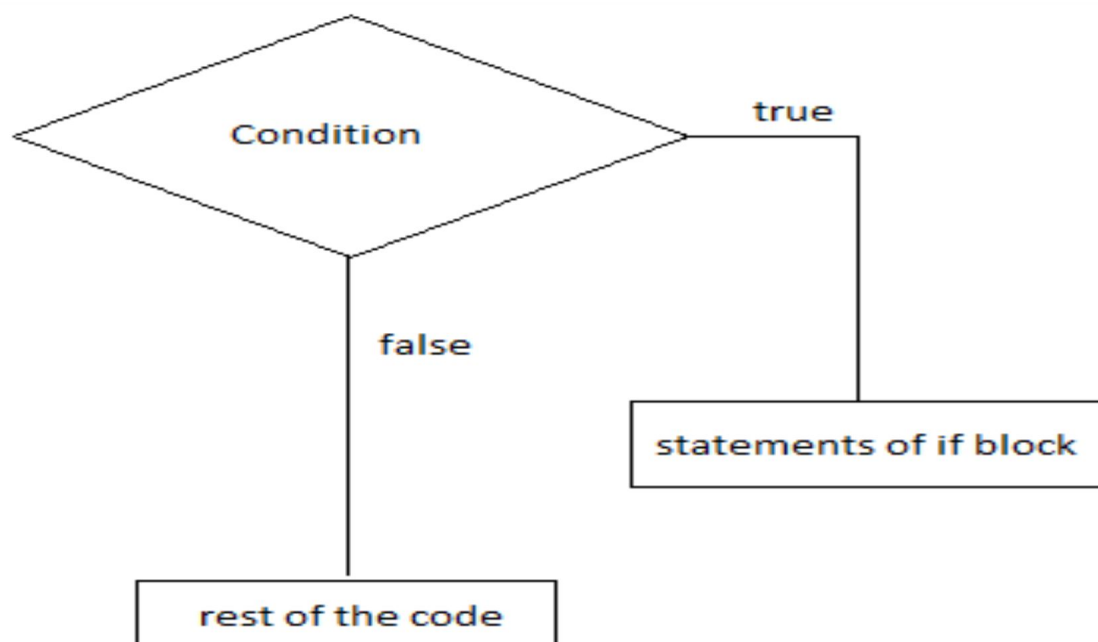
This chapter will take you through Python conditional statements.

The if statement is one of the most popularly used conditional statements in most programming languages. It determines whether a given statement is supposed to be executed or not. When the statement checks for a given condition, and it's found to be true, then the set of code presented inside is executed.

The if condition computes a Boolean expression and executes the block of code only when the Boolean expression is TRUE.

Based on the syntax of the statement, the condition will be implemented to a Boolean expression (true or false). In case the condition is true, then the statement inside the if block is executed, and when the condition is false, program present inside the if block will not be implemented.

Below is the flow chart of the if statement:



If you look at the above flow-chart, the first controller will reach an if condition and determine whether the condition is true, then the statements will be implemented; otherwise the code within the block will be executed.

If-else Statements

The statement itself indicates that if a given a condition is found to be true then execute the statements inside if block and if the condition is false, then execute the else block.

The else block will run only when the condition becomes false; this is the block where you will execute some actions when the condition is not true.

If-else statement computes the Boolean expression and implements the block of code available inside the if block if the condition is True and executes a block of code in the else block when the condition becomes false.

The syntax is:

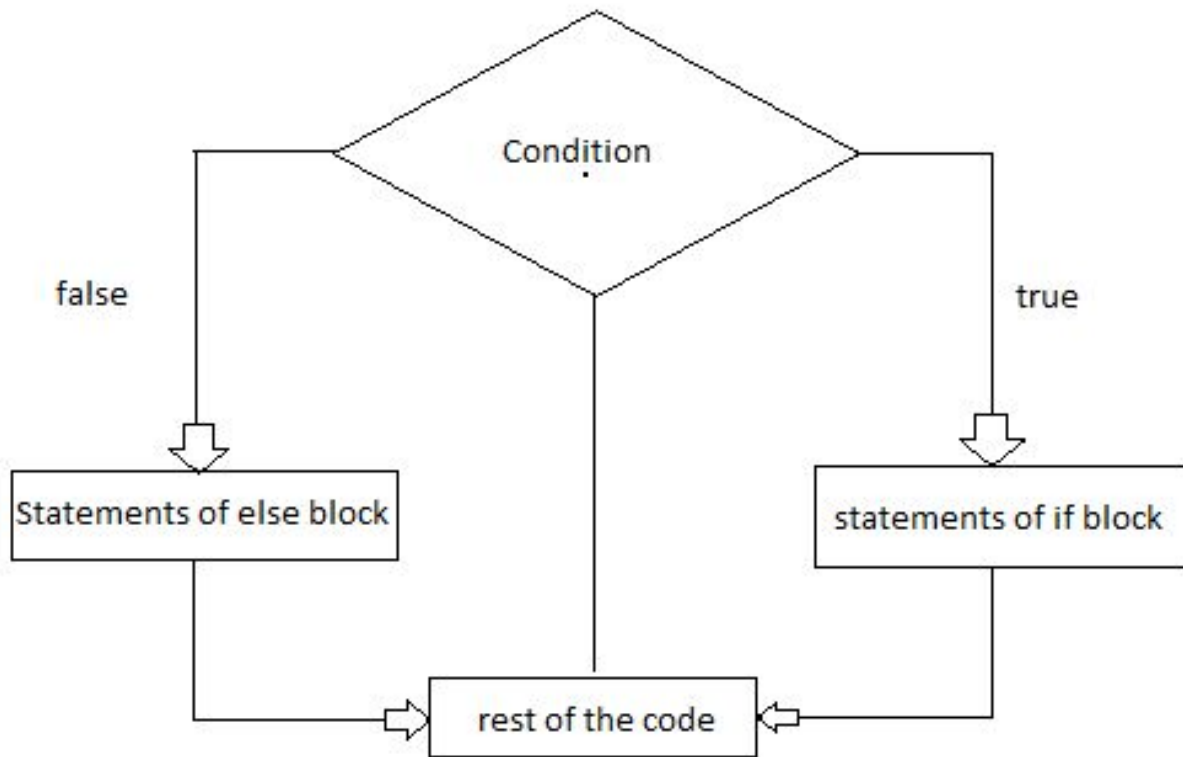
If (Boolean expression):

Block of code

else:

Block of code

In this case, the condition will be executed to a Boolean expression. If the condition is found to be true, then the statements present inside the if block will be implemented and if the condition is false, then the statements inside the program will be implemented.



Let's explore the above flowchart

If you observe keenly, you will notice that the first controller will reach the if condition and check whether the if the condition is true, if found to be true, then the statements of the if block will be implemented otherwise, the block will be executed and later the remaining code outside the if-else block will be implemented.

The Nested if –else Statements

The nested if-else statements refer to an if statement or if-else statement available inside another if or if-else block. Python provides this property as well; this will help us to confirm multiple conditions in a given program.

An if statement available inside another if statement which is present inside another if statements and so forth.

Nested if Syntax

The syntax goes as follows:

If (condition):

#statements to run if the condition is true

If (condition):

#statements to execute if the condition is true

#end of nested if

#end of if

The above syntax clearly shows that the if block will feature another if block in it and so forth. The if block can hold 'n' number of if block within it.

Consider the following program:

Number =10

if (number >0)

 print ("this number is greater than 10")

if (number <10)

 print ("This number is less than 10")

In the above example, a variable number is declared with the value as 10.

First, it will confirm the first statement is true, then the block of code inside the first statement will be executed then it will determine whether the second if statement is true and so on.

Elif ladder

You have learned about the elif statements, but not the elif ladder. As the name implies a program that has a ladder of elif statements structured in the form of a ladder.

if (condition):

 #Statements to execute if condition is true

elif (condition):

#Statements to be executed when if condition is false and elif condition is true
elif (condition):

#Set of statements to be executed when both if and first elif condition is false
and second elif condition is true

elif (condition):

#Set of statements to be executed when if, first elif and second elif conditions
are false and third elif statement is true

else:

#Set of statement to be executed when all if and elif conditions are false
is statement is handy when you want to test multiple expressions.

Example of elif ladder

```
my_marks = 89
```

```
if (my_marks < 35):
```

```
    print ("Sorry!!!, You are failed in the exam")
```

```
elif(my_marks < 60): print("Passed in Second class") elif(my_marks  
> 60 and my_marks < 85):
```

```
    print ("Passed in First class")
```

```
else:
```

```
    print ("Passed in First class with distinction")
```

This program demonstrates the elif ladder. First, the control enters the if statement and determines the condition if it is true, if found to be true, then the set of statements inside the if block will be implemented else it will be skipped and the controller will come to the first elif block and compute the condition.

The same process will proceed for all the remaining elif statements and in case all if and elif conditions are found to be false then the else block will be executed.

The if-else in a Single Line

In Python, still it's possible to write if statements and elif statements in a single line without fearing the indentation.

If Statement in a Single Line

We know we can write if statements as indicated below:

If (condition):

 #set of statements to implement if condition is true

In Python, it is okay to write the above block of code in a single line.

For example:

If (condition): statement 1; statement 2; statement 3; ...statement n

In case the condition is true, then execute statement 1, statement 2, and so forth.

However, if the condition is false, then none of the statements will be executed.

elif statements in a single line

syntax

if(condition):

 #set of statement to implement if it's true

 elif(condition1):

 #set of statements to run if the condition is true

 S else:

 #set of statements to run if condition and condition1 is false

This block can still be written as shown below:

Syntax:

if (condition): #Set of statement to execute if condition is true

elif (condition1): #Set of statement to execute if condition1 is true

else: #Set of statement to execute if condition and condition1 is false

There can be different statements as well, you simply need to differentiate it using a semicolon (;)

Syntax:

if (condition): statement 1; statement 2; statement 3;...;statement n

elif (condition): statement 1; statement 2; statement 3;...;statement n

else: statement 1; statement 2; statement 3;...;statement n

Multiple conditions in if statements

It's not that you can just write one condition within an if statement, it is still possible to evaluate multiple conditions in if statement like below:

```
num_1=10
```

```
num_2 = 20
```

```
num_3 = 30
```

```
if (num_1== 10 and num_2 == 20 and num_3 == 30);
```

```
    print ("All the conditions are true")
```

In this example, the if statement we're evaluating multiple conditions using AND operator, this means if all the conditions are true only when the statements within an if block will be implemented.

It's also possible to specify the OR operators too.

For example:

```
namefruit = "Apple"
```

```
if (namefruit == "Mango" or namefruit == "Apple" or namefruit == "Grapes"):
```

In this example, out of the three conditions, only one condition is true, and that's the rule of OR operator. If any one condition is true, then the condition is true and the statement available inside the if block will be implemented.

Let's explore a real-time scenario to determine the number of days present in a month. You know that in a leap year the number of days will change. You will see this in a programmatic way using if, elif, and else statements.

```
currentYear = int(input("Enter the year: "))
month = int(input("Enter the month: "))
if ((currentYear % 4) == 0 and (currentYear % 100) != 0 or (currentYear % 400) == 0):
    print("Leap Year")
    if (month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10 or month == 12):
        print("There are 31 days in this month")
    elif (month == 4 or month == 6 or month == 9 or month == 11):
        print("There are 30 days in this month")
    elif (month == 2):
        print("There are 29 days in this month")
    else:
        print("Invalid month")
elif ((currentYear % 4) != 0 or (currentYear % 100) != 0 or (currentYear % 400) != 0):
    print("Non Leap Year")
    if (month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10 or month == 12):
        print("There are 31 days in this month")
    elif (month == 4 or month == 6 or month == 9 or month == 11):
        print("There are 30 days in this month")
    elif (month == 2):
        print("There are 28 days in this month")
    else:
        print("Invalid month")
else:
    print("Invalid Year")
```

Enter the year: 2020
Enter the month: 4
There are 30 days in this month

Output: 2

Enter the year: 2020
Enter the month: 1
There are 31 days in this month

Output: 3

Enter the year: 2019
Enter the month: 2
There are 28 days in this month

Output: 4

Enter the year: 2020
Enter the month: 2
There are 29 days in this month Output:

Boolean Expressions

A Boolean expression may comprise one of two possible values: true or false.

The simplest Boolean expressions comprise of False and True. An expression that compares numeric expressions for equality is called Boolean expression. The simplest form of Boolean expressions contains relational operators to create a comparison of two expressions.

An expression such as $5 < 10$ is legal but of little significance, because it's always true. The expression is less likely to result in confusion among readers. However,

because variables are dynamic, the value they hold during a program's implementation is bound to change.

Iteration

When it comes to programming, iteration means the repetition of lines of code. It's an essential property in computer programming that helps find solutions to problems. Iteration and conditional execution are the main stems of algorithm development.

Let us start with the:

While Statement

Say you want to write a program that can count to 10,000? How will you approach this problem? Will you sit down and write 10,000 printing statements? Although you can, that is going to consume a lot of your time. However, counting is frequent in computers, in fact, computers can count extraordinary values. So, there must be a way out. What you need to do is to print the value of a variable, and repeat the process until you reach 10,000. The method of executing the same code repeatedly is known as looping. Python language has two special statements, *while* and *for*, that handle iteration.

Here is an example of a program that uses a while statement.

```
num = 2
while num <= 5:
    print (num)
    num += 1
```

The while statement in the above program will regularly display the variable num. The program then executes this code snippet 4 times.

```
print (num)
count += 1
```

After each output, the program increases the num variable by one. After executing the code four times, the condition will become false, and hence the block of code is

not implemented anymore.

The line `while num <= 5:` opens the while statement. The next expression after the while keyword is the condition that evaluates whether the block should be executed. As long as the outcome of the condition is true, the program will continue to run the block of code. But, once the condition evaluates to false, the loop terminates.

Additionally, when the condition is found to be false at the start, the program exits.

The standard syntax of the while statement comprises of:

while condition:

block

The term while refers to a Python reserved word that opens the while statement.

The condition indicates whether the body will be implemented or not. A colon (:) to appear after the condition.

The block consists of one or more statements that need to be executed in case the condition is true. All statements that complete the block must be indented one level deeper than the first line of the while statement. In general, the block is a member of the while statement.

The while statement looks similar to the if statements, and hence it's easy for new beginners to confuse the two. Sometimes, they can use if when they wanted to use while statement. As such, new programmers should be keen on the application of a while and if statement.

For the while statement, the running program implements the condition before it executes the while block, and then verifies the condition after the while block. Well, if the condition remains true, the program will proceed to execute the code inside the while block until it reaches a point where the condition is false. At this point, the loop terminates from execution.

Definite and indefinite loops

Let's explore the following code snippet:

```
n = 1
```

```
while n <= 10:
```



```
print(n)
```

```
n += 1
```

Let's find the number of iterations in the above program segment. If you carefully evaluate this code, you'll learn that there are 10 iterations. Hence, this loop is said to be definite because you can accurately determine the number of times the loop will iterate.

Now, consider the following code:

```
c = 1
```

```
terminate = int (input ()))
```

```
print (n)
```

```
n += 1
```

For this code, it is difficult to tell the number of iterations. That number depends on the input by the user. However, you can count the number of repetitions the while loop will when the user enters input before the next execution begins...

In general, the while statement is best for indefinite loops.

The for Statement

The while loop is suitable for indefinite loops. Initially, a while loop was used to execute a definite loop like:

```
n = 1
```

```
while n <10;
```

```
print(n)
```

```
n += 1
```

In the above code snippet, the print statement will only execute 10 times. This code requires three vital sections to control the loop. That is the initialization, check and update.

Python has a handy method to display a definite loop. The for statement repeats over a series of values. One way to illustrate a series to apply a tuple. For example:

```
for n in 1, 2, 3, 4, 5, 6, 7:
```

```
    print(n)
```

The above code will run exactly as the while loop in the previous section. In this example, the print statement executes 7 times. The code will print first 1, then 2, and so forth. The last value printed is 7.

The for statement differs from the while statement by expecting a set of features that it will repeat over. For example, you may need a loop that runs over different numbers such as the integer values from 1 to 10. It is possible to define a counter inside a while loop to ensure this happens, but the statement is designed to handle this kind of circumstance easily and efficiently. Take for instance, a hello, world program. This program will print the string five times using a for a loop. To achieve that, you'll learn about the range function.

```
for x in range(5):
```

```
    print("Hello, World!")
```

The range provides a means to access different numbers efficiently. When it's combined with a for a statement, it computes a loop that executes a fixed number of times. The program presented above will print the string "Hello, World!" five times. It achieves this by allocating the new variable x a certain value in the requested range for every iteration of the loop. If you can remember how the following indices are computed using string character positions, you'll remember that these indices begin at zero. To illustrate this, review the following code:

```
X for x in range (5)
```

```
    print ("The value of x is {0}.".format(x))
```

When you combine the for function plus the range built-in function, each loop iteration sets x to the next value in the range requested. It's like there was a counter with a while loop that you're manually requesting. The for loop takes the next value automatically, the variable x acquires a new value, and implements the inner code block.

Python Continue, Break and Pass

The control statements in Python language are used to regulate the sequence of execution of the program depending on the logic and values.

Python has 3 types of control statements

-Continue

-Break

-Pass

In summary, this chapter has looked at a conditional statement in Python as well as iteration statements. These are statements that affect the flow of execution of a program.

There are different types of conditional statements such as if, if-else, elif, nested if and many more which alter the execution of a program.

If the statement computes a Boolean expression to true or false, if the condition is true then the statement within the if block will be implemented. But if the condition is false then the statement present within the else block will be executed only if you've written the else block.

We have an additional statement known as the elif statement where the else statement is combined with an if statement, which runs depending on the previous if or elif statements.

Notes:

- Python offers conditional statements that are useful for verification and validation reasons.
- Python has two types of looping statements that help implement a specific block of code or statements repeatedly.
- You use the "while loop" when you don't know the number of times that you need to iterate and if you see the number of times you need to iterate, then 'for loop' is perfect.
- Python also has 3 control statements that allow you to control the flow of execution of a program.



Chapter 5: Python Functions

This chapter will take you through functions in Python. At the end of this chapter, you will be able to write functions in Python, call Python functions and more!

First, functions are a critical part of the Python language. Already you have come across several Python functions, including built-in features, or tasks that come with its own libraries. But, as a Data Scientist, you'll regularly need to write your own functions to solve issues that you encounter working with data.

That is why this chapter will walk you through functions in Python.

Let's dive in:

Functions in Python

You require functions in Python to help control inputs and outputs in programs. All languages of programming are built to execute data. Therefore, functions are perfect to handle and change this kind of data.

The changes typically happen to trigger results such as finding results. And, the set of instructions required to accomplish this originates from logically functional blocks of code that can be reused again.

The main code is a function. It's an essential function because every other function is connected to it, and runs from your main code. However, if the function has not yet been defined, you'll need first to define it before you proceed to use it. The definition of a function outlines the steps of its operation.

Consider this: which is better, write a code snippet 10 times, or only once and execute it 10 times.

Therefore, functions are simply tasks that a user wants to implement. However, by defining a function once using a name will allow you to reuse that functionality without making your main programs appear scary. This generally cuts down on the lines of code, and even simplifies debugging.

We'll jump into that shortly, but for now, you need to understand the reason why you need to use function is because of the property of reusability. The fact that advanced

operations can be compiled into single tasks that would perform with a single call by its name is what has allowed computer codes to become more evident.

Every computer programming language nowadays allows you to create and make use of these functions to complete different tasks with just a single call. And, you can even call it any number of times without worrying about logically structuring its code into your major code every time.

Now, let's try to understand their significance to us by using a simple example:

Assume, you've got a television that can store a lot of channels, receive digital radio broadcasts, changes them into what we watch, while at the same time providing us additional options for various features.

But that doesn't imply there's an individual logically scripting the lines of codes for what you watch every time you switch on your TV. Instead, functions for every task in its working have been reasonably defined once and are often reused again and again depending on the features you attempt to use.

This happens by calling different functions as many times as possible from the main feature that is running. Therefore, even if you're increasing or decreasing the volume, its defined function is called repeatedly.

And, when you've got a system running the main code to continue calling these functions when needed has also simplified designing and innovation.

The critical aspect to remember is that whenever this function is called, it implements its tasks based on the instructions outlined.

That's how machines acquire different functions. A calculator is perhaps the most common example. It has the property of addition, multiplication, division, and other features. All these functions have been predefined into it, but it only completes those that you decide to call by pressing its respective button.

Programmers limit the time they spent to code and debug using functions, hence reducing the total development time by applying functions.

Now, let's examine what Python functions exactly are.

So, what are Python functions?

If someone was to ask you this question, what would be your response?

The functions in Python are a great example of the property of reusability. So, to serve a wide range of applications from GUI and mathematical computing to web development and testing, the Python interpreter has built-in functions that are available for use. And, you may also add other libraries or modules to your program that feature pre-defined functions available for use.

All you'll need to do is to download the necessary packages based on their documentation and will freely avail of all its important functionalities by importing them over to your code.

Therefore, once it's defined, a function can be used at different times at any place in any of your codes. Now, this because Python is in line with the DRY principle of software patterns or codes featuring abstractions to avoid redundancy and make sure they can be used freely without exposing any inner parts of their implementations.

The DRY in full means Don't Repeat Yourself, and this tendency of having reusable blocks of code is significant for realizing abstraction in Python. Therefore, to use a function all that you'll need is the name, arguments, purpose and the type of results if it returns any.

It's almost like using a telephone, where you don't really need to master the working of its components to use them. Instead, they've been built to serve standard functions that you can apply directly to accomplish your goals and commit your time to implement all the innovative features of your application program. And, no one wants to understand how a function in your program works on the inside, as long as it does the work.

So, with Python, you should not be concerned about what happens inside the function unless you want to write or modify an existing function.

A function is a small program that executes data to generate output.

To use a function, first, you need to define it. Functions in Python language are defined using the `def` keyword before the name of the function, and include parentheses to its end, followed by a colon.

Types of Functions in Python

Python has three different categories of functions.

- Built-in functions, such as `min ()` to determine the minimum value, `print ()` to display an object to the Python terminal.
- User-defined functions, refer to functions that users can define to use them.
- Anonymous functions. These functions are also known as lambda functions because they're not declared using the standard `def` keyword.

Functions vs. Methods

A method is an example of a function that belongs to a class. You can access a method using an instance of a class. However, functions lack this limitation. It only points to a one-stop function. This means all methods are functions, but not all functions can be methods.

Take the following example. The function `plus ()` is defined, and a class of `sum ()` is also included.

```
# Define a function `plus()`
```

```
def plus(a,b):
```

```
    return a + b
```

```
# Create a `Summation` class
```

```
class Summation(object):
```

```
    def sum(self, a, b):
```

```
        self.contents = a + b
```

```
        return self.contents
```

So, if you want to run the method `sum ()`, first you will have to define an instance of the object class. So, let's proceed to define this object:

```
# Instantiate `Summation` class to call `sum()`
```

```
sumInstance = Summation()
```

```
sumInstance.sum(1,2)
```


Keep in mind that the above instantiation is not important when you need to call the plus () function. Still, you can execute the plus (1,2) without experiencing any problems.

Arguments vs parameters

Parameters refer to names used at the time of function definition, and into which arguments will be enclosed. This means arguments are elements included within the function.

In the preceding example, the sum () function had two arguments although the class featured three parameters, self, a, and b.

The first argument that appears in every class method points to the current class instance. In the above example is Summation.

Therefore, it's impossible to reference self because self is the parameter name for an implicitly passed argument that references to the instance method being invoked.

Defining User Functions

The four steps to follow when you want to define Python functions comprise:

1. First, write the def keyword and then write the name of the function.
2. Supply parameters to the function. The parameters need to be within the parenthesis of the function. End the line with a colon.
3. Include statements that the functions should implement.
4. Add a return statement

Example:

```
def hello():  
    print("Hello World")  
    return
```

Don't forget that you can define one or more function parameters for your user defined functions.

Return Statement

```
def hello():  
    print("Hello World")  
    return("hello")  
  
def hello_noreturn():  
    print("Hello World")  
  
# Multiply the output of `hello()` with 2  
hello() * 2  
  
# (Try to) multiply the output of `hello_noreturn()` with 2  
hello_noreturn() * 2
```

The second function returns an error because it's impossible to work on a None value. Hence, if you run this code, you will get a TypeError that will notify you that you cannot complete the multiplication operation using NoneType.

Functions immediately terminate once they come across a *return* statement.

```
def run():  
    for x in range(10):  
        if x == 2:  
            return  
  
print("Run!")  
  
run()
```

A return statement can display more than one value. Tuples allow you to return more than one statement.

Take a look at the following example to learn how you can return more than one value.

```
# Define `plus()`  
  
def plus(a,b):  
    sum = a + b  
    return (sum, a)  
  
# Call `plus()` and unpack variables  
  
sum, a = plus(3,4)  
  
# Print `sum()`  
  
print(sum)
```

Calling a Function

In the previous examples, you learned how to call a function. Calling function as the name suggests means that you execute the function either directly from the Python prompt or from a different function.

You call your hello () function by running hello ().

Adding Docstring to a Python Function

The most vital property of defining Python functions is the docstring. The docstring

Another essential feature of creating Python functions is the docstrings. Docstrings describe the tasks that your functions do such as its return values.

The descriptions work as documentation for your function so that anyone can understand the function's docstring, without the need to monitor all the code within the function definition.

The function docstrings appear instantly after the function header and are enclosed within triple quotation marks. The right Docstring for the hello () function is prints "Hello World".

```
def hello():  
    """Prints "Hello World".  
  
    Returns:  
        None  
    """  
    print("Hello World")  
    return
```

Remember that docstring can be more than the one presented here. If you would like to study docstring further, then you should check out Github repositories of Python libraries where you'll get a lot of examples.

Python Function Arguments

At the start, you read the difference between arguments and parameters. Briefly, arguments refer to things given to any function or method call, while the function or method code refers to the arguments by their parameter names. There are four categories of arguments that Python user-defined function can assume:

- Keyword arguments
- Default arguments
- Required arguments
- Variable number of arguments

Keyword Arguments

If you want to call the parameters in the correct order, you can include the keyword arguments # in your function call. Example:

```
#Define `plus()` function
```

```
def plus(a,b):
```

```
    return a + b
```

```
# Call `plus()` function with parameters
```

```
plus(2,3)
```

```
# Call `plus()` function with keyword arguments
```

```
plus(a=1, b=2)
```

By using the keyword arguments, you can also alternate around the order of the parameters and still attain the same result when you run your function.

```
# Define `plus()` function
```

```
def plus(a,b):
```

```
    return a + b
```

```
# Call `plus()` function with keyword arguments
```

```
plus(b=2, a=1)
```

Required Arguments

As the name suggests, these are arguments that have to be present. These arguments require to be passed during the function call and in exactly the correct order like in the example below:

```
# Define `plus ()` with required arguments
```

```
def plus(a,b):
```

```
    return a + b
```

Global and Local Variables

Variables defined within the body of a function have a global scope. In other words, local variables are defined inside a function block and can only be accessed within that function, while global variables can be acquired by all functions that could be in your script.

```
# Global variable `init`

init = 1

# Define `plus()` function to accept a variable number of arguments
def plus(*args):

    # Local variable `sum()`

    total = 0

    for i in args:

        total += i

    return total

# Access the global variable
print("this is the initialized value " + str(init))

# (Try to) access the local variable
print("this is the sum " + str(total))
```

You will notice that you'll get a `NameError` that shows the name 'total' isn't defined when you attempt to display the local variable `total` that was defined within the function body. On the flipside, the `init` variable can be displayed without any issues.

Anonymous Functions within Python

Anonymous functions also known as `lambda` because they use `lambda`. Example:

```
double = lambda x: x*2

double(5)
```

The special thing about this function is that it doesn't have a name like the previous examples you've come across in the first part of this chapter.

You use lambda functions when you want a nameless function for short term purposes. Specific contexts where this can be relevant is when you're dealing with `filter()`, `map()`, and `reduce()`:

```
from functools import reduce

my_list = [1,2,3,4,5,6,7,8,9,10]

# Use lambda function with `filter()`
filtered_list = list(filter(lambda x: (x*2 > 10), my_list))

# Use lambda function with `map()`
mapped_list = list(map(lambda x: x*2, my_list))

# Use lambda function with `reduce()`
reduced_list = reduce(lambda x, y: x+y, my_list)

print(filtered_list)

print(mapped_list)

print(reduced_list)
```

The Main () Function

If you've programmed with another language such as Java, you'll agree that the main function is needed to run functions. As you've seen in the previous example, this function is not a must for Python. However, if you use a `main()` function within your Python program, it can be handy to organize your code logically.

It's easy to define a `main()` function and call it the same way you've done with other functions.

```
# Define `main()` function

def main():
```

```
hello()
```

```
print("This is a main function")
```

```
main()
```

Currently, the code of your main () function will be called once you import it as a module. To ensure that this doesn't happen, you call the main () function when `__name__ = '__main__'`

Apart from the `__main__` function, there is also the `__init__` function that will initialize an instance of a class or an object. This acts as a constructor and is automatically called once you create a new instance of a class. With that function, the newly defined object is allocated to the parameter self.



Chapter 6: Python Operators

Python operators implement operations on values and variables. Operators can modify individual items and output results. The data items are known as operands. Operators are either represented using keywords or unique characters.

Arithmetic operators

These implement different arithmetic computations such as subtraction, division, addition, exponent, etc. There is a different method for arithmetic computation in Python like you can apply the declare variable, or call functions.

For the arithmetic operators, consider a simple example of addition where you add two-digit $2+3=5$

```
b= 4
```

```
c= 5
```

```
print (b + c)
```

Still, you can use arithmetic operators such as division, subtraction, and multiplication.

Python logical operators

There are 3 logical operators in Python.

or, and, not

Example:

```
a or b
```

```
a and b
```

```
not a
```

not b

Operator precedence

Python operators are computed based on a set of priority. Below is a table that shows the operator precedence in Python.

	Description	Operators
1	Exponentiation	**
2	Complement, unary plus, and minus	~, +, -
3	Multiplication, division, modulo, and floor division	*, /, %, //
4	addition and subtraction	+ -
5	Right and left bitwise shift	>>, <<
6	Bitwise 'AND'	&
7	Regular 'OR' and Bitwise exclusive 'OR'	~, ^
8	Comparison operators	<= < > >=
9	Equality operators	== !=
10	Assignment operators	=, +=, -=, *=, /=, %= //= **=
11	Identity operators	is, is not
12	Membership operators	in, not in
13	Logical operators	or, and, not



Chapter 7: File Handling

Python provides a critical feature for reading data from the file and writing data into a file.

In most programming languages, all the values or data are kept in some volatile variables.

Since data will be stored in those variables during run-time only and will disappear once the program execution ends, therefore, it's better to save these data permanently using files.

Once you store data on a file, the next important thing is its retrieval process because it's stored as bits of 1s and 0s, and in case the retrieval does not occur well then it becomes completely useless and that data is said to be corrupted.

How Python Handles Files?

If you're working in an extensive software application where they execute a massive amount of data, then we can't expect those data to be kept in a variable because variables are volatile.

Therefore, when you want to deal with these situations, the role of files will come into the picture.

Since files are non-volatile in nature, the data will remain permanently in a secondary device such as Hard Disk and using Python to deal with these files in your applications.

Do You Consider How Python Will Handle These Files?

Let's assume how normal people will deal with these files. If you want to read the data from a file or write the data into a file, then you need to open the file or create a new file if the file doesn't exist and then conduct the normal read/write operations, save the file and close it.

Similarly, the same operations are accomplished in Python with the help of in-built applications.

Types of Files in Python

There are two kinds of files:

1. Text files
2. Binary files

A file whose contents can be examined using a text editor is known as a text file. A text file refers to a sequence of ASCII characters. Python programs are examples of text files.

A binary file stores the data in the same manner as stored in the memory. The mp3 files, word documents are some of the examples of binary files. You cannot read a binary file using a text editor.

In Python language, file processing takes the following steps.

- Open a file that returns a filehandle.
- Use the handle to read or write action.
- Close the filehandle.

Before you perform a read or write operation to a file in Python, you must open it first. And as the read/write transaction finishes, you should close it to free the resources connected with the file.

Let's now look at each step in detail.

Access_mode: This is represented with an integer e.g read, write, and append. The default setting is the read-only <r>.

Buffering: The default value for buffering is 0. A zero value shows that buffering will not happen. If the value is 1, then the line buffering will happen while accessing the file. If it's more than 1, then the buffering action will proceed based on the size.

File_name: This is a string that represents the name of the file you want to access.

File open modes in Python language

<r>

<rb+>

<rb>

<w+>

<wb+>

<r+>

<w>

<wb>

Python File Object Properties

Once you call the Python `open ()` function, it returns an object, which is the filehandle. Additionally, you need to understand that Python files have different features. And you can take advantage of the filehandle to list the features of a file it belongs.

Close a File in Python

It is good always to close a file when you finish your work. However, Python has a garbage collector to clean up the unused objects. However, you need to do it on your own instead of leaving it for the GC.

The Close Method

Python offers the `<close ()>` method to close a file.

When you close a file, the system creates resources allocated to it. And it's easy to accomplish.

Closing a file releases essential system resources. If you forgot to close the file, Python will do it automatically when the program ends or the file object is no longer referenced inside the program. However, in case your program is large and you're reading or writing multiple files that can consume a massive amount of resources on the system. If you continue opening new files carelessly, you might run out of resources.



Chapter 8: Dictionaries

In this chapter, you will be learning about a new data structure—the dictionary. The dictionary (also known as a hashtable or hashmap in other languages) is one of the most powerful data structures Python has available to use. Luckily, since it's built in to the Python language, you don't have to implement it yourself. You will also be learning about tuples. What you will learn:

- How to use a Dictionary Data Structure
- How to use Tuples

Mastering the Python dictionary will help you to model different real-world objects more accurately. You will create a dictionary representing a person and store unlimited information as you would like about the individual. You can store their location, name details, profession, and other features you can describe. You'll be able to store any two kinds of information that can be compared, a list of words, and their meaning.

A dictionary in Python has key-value pairs. Each key is related to a value, and you can use a key to access the value linked to that key. The value of a key can be a string, number, or even a different dictionary. In fact, you can include any object that you can build in Python as a value in a dictionary.

Python dictionaries are surrounded using braces, plus a series of key-value pairs within the braces.

A key-value pair is a collection of values bounded to each other. When you present a key, Python will return the value linked with that key.

The simplest dictionary example has exactly a single key-value pair.

Example:

```
alien = {'color': 'green'}
```

This dictionary holds a single piece of information about alien, and that is the color. The string 'color' is a key inside this dictionary, and it's connected to the value 'green'

Getting Setup

In this section, you are going to be working in the Python Idle editor as well as the terminal. You will learn about the dictionary in the Idle editor, and then you are going to graduate to creating your own files and working with data.

How to Access the Values Inside a Dictionary?

To acquire the value linked with a key, provide the name of the dictionary and place the key within a set of square brackets as shown below:

```
print(alien['color'])
```

This returns the value related to the key 'color' from the dictionary.

It is possible to have an unlimited number of key-value pairs inside a dictionary.

Adding New Key-value Pairs

Dictionaries change every time, and you can include new pairs of key-values to a dictionary at any given time. For instance, to include a new key-value pair, you would provide the name of the dictionary followed by the new key within square brackets.

```
alien = {'color': 'green', 'points': 5}
print(alien)
u alien['x_position'] = 0
v alien['y_position'] = 25
print(alien)
```

Now, we want to add two kinds of information to the alien dictionary: That is the alien's x and y coordinates. These coordinates will allow you to show the alien in a specific position on the screen. Let's position the alien on the left edge of the screen. This will be 25 pixels from the top going downwards. Since the coordinates of a screen begin at the upper-left corner of the screen, you will position the alien on the left edge of the screen using the x-coordinate to 0 and 25 pixels from the top by adjusting its y-coordinate to positive 25, as shown below:

```
{'color': 'green', 'points': 5}  
{'color': 'green', 'points': 5, 'y_position': 25, 'x_position': 0}
```

We

begin by first defining the same dictionary that we have been using. Next, print this dictionary, and display a snapshot of its details. In the second line, you include new key-value pair to the dictionary:

The final dictionary version has a total of 4 key-pair values. Remember that Python doesn't consider the order of storage, it only handles the connection between every key and its value.

The Second Major Data Structure – The Dictionary

The Dictionary is a powerful data structure that has a 'key' and a 'value'. Each key is unique in the dictionary, and it has an associated value. The associated value however, does not need to be unique.

Examples of dictionaries in real life include:

- o A phone book:
 - o Key – The phone number
 - o Value – The person's name

- o A physical dictionary (hence where the name of this data structure comes from)
 - o Key – The word
 - o Value – The description of the word (i.e. the definition).

- o A Student identification number at a university
 - o Key – the number
 - o Value – The person's name

- o Your Subway Loyalty Card:
- o Key – Your card number
- o Value – Your points you’ve amassed!

The key in the dictionary can be a string or a number. In fact, it can be any data type! The takeaway though is that the key must be unique!

Here’s a look at some sample tables of the above examples to again illustrate keys and values.

Key(Phone number)	Value(Persons name)
7323245	‘Joe’
9822912	‘Sue’
6323421	‘Moe’

Note that the phone number is represented as an integer and each of the values as a string.

Key(Word)	Value(Definition)
‘Cat’	‘A small domesticated carnivore’
‘Python’	Any of several boa constrictors in the subfamily Pythoninae’
‘Byte’	Adjacent bits, usually eight, processed by a computer as a unit’

Note that the key is a string and the value also a string in this example.

Key (Student ID Number).	Value (Persons Name)
127323	‘Mike’
187428	‘Tomoki’
493209	‘Raoul’

Key (Reward Card Number).	Value (Points)
1209482104812	47
2098520935820	434
3248098324093	434

Note that in this example, the points might change, and dictionaries allow us to modify values. Values also can be duplicated, but remember the keys cannot (the older key will be overridden if there is a duplicate!).

Note: On Learning to Program: A dictionary is one of many data structures available to us that is built into Python. In fact, there are numerous other data structures and algorithms available to us that we always have at our disposal. It may even become intimidating or overwhelming! However, what I urge you to do is to work through this exercise (and future exercises) at least once through even if you don't understand all of the details. Then revisit the details. The best way to learn is often to complete a project, and then when you revisit it you will have a better idea of what problem you are trying to solve, and what is important to learn and understand in intimate detail.

Dictionary Examples

Let's first create a dictionary to model a phonebook. The first thing we need to do is decide whether our key will be a name or a value. As you saw in the previous example, you can use a phone number. Phone numbers themselves are unique, so they are a good candidate for a key.

However, you can also use a name as a key, because it is typically easier for us to remember a person's name.

```
phoneBook = {'Mike': 55555555} # 'Mike' is the key
```

In this example, you can only have one friend named Mike. If you want more Mike's, you'd have to store them as Mike01, Mike02, Mike03, etc.

You can print out a specific entry from our phonebook using the following:

```
print phoneBook['Mike']
```


If you want to output the entire contents of the phonebook, you can simply use the print

command.

```
# Print phoneBook
```

```
print phonebook
```

Over time, you will want to grow and modify your dictionary, so you can add entries like the following.

```
# Add a new item
```

```
phoneBook['Michelle'] = 43255322
```

```
# Lets confirm our entry went in.
```

```
print phoneBook
```

```
# Delete Mike from our phonebook, we'll never need to call him!
```

```
del phoneBook['Mike']
```

```
# If Michelle changes her number, we can update it by accessing her  
entry with her key ('Michelle') and then simply re-assigning a new  
value.
```

```
phoneBook['Michelle'] = 3252352
```

```
# Confirm our changes have been made.  
print phoneBook
```

```
# Sometimes we are not sure who is in our phonebook, so we have  
to  
# iterate over all of # the keys. When we know what keys are  
available,  
# we can then use those keys to quickly index into our phone book  
and  
# retrieve the value (a phone number in this case).  
# print keys  
for x in phoneBook:  
    print x  
# Alternatively, if we just need the numbers, we can print out all of  
the values.  
# This might be a nice thing to do if you want to call everyone and  
wish them a happy  
# new year.
```

```
# This might be an evil thing to do if you want to call everyone  
and  
try to scam them!  
# Look out!  
# print values
```

```
for x in phoneBook:  
    print phoneBook[x]
```

ANOTHER LOOK AT THE DICTIONARY

The dictionary is what is known as an 'associative data structure'. This means that a value is associated with a key. Great, that makes perfect sense! This should be intuitive, because this is often how our brain works. We generally do not think of a number and say, oh that is Mike's number. We generally think of a name (e.g. 'Mike'), and then recall what his number is. Our brains work very well by associating one thing with the next, so it should not be a surprise we can do something similar with computers.

THE TUPLE

The Tuple is a way in Python to group information together. It is like a list, except that we cannot modify it once we have created a tuple.

Let's go ahead and create a tuple that groups together information about an individual.

```
Person = ('Mike', 100)  
>>> Person  
(('Mike', 100))
```

Let's create a Person Tuple that will take a str, and an int as the two types of data we want to store. The first value is a string storing a name, and the second is an integer value for how many miles they ran this month.

```
>>> Mike = ('Mike', 1)  
>>> Willie = ('Willie', 2)  
>>> Tomoki = ('Tomoki', 17)  
>>> Raoul = ('Raoul', 14)  
>>> BestFriends = [Willie, Mike, Tomoki, Raoul]  
>>> BestFriends  
[('Willie', 2), ('Mike', 1), ('Tomoki', 17), ('Raoul', 14)]
```

So, the tuple itself is just a collection of values held together. We can actually store tuples in

a list if we want. Let's create some more tuples with names and each person's favorite number as a value in a second field.

```
>>> BestFriends[0]
('Willie', 2)
>>> type(BestFriends[0])
<type 'tuple'>
```

Now when the BestFriends list is output, we see that each entry (separated by a comma) is a tuple that we created. We can then access elements in our list by using their position in the list as we have previously done. Our list will now return a tuple. And if we check the type of the entry, we see that it is indeed a tuple.

This might take some getting used to, as we're using two levels of indirection to access the

individual element we would like. With some practice however, you will be able to do this with no problem. Let's now combine what we have learned with dictionaries and tuples. Let's make the....NBA

Superstar Basketball Dictionary!

CHALLENGE PROBLEM

Take in the below NBA player data, and create tuples for them. Here is the format:

The first value in the tuple is the number of championships won, and the second is the number of season the player has played.

Key	Value
Bill Russel	(11,13)
Sam Jones	(10,12)
Robert Horry	(7,16)
Michael Jordan	(6,15)
Shaquille O'Neil	(4,19)
Mike Shah	(0,0)

Example:

Creating a tuple.

```
samJones = (10,12)
```

This makes the variable 'samJones' hold the tuple (10,12).

We then want to put all of these players (or rather, their tuples) into a dictionary, and then output all of the NBA players who have won at least 1 championship.

Here is an example of creating an empty dictionary, and adding our player to it.

```
# Now let's create an empty dictionary
nbaDictionary = {}

# Add a key of 'Sam Jones' and a
# value of samJones (which is a tuple)
nbaDictionary['Sam Jones'] = samJones
```

Goals we want to achieve:

- Output all NBA players in our dictionary who have won at least one championship
- Output all of the NBA players in our dictionary who have won a championship in greater in 50% of the seasons they have played.

Hints and gotchas:

- print out your dictionary after you input your entries to visually see the data.
- Can we divide by zero in mathematics?
- Use an 'and' statement to check two conditions.

o e.g. if value > 5 and value != 0:

- When dividing two integers, we get an integer back. If we want a float (i.e. decimal number) as a result, we have to divide to floats.

```
>>> 7/5
```

```
1
```

```
>>> float(7)/float(5)
```

```
1.4
```

o Example:

```
NAVIGATION
<li><a href="index.html">Home</a></li>
<li><a href="home-events.html">Home Events</a></li>
<li><a href="multi-col-menu.html">Multiple Column Menu on Larger Viewports</a></li>
<li class="has-children"><a href="#" class="current">Header Options</a>
<ul>
<li><a href="tall-button-header.html">Tall Button Header</a></li>
<li><a href="image-logo.html">Image Logo</a></li>
<li class="active"><a href="tall-logo.html">Tall Logo Image</a></li>
</ul>
</li>
<li class="has-children"><a href="#">Carousels</a>
<ul>
<li><a href="variable-width-slider.html">Variable Image Width Slider</a></li>
<li><a href="testimonial-slider.html">Testimonial Slider</a></li>
<li><a href="featured-work-slider.html">Featured Work Slider</a></li>
<li><a href="equal-column-slider.html">Equal Column Slider</a></li>
<li><a href="video-slider.html">Video Slider</a></li>
<li><a href="mini-bootstrap-carousel.html">Mini Slider</a></li>
</ul>
</li>
```

Chapter 9: Object-Oriented Programming

Object-Oriented programming is an extensive concept used to create powerful applications. Data scientists are required to build applications to work on data, among other things. This chapter will explore the basics of object-oriented programming in Python.

Object-Oriented Programming abbreviated as OOP has several advantages over other design patterns. The development process is faster and cheaper, with great software maintainability. This, in turn, results in better software, which is also filled with new attributes and methods. The learning curve, is, however, complex. The idea might be complicated for newbies. In terms of computation, OOP is slower and consumes a lot of memory because more lines of code have been written.

Object-oriented programming relies on the important programming concept, which makes use of statements to change a program's state. It concentrates on illustrating how a program should operate. Examples of imperative programming languages are Java, C++, C, Ruby, and Python. This is different from declarative programming which deals with the type of computer program should achieve, without detailing how. Examples consist of database query languages such as XQuery and SQL.

OOP relies on the property of classes and objects. A class can be considered as a 'blueprint' for objects. These can feature their own characteristics and methods they execute.

Example of OOP

Take an example of a class Dog. Don't consider it as a specific dog, or your own dog. We're describing what a dog is and what it can do in general. Dogs have an age and a name. These are instance properties. Dogs can also bark; this is a method.

When you discuss a certain dog, you would have an object in programming: an object is a class instance. This is the basic state on which object-oriented programming depends.

Now let's look at OOP in Python language.

Python is a powerful programming language that allows OOP. You will use Python language to define a class with properties and methods, which you will later call. Python has extra benefits than other languages. First, the language is dynamic, and a high-level data type. This implies that development takes place faster than Java. It doesn't need the programmer to declare variable types and arguments. This makes Python easy to learn for beginners. Its code is more intuitive and readable.

It is important to remember that a class basically provides the structure. This is a blueprint that outlines how something needs to be defined. However, it doesn't offer any real content. For example, shape () class may specify the size and name of shapes, but it will not indicate the exact name of a shape.

You can view a class as a concept of how something should be executed.

Python Objects

Although the class is the blueprint, objects or instances are members of a given class. It's not a concept anymore. It's an actual shape, like a triangle with three sides.

Put differently, a class is like a questionnaire. It will define the required information. Once you complete the form, your actual copy is an instance of the class. It has original information relevant to you.

You can complete different copies to have multiple instances, but without the form, you'll be lost, not knowing the kind of information required. Therefore, before you can create individual objects, you need to define what is required by the class.

Defining a Class in Python

Below is a simple class definition in Python:

```
Class Dog (object)
```

```
    Pass
```

When defining a class in Python, you begin with the class keyword to show that you're writing a class, then you follow it with the name of the class. In the above example, Dog is the name of the class.

The above class definition has the Python keyword pass; this is normally used as a placeholder where code will finally go. Why this keyword has been used is to avoid the code from throwing an error.

The object section enclosed in parentheses demonstrates the parent class that you're inheriting from. But this is no longer required in Python 3

because it's the implicit default.

Objects Attributes

All classes define objects, and all objects have properties known as attributes. The `__init__()` method is used to specify an object's original properties by outlining their default value. This method requires at least one argument as the self-variable, which describes the object itself.

```
class Dog:
    # Initializer / Instance Attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

In the following example, each dog has a unique name and age, which is critical to know especially when you begin to define different dogs. Don't forget that the class is only defining the Dog, and not creating objects of individual dogs with unique names and ages.

Similarly, the self-variable also belongs to an instance of the class. Because class instance has different values you can write `Dog.name = name` instead of `self.name = name`.

Class Attributes

While instance attributes are unique to every object, characteristics of a class are the same for all instances. In this case, all dogs.

Methods

When you have attributes that belong to a class, you can proceed to define functions that will access the class attribute. These functions are referred to as methods. When you declare methods, you will want to provide the first argument to the method using a self-keyword.

For instance, you can define a class Snake, which contains the attribute name and the method change_name. The method change name will accept an argument new_name plus the keyword self.

```
>>> # instantiate the class
>>> snake = Snake()

>>> # print the current object name
>>> print(snake.name)
python

>>> # change the name using the change_name method
>>> snake.change_name("anaconda")
>>> print(snake.name)
anaconda
```

Now, you can

instantiate this class with a variable snake and change the name using the method change_name.

Instance Attributes and the init Method

```
class Snake:

    def __init__(self, name):
        self.name = name

    def change_name(self, new_name):
        self.name = new_name
```

You can still provide the values for the attributes at runtime. This occurs by defining the attributes within the init method. Check out the example below:

Now you can proceed to directly define different attribute values for different objects.

So far you know how to define Python classes, methods, and instantiate objects, and call instance methods. These skills will be useful when you want to solve complex problems.

With object-oriented programming, your code will increase in complexity as your program expands. You'll have different classes, objects, instance methods, and subclasses. You'll want to maintain your code and ensure it remains readable. To accomplish this, you will need to adhere to design patterns. These are principles that help a person to avoid bad design. Each represents a particular program that always reoccurs in OOP, and describe the solution to that problem, which can then be repeatedly used.



Chapter 10: Inheritance

We often come across different products that have a basic model and an advanced model with added features over and above the basic model. A software modeling approach of OOP enables extending the capability of an existing class to build a new class, instead of building from scratch. In OOP terminology, this characteristic is called inheritance, the presence of a class is known as the parent or base class, while the new class is referred as the child or subclass.

Inheritance comes into the picture when a new class possesses the 'IS A' relationship with an existing class.

Dog IS an animal. Cat also IS an animal. Hence, an animal is the base class, while dog and cat are inherited classes.

A quadrilateral has four sides. A rectangle IS a quadrilateral, and so IS a square. A quadrilateral is a base class (also called parent class), while rectangle and square are the inherited classes - also called child classes.

The child class inherits data definitions and methods from the parent class. This facilitates the reuse of features already available. The child class can add a few more definitions or redefine a base class method.

This feature is extremely useful in building a hierarchy of classes for objects in a system. It is also possible to design a new class-based upon more than one existing classes. This feature is called multiple inheritances.

The general mechanism of establishing inheritance is illustrated below:

Syntax:

```
class parent:
```

```
    statements
```

```
class child(parent):
```

```
    statements
```

While defining the child's class, the name of the parent class is put in the parentheses in front of it, indicating the relation between the two. Instance attributes and methods defined in the parent class will be inherited by the object of the child class.

To demonstrate a more meaningful example, a quadrilateral class is first defined, and it is used as a base class for the rectangle class.

A quadrilateral class having four sides as instance variables and a perimeter () method is defined below:

```
class quadrilateral:
    def __init__(self, a, b, c, d):
        self.side1=a
        self.side2=b
        self.side3=c
        self.side4=d

    def perimeter(self):
        p=self.side1 + self.side2 + self.side3 + self.side4
        print("perimeter=",p)
```

The constructor (the __init__() method) receives four parameters and assigns them to four instance variables. To test the above class, declare its object and invoke the perimeter() method.

```
>>>q1=quadrilateral(7,5,6,4)
>>>q1.perimeter()
perimeter=22
```

We now design a rectangle class based upon the quadrilateral class (rectangle IS a quadrilateral!). The instance variables and the perimeter() method from the base class should be automatically available to it without redefining it.

Since opposite sides of the rectangle are the same, we need only two adjacent sides to construct its object. Hence, the other two parameters of the __init__() method are set to none. The __init__() method forwards the parameters to the constructor of its base (quadrilateral) class using the **super()** function. The object is initialized with side3 and side4 set to none. Opposite sides are made equal by the constructor

of the rectangle class. Remember that it has automatically inherited the perimeter() method. Hence there is no need to redefine it.

Example: Inheritance

```
class rectangle(quadriLateral):  
    def __init__(self, a, b):  
        super().__init__(a, b, a, b)
```

We can now declare the object of the rectangle class and call the perimeter() method.

```
>>>r1=rectangle(10,20)  
>>>r1.perimeter()  
perimeter=60
```

Overriding in Python

In the above example, we see how resources of the base class are reused while constructing the inherited class. However, the inherited class can have its own instance attributes and methods.

Methods of the parent class are available for use in the inherited class. However, if needed, we can modify the functionality of any base class method. For that purpose, the inherited class contains a new definition of a method (with the same name and the signature already present in the base class). Naturally, the object of a new class will have access to both methods, but the one from its own class will have precedence when invoked. This is called method overriding.

First, we shall define a new method named area() in the rectangle class and use it as a base for the square class. The area of the rectangle is the product of its adjacent sides.

Example:

```
class rectangle(QuadriLateral):  
    def __init__(self, a,b):  
        super().__init__(a, b, a, b)
```



```
def area(self):  
    a = self.side1 * self.side2  
    print("area of rectangle=", a)
```

Let us define the square class which inherits the rectangle class. The area() method is overridden to implement the formula for the area of the square as the square of its sides.

Example:

```
class square(rectangle):  
    def __init__(self, a):  
        super().__init__(a, a)  
    def area(self):  
        a=pow(self.side1, 2)  
        print('Area of Square: ', a)
```

```
>>>s=Square(10)  
>>>s.area()  
Area of Square: 100
```



Chapter 11: Introduction to Django Framework

Django is one of the most advanced Python web frameworks that allows rapid development and maintenance of websites. The Django framework handles most of the pros and cons of web development, so you can concentrate on building your app without the need to build something afresh. Django is free and open-source and has an active community and excellent documentation.

Django will help you to build software that is:

- **Complete**

Django uses the “Batteries included” principle and has about everything that developers may want to do. Since everything you require is part of the one-stop-shop, it works seamlessly together, follows a regular design principle, and has up-to-date documentation.

- **Scalable**

Django features a component-based “shared-nothing” format and can thus be substituted or changed when necessary. Having a clear distinction between the different parts implies that it can scale for increased traffic by including hardware at any level. Some of the busiest websites have successfully scaled Django to achieve their demands.

- **Versatile**

You can use Django to create any website that you want-right from content management systems to social networks and news sites. This framework can work with any client-side framework and can support content in any format.

- **Easy to maintain**

The Django framework is built according to the principles of design that support code reusability. It also allows the grouping of similar functions into reusable “applications”.

- **Portability**

Django is developed in Python which runs on multiple platforms. In other words, you’re not forced to use a specific server platform alone, but you can run applications on any platform of your choice.

- **It's secure**

Developers have nothing to fear that hackers will steal vital information from their websites or gain entry into their website applications. Django has an excellent security system that automatically protects the website.

Where Did Django Come From?

Django was developed between 2003-2005 by a team of web experts who were responsible for building and managing newspaper websites. After making several websites, the team started to reuse most of the standard code and design patterns. This common code resulted in a general web development framework.

Since 2005 when it was open-sourced, Django continued to expand with new releases. Each release brought new functionality and fixed several bugs.

Today, Django is driving the web development environment with thousands of users going for Django. Although it still has some properties that reflect its origin, Django has evolved into powerful framework that can build any website.

What is the Level of Popularity of Django?

There's no readily-available scale of measurement of server-side frameworks' popularity. An excellent question will be if Django is good enough to handle problems of unpopular platforms. Is it still developing? Can you get assistance if you need it? Is there a chance for you to get a job if you learn Django?

Depending on the number of high-profile websites that use Django, the number of people who contribute to the codebase, and the number of people providing free and paid help, Django is a popular framework.

Is it Opinionated?

Web frameworks are said to be “opinionated”, and “unopinionated.”

An opinionated framework is one that has opinions about the “correct way” to do a given task. These frameworks support rapid development in a specific domain

because the proper way to do anything is well-documented and understood. Despite that, they can be less flexible at finding solutions to problems and tend to provide fewer choices for what methods they can use.

On the other hand, unopinionated frameworks have fewer limitations on the right way to integrate components to accomplish a given goal, or even what parts should be applied. They simplify everything for developers to employ the most suitable tools to finish a given task.

In one way, Django is a bit opinionated, and therefore offers the “best of both worlds”. It provides a set of components to take care of web development tasks and one preferred style to use them. Despite that, Django’s decoupled architecture implies that you can select and choose from different options, or include support for new ones if necessary.

Who’s Using Django?

It is also good to know who is using this particular framework out there so that you can get an idea of what you can accomplish with it. Some of the most important sites using Django include Bitbucket, Mozilla, Instagram, National Geographic, Last.fm, and many more.

For more examples, you can navigate to the Django sites database; they have a list of over 5k web sites driven by Django.

8 Unique Characteristics of Django

This section will explore the unique properties of Django. The features of any technology can explain what it is best created for. Any time you learn new technology is because of a given reason. There are countless reasons as to why you should learn Django. Here is a more in-depth look at the unique characteristics of Django, and why it is such a robust framework.

1. In-depth Documentation

If there is an open-source framework with quality documentation, then Django comes first. No doubt, Django has the best documentation in the market.

Proper documentation is vital to any developer. This is like a well-documented library where you can search for anything and find whether it is a syntax or function.

The documentation for any technology is again one of the best features to rate technology. As such, it allows other developers apart from its creators to optimize the technology.

Django has been a leader at spearheading the best documentation from the start when it was made open source to the present date. In fact, the documentation has only been getting better and better with the active development of technology, and it is also available in multiple languages.

2. SEO Optimized

This is an excellent feature of Django, and makes it competitive over others. Search engine optimization is a practice of making your website highly visible to search engines. In this case, it appears in the top results when users search for Django. Search engines rely on an algorithm that sometimes may fail to be in line with the developer. However, because the website is created for humans to read and understand, and the URL from on the server too.

Django polishes the above idea by maintaining the website via URLs instead of IP addresses on the server, which increases its visibility.

3. It's a Python Web-framework

One of the primary reasons that have pushed a lot of people to learn Django is because it's a Python framework. Python is a language that solves all your problems and in any situation. It's straightforward and easy to use language. Currently, Python is the most popular language in the market. And that is because it is the most straightforward language to learn. You can use Python in almost everything, from web-development to machine-learning.

These beautiful features make Django and Python the most powerful and easy to learn the framework.

Yet you need to have some basic knowledge of Python and web-working to begin developing using Django. It provides fast development, and it accomplishes so by being logical and straightforward.

4. Highly Scalable

Most of the MNCs use Django, and it's applied without any problems. This is an excellent example of Django being scalable.

Scalability refers to the level at which the technology is implemented. For larger websites such as Instagram, there are many active users, which creates massive data. This kind of level demands application to be error-free and accurate. Of course, it's difficult for programmers and developers that have a lot of experience.

Expert programmers design Django from scratch without borrowing any Python library other than what the developers build themselves. Multiple tests and debugging and now with enough time on the market side have made Django perfect for anyone who wants to make their websites error-free.

5. Highly Versatile

Django is versatile in nature. The logical project structure and Django architecture sometimes may appear limiting. However, that is just the opposite because by presenting us the files, it is creating a solid ground that can then be transformed to whichever application we want to build.

It enables Django integration with all the technologies we use and other upcoming ones. As a result, Django is the property of web development and everyone who was initially using PHP will mainly go for Django.

6. Intensively Tested

Anytime you're learning new technology; you expect it to robust and durable enough to handle dynamic changes taking place in the industry. Well, Django accomplishes that task with significant perfectionism.

MNC across the world uses Django to build projects so it's right to say that it works well to deal with all the traffics and accomplish international standards.

The framework has been in the industry for some time, and lots of bugs and errors have been fixed. So, this is the right time to start learning the Django framework. For that reason, the number of developers who use Django for web development keeps expanding every day.

7. Supports Rapid Development

While most technologies support this particular feature as the main feature, Django has many other great features.

In this case, rapid development implies that you don't need extensive backend knowledge to build a complete website. You will also not build separate server files to design the database and connect the same while also developing another file for data transfer to and from the server. Django takes care of this work and many other tasks. You don't need additional files for each task.

8. Provides Better Security

Django is a super-secure language. It's a secure language because it explores all loopholes automatically which were once left for a backend developer to handle. While it is hard to note, but experienced backend developers can understand the nature of security and work achieved by Django.

The development of Django's code is right from scratch and that accounts for its other properties as well as security. Since the framework is designed by web experts, you can expect that most of the problems faced by web developers are addressed.



Chapter 12: Django Installation

The first thing you need to do is to install a few programs on your machine so you can be able to begin using Django. The basic setup involves installing Python, Virtualenv, and Django.

Briefly:

- You first need to install Python
- Then you can proceed to install the virtual environment that will help you isolate your Python and Django projects.
- Next, you should install Django within the virtual environment. This way, you can isolate your project.

Install Python

The first thing you need to do is install the latest Python distribution. You can go to early chapters to learn how to install Python, or go to www.Python.org and scroll down until you see the download files listed.

Select the correct version based on your Windows distribution. If you're not sure which one is perfect for you, the possibilities are you want to download the Windows x86-64 executable installer version.

Navigate to your Downloads directory, right-click on the installer and select Run as administrator.

After Python installation, the next thing you need to do is search for the Command Prompt program and open it.

To check whether everything is working fine, enter the following command:

```
Python - -version
```

The output should display the Python version you've installed on your computer.

If you can see the version of Python installed on your computer, Congratulations, Python is running. Now is time to install Virtual Environments.

Installing Virtualenv

In the next step, you'll use pip, this tool will help you control and install Python packages. In order to install virtualenv, type the following command in the Command Prompt.

```
pip install virtualenv
```

Until this point, the installation that have occurred consists of system-wide. Moving on, everything you install, including Django itself, will be installed inside a Virtual Environment.

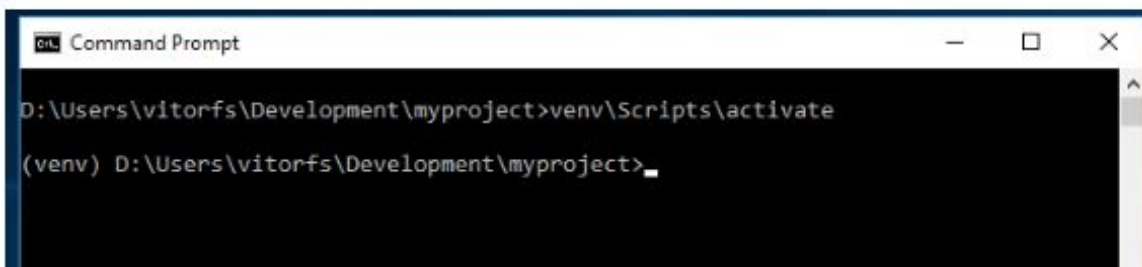
Now that you have your virtual environment installed, everything that you're going to install from now will go right into a Virtual Environment.

Consider this: for every Django project you start, you will first build a virtual environment for it. This is like creating a sandbox for every Django project. So you can play around with it, without damaging anything.

After you create your virtual environment, you need to activate it before you use it.

Type the following commands to activate it:

```
venv\Scripts\activate
```



You'll know that it worked if you see (venv) in front of the command line. For example:

Another critical thing is that pip program is already running, and when you use it to install a Python package such as Django, it will be installed within the Venv environment.

Next, in case you want to deactivate the venv, you should execute the following command:

```
venv\Scripts\deactivate.bat
```

Well, but let's have it activated for the next steps.

Django Installation

The installation of Django is a straightforward step. Now that you have the venv active, you can type the command below in the Command Prompt.

Pip install Django

That's it. You are now set up. The next step is to start the Django project. Now is time to build something.

Starting a New Project

To start a new Django project, type the command below:

```
django-admin startproject myproject
```

After you execute the above command, it will generate the base folder structure for a Django project.

So far, your myproject directory is composed of the following:

```

myproject/                                <-- higher level folder
|-- myproject/                            <-- django project folder
|   |-- myproject/
|   |   |-- __init__.py
|   |   |-- settings.py
|   |   |-- urls.py
|   |   |-- wsgi.py
|   +-- manage.py
+-- venv/                                <-- virtual environment folder

```

Django Apps

The Django philosophy has two major features:

- App: this is a web application that performs something. An app is made up of a set of models, templates, test, and views.
- Project: this is a collection of configurations and apps. A single project is made up of multiple apps, or a single app.

It's important to understand that you can run a Django app without the presence of a project. A simple website such as a blog can be powered inside a single app, which can be named weblog.

Django embraces the concept of an app.

This is a way to arrange the source code. At the start, it's not trivial to determine what's an app or what is not. How you can organize the code, and so forth. However, don't worry about that right now! Let's first familiarize ourselves with Django's API and the fundamentals.

Well, so to demonstrate, let's build a simple Web Forum Board. To develop the first app, navigate to the directory where the manage.py file is and run the command below:

Django-admin startapp boards

Notice that startapp command is used this round.

This will present us with the following directory structure:

```
myproject/
|-- myproject/
|   |-- boards/                <-- our new django app!
|   |   |-- migrations/
|   |   |   +-- __init__.py
|   |   |-- __init__.py
|   |   |-- admin.py
|   |   |-- apps.py
|   |   |-- models.py
|   |   |-- tests.py
|   |   +-- views.py
|   |-- myproject/
|   |   |-- __init__.py
|   |   |-- settings.py
|   |   |-- urls.py
|   |   |-- wsgi.py
|   +-- manage.py
+-- venv/
```

Now, that you have created the first app, let's configure the project to use it.

To accomplish that, open the settings.py and try to search for Installed_apps variable:

Settings.py

Django already has 6 built-in apps installed. These apps provide common functionalities that many Web applications require, such as sessions, css, static files

management, etc.

You will learn more about these apps in the later chaps. But for now, let them be and only add your boards app to the list of `INSTALLED_APPS`:

Hello, World!

Let's create the first view. You will learn more about in the next chapter. But for now, let's check out how it feels like building a new page using Django.

First, open the `views.py` file within the boards app, and include the following code:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse('Hello, World!')
```

Views are Python functions that accept an `HttpRequest` object and displays an `HttpResponse` object. Accept a request as a parameter and return a response as a result. That's the flow you need to remember.

So, here is a simple view definition known as `home` which returns a message Hello, World

Next, you have to instruct Django when to serve this view. It happens within the `urls.py` file:

```
from django.conf.urls import url
from django.contrib import admin

from boards import views

urlpatterns = [
    url(r'^$', views.home, name='home'),
    url(r'^admin/', admin.site.urls),
]
```

Urls.py



Chapter 13: MVC Pattern

The previous section dealt with the installation of everything you need. Hopefully, you're all set up with Python and Django. We already created the project you're going to use.

The MVC Pattern

This section will review the MVC pattern. You will understand more about the MVC pattern. Django MVC architecture solves many problems that were available in the traditional approach for web development.

For every website on the internet, it has three main components; Input Logic, UI logic, and Business Logic.

These sections of code perform different roles, the input logic within the dataset, and how the data organizes in the database.

It merely accepts input and directs it to the database in the right format. Business logic is the primary controller that takes care of the output from the server within the HTML. The UI logic as the name goes, refers to the HTML, CSS, and JavaScript pages.

When the traditional technique was used for running all this code was implemented in a single file. This was not a huge problem back in the time, the web pages were highly static, and websites didn't feature multimedia and extensive coding. Additionally, this architecture presents challenges for developers during testing and project maintenance.

Now, the times are different, and the websites are getting larger and larger every day while supporting applications such as online artificial intelligence, cloud computing, and online development environment, these projects are all built within the MVC architecture.

Well, what is MVC? This is a short form for Model View Controller. Don't be scared, and you'll learn every feature of the MVC pattern and associate it to Django.

The MVC pattern is one of the Product Development Architecture. It computes the traditional approach' challenges of code in a single file. There are three major parts of the MVC. That includes the Model, View, and Controller.

The difference between these parts allows the developer to concentrate on one major feature of the web-app and thus, robust code for one function with significant testing, scalability and debugging.

Let's look at the components:

1. Model

The Model represents the section of web-app, which works as a mediator between the website interface and the database. In technical terms, the object executes the logic for the application's data domain. There are moments when the application may only consume data in a given dataset, and directly send it to the view without requiring any database then the dataset seen as a model.

While nowadays, if you want any website you need to have some type of database because its extra user input even if you're building a simple blog site.

The Model refers to the component which carries the Business Logic within the Django architecture.

For example:

Once you sign up on any website, you'll be sending information to the controller, which later transfers it to the models which have business logic and stores in the database.

2. View

The View section contains the UI of the Django framework. The View is made up of HTML, CSS and other technologies. Overall, the UI develops from the Model components.

3. Controller

The controller is the engine part. In other words, the controller takes care of the user interaction and clicks a view based on the model.

The main function of the controller is to choose a view component depending on the user interaction and apply the model component.

This architecture has numerous benefits, and that's why Django is built on this architecture. It takes the same model to a complex level.

For example:

When you combine the previous standards, you can clearly understand that the component which is choosing different views and transferring the data to the model's parts is the controller.

MTV Pattern

MTV stands for the Model-Template-View framework. It carries the Templates terminology for Views and Views for Controller.

Templates are connected to the View within the MVC pattern because it describes the presentation layer that handles the presentation logic in the framework and controls the content to display and how to present it for the user.

Advantages of Using the Django Architecture

The Django framework is developed on the above architecture, and it communicates between all these three parts without the need to create complex code. That's the reason why Django is becoming popular.

There are various advantages that Django architecture comes with:

1. Rapid Development

This Django architecture divides into different components that simplifies it for numerous developers to work on separate features of the same application simultaneously. That is one of the properties of Django.

2. Loosely Coupled

The Django architecture has different parts that depend on each other to varying stages of the application at every point that enhances the security of the general website.

3. Easy to Make Changes

This is a great feature of development because there are different sections in the Django architecture. If there is a change in section parts, you don't need to change it in other parts.

This is one of the key features of Django because it provides more adaptability to the website than other frameworks.

This section has explored the MVC pattern of the Django framework and described the parts in detail. You have also learned some benefits of the Django framework.

Django Project Layout and Different Files Contained in Root Directory

When you build a Django project, the Django framework develops a root directory of the project using the project name. This root directory contains files and folder, which deliver the basic functionality to your site and on that robust ground you'll be creating your complete website.

Files That Make Up the Django Project Root Directory

The root directory refers to the root that carries the manage.py file. Additional files such as db.sqlite, are database files that may be available when you migrate the project.

The Django root directory has the default app which Django provides to you. It features the files which will be relevant in the entire project.

The files within the root directory contain specific functions, and once you understand them Django will appear more logical to you. All these files are meant for completing special functions and are in a very reasonable order.

Let's start with the manage.py file:

1. Manage.py

This is the file that holds details of the command line for the project and you will use this file to run, debug, and test the project.

The files stores code for running the server, migrating the server and controlling the project using the command-line.

The above file provides all the functions just like Django-admin and it also delivers certain project functionalities.

While building your project, you will make use of the following commands often:

Runserver: You will use this command to run the test server in the Django framework. This is also one of the advantages of Django over other frameworks.

Makemigrations: This command is useful for combining the project files you have added in it. This command will basically review for any new additions within your project and add that to the same.

Migrate: The last command is to include those migrations you have engineered in the last command with the entire project. The difference between this command and the previous one is that the initial command is used to save the changes within the file, and the later one applies that change to the entire project.

2. My_project

This is a Python package for the project. It contains the files for configuring the project settings.

`_init_py`

This particular file is empty, and it's available in the project for the main reason of letting the Python interpreter understand that the above directory is a package. That's one of the standard guidelines for Python packages.

`Settings.py`

The settings.py is the primary file where you will be adding all applications and middleware applications. As the name goes, this is the main settings file of the Django project. This file has installed applications and middleware information which are installed on this Django project.

Any time you install a new application, you will be adding that in this file.

Notice that there are certain pre-installed applications. By default, these applications provide all the basic functionality you will ever want for your website such as the Django admin app.

Urls.py

This file features the project level URL details. The URL is the general resource locator and it presents you with the address of the resource plus other components for your website.

The major function of this file is to link the web-apps with the project. This urls.py file will execute anything that you'll be typing inside the URL bar. Then, it will relate your request to the particular app you connected it to.

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

In the above example, by default, the file adds one URL to the admin app. The path () accepts two arguments.

First, the url to be looked inside the URL bar on the local server, and secondly, the file you want to run when that URL request is matched. The admin is the pre-made application and the file is URL's file of that app. This file acts as a map of your Django project.

Wsgi.py

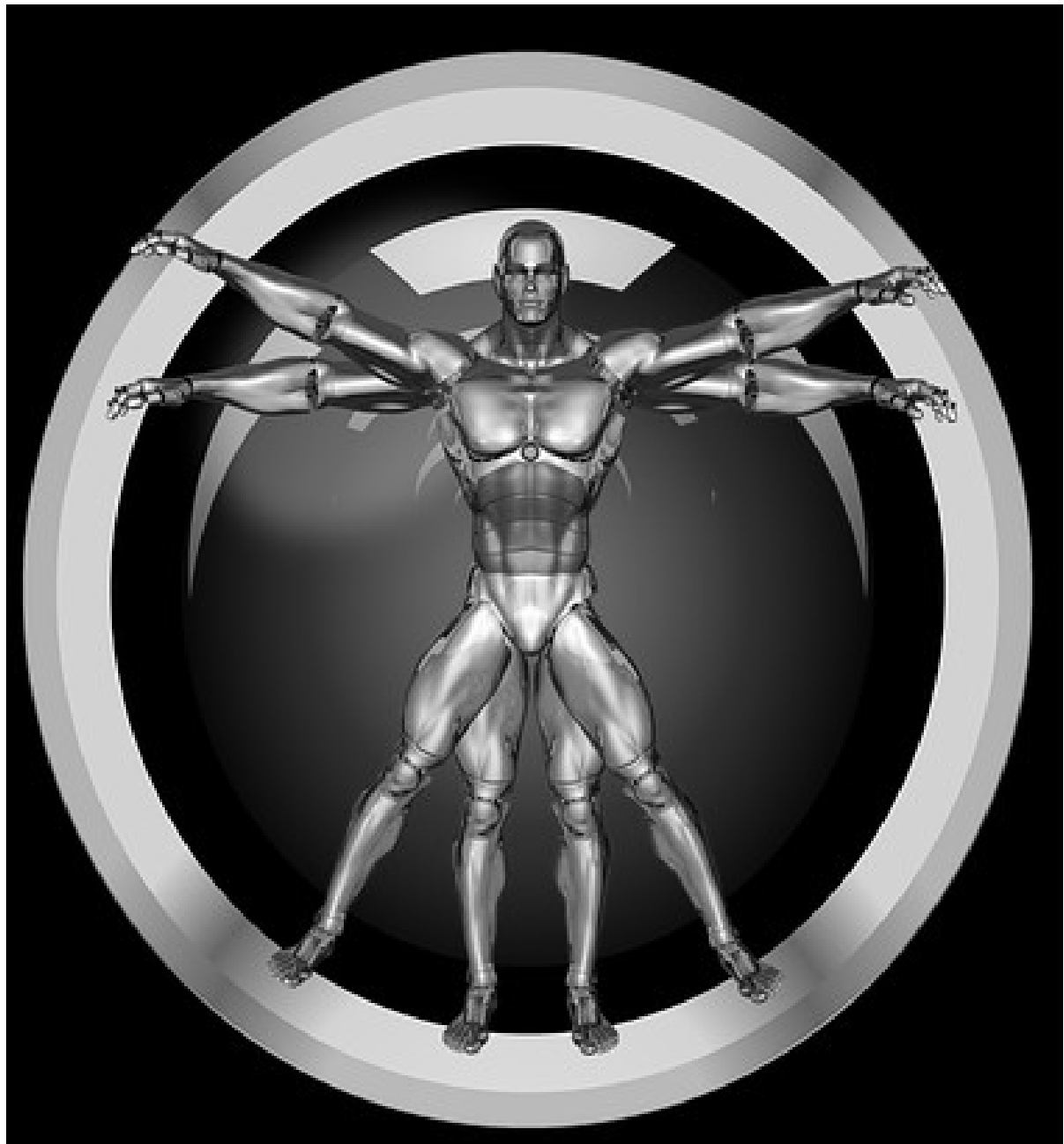
Remember that Django is developed using Python, which relies on the WSGI server for web development. This file handles that and you won't be using this file a lot.

Wsgi is still significant though if you want to run the applications on Apache servers, or a different server because Django is backend, you will require its support with various servers. However, you should not worry because, for each server, you will get a Django middleware which will solve all the connectivity and integration challenges.

You only need to have an important that relevant middleware for your server. It's that simple.

This last section has explored more about the Django project layout and the way all files in the Django architecture align, and what's their purpose.

These files are fundamental and will be available in any Django application you build individually. Their primary function is to make available all the backend support and resolve connectivity problems. It takes place while you handle the database creation. It also handles frontend and the uniqueness of your website or web-application.



Chapter 14: Create, Install, Deploy First Django App

In this chapter, you will learn how to create, install, and run your first Django app. Furthermore, you'll learn how to add the app in `urls.py` file and build `views.py` file for the Django project.

Let's dive in.

1. Creating a Django App

The main reason web applications are used is to make use of the Django code's reusability property. This will allow you to not only migrate the pre-built apps within the project but also personalize web applications.

All the commands are available in the root directory or within the directory where there is the `manage.py` file.

Additionally, before you build Django application, run this command on your system within the project directory.

```
python manage.py makemigrations
```

Once this command is finished executing, run the following command:

```
python manage.py migrate
```

Building a Django project is simple because you only need to write some commands that will be implemented on your system.

In the case below, both terminal and PowerShell users share the same commands:

```
$ django-admin startapp application-name
```

Now, running these commands will create the application. If you look at the root directory, you will see files that the new application will require, and you will be editing them to accomplish set goals.

Here are some tips that might help you:

- Name your applications based on different tasks that they're going to execute.
- Build applications if the task can be accomplished using a separate application.

If you adhere to the above tips, you will create modularity within your project. This habit will not only help you in general practice, but anytime you will be building future projects you can be specific as to what kinds of apps to include in your current project. Hence, boosting your development speed and reducing the workload without losing any quality in your work.

Below are files that will come pre-installed when your application is developed and the formats for every application file doesn't change.

2. Install Django App

Once you have defined your application, the next step is to install the application.

So, you need to open the settings.py file and change it.

Keep in mind that you will be changing the main settings.py file of your Django project, not the one within the app directory.

You will only need to type your app-name in the INSTALLED_APPS list once you add the application in the list of Installed Apps.

Once you complete that, congratulations, you have installed custom Django application.

Now, you will need to include the app to your urls.py file so that anyone who searches the URL of the app within the browser will open the above app.

3. Adding App in urls.py file

```
1  from django.urls import path
2
3  from . import views
4
5  urlpatterns = [
6      path('', views.index, name='index'),
7  ]
```

To include the app in the urls.py, you will need to write some code. First, you need to

define a new Python file in the demo directory and paste the following code in that file.

In the above code, you are passing a message to the Django project that you've to trigger this function in the views.py file.

Don't be scared; you'll be changing that file too.

In the above code, you've imported the django.urls package and path function from there.

The path () function is new functionality in Django 2.0. If you have any previous versions, then it's better to upgrade.

This code passes two arguments, first, there is the URL which was searched and relayed to the URL bar from the browser. The other argument is to run the file or function, which is an index function.

Now, you'll be changing the urls.py file in the main Django project.

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

There is only a single item on the list and you'll need to add the new data.

After including the apps urls.

```
urlpatterns = [  
    path('demo/', include('demo.urls')),  
    path('admin/', admin.site.urls),  
]
```

By completing the following step, you'll be making the system direct your server to search the URL for the demo keyword. Then next, direct the URL to the urls.py file within the demo application directory.

4. Building the views.py file

Lastly, you'll create the views file of the Django project. This file will create a view on the browser.

Just use this code snippet, it's already in the demo directory and views.py file.

```
from django.http import HttpResponse  
def index(request):  
    return HttpResponse("DataFlair Django Tutorial<html><body><h1> Hello World DataFlair Dango tutorials</body></html>")
```

Going by its name, this is the file where you'll create views that the browser will render.

As you can see, you will be importing a function of HttpResponse and also create a function called index, which has been used in the urls.py file.

That function accepts the request the same manner a server would interact with the server. You'll be returning HttpResponse() and within that, anything that you include in the argument of this function will be deployed by the browser.

There are no extra responses besides the HttpResponse and you will be using that in the future.

If you have followed everything until this point, then you've just created your first Django App and webpage.

If you like, you can proceed to open the admin app, but that will be covered in the next section.

localhost:8000/demo/

If

your app doesn't start, then you will need to run the following command in the URL:

This section has taught you more about Django file structure, and how to create your first app. Don't worry if the app failed to start there could be some errors. Just go through this section again.

Django Admin Interface

Django Admin interface is the most prominent feature of Django Framework and the convenience it delivers to the site maintainers is powerful. Additionally, it is a vital part of the Django Framework and will benefit you in many different ways.

First, let's explore the Admin Interface of the website.

The Admin interface as the name suggests is the engine of your project. This is used by site administrators to add, delete, and modify the content on the site. Besides, it's used to maintain other functionalities created by developers.

Well, there is a tricky part, admin interface is the main controller, it's required to be coded by the developer individually. You need to handle everything right from database connection to enhancing security of the admin interface because anyone who uses it is a controller, and any loophole left will be devastating for your website.

Interacting with Django Admin Interface

Django Admin is the kind of interface designed to meet all the needs of the developers. Its language is a bit generalized.

The Django Admin has a complete interface and you won't need to worry to write an Admin interface for your project now. To start with, it is better to master the process of setting up Django Admin site because you will need it in your future projects.

Getting Ready Django Admin Site

Here are steps you should follow to set the Django Admin Site and run user models

1. Start the Django server, then search in the URL bar:

localhost:8000/admin/

You should see an image appear, and the site should ask for the username and password. Now, you can proceed to create one.

2. Open the PowerShell or terminal and break the server by pressing CTRL +C.

Next, type the following command:

```
python manage.py createsuperuser
```

3. After you run the above command, it will ask you to enter a username, which should be your preferred choice. Then it will request you for the email id, and password. So, you should be able to complete the details without any problems.

There are certain inbuilt security features such as if your password is numeric it will generate a complex password, and many such security problems have been fixed up for users.

Superuser

A superuser refers to the details of the admin for your website, which can complete any data manipulation and content manipulation connected to your website.

Additionally, you should never reveal or share the superuser details with anyone because it is the key driver of your website. When the website is active, it is crucial for managing and controlling your website.

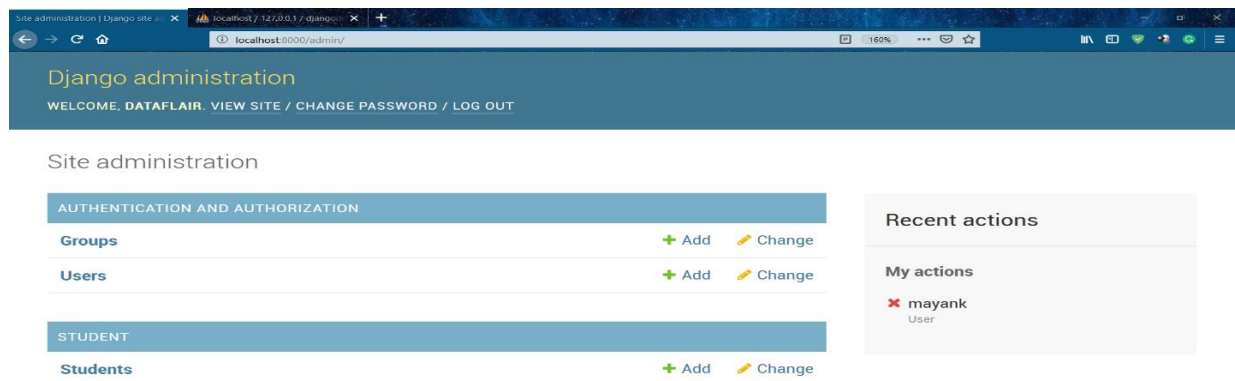
After you successfully build the superuser, you will log into that admin page to start your Django server once more.

1. Login in admin using the details you've created.

You will see the options and a beautiful looking interface:

Since the custom applications aren't registered with the Django admin interface, it will not show any of them. There are only two fields here, and you have two options with each of them.

2. Now select the users' field, and you should see something like this.



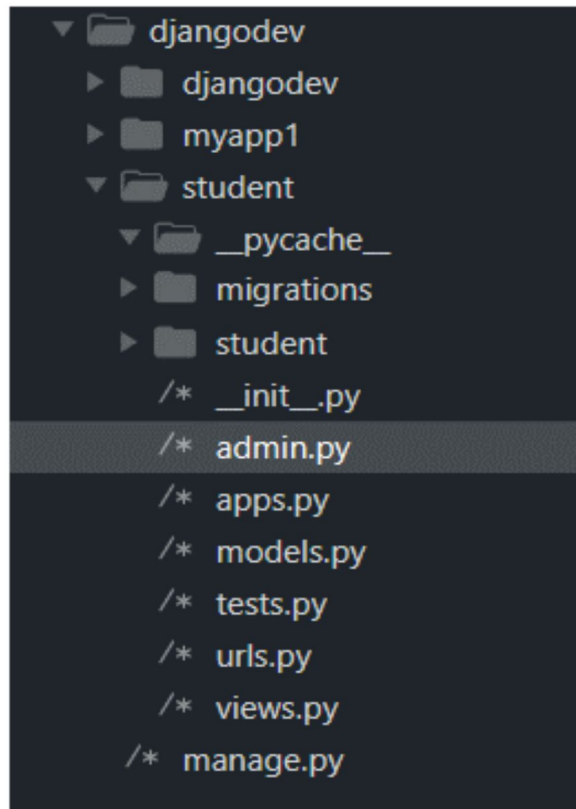
Here, you should see only one user and the details which you inserted inside. Though you wouldn't be able to see the password. That is an advantage from the Django's side, all passwords are encrypted using a unique key in the settings.py file.

3. Confirm whether the user you created was saved.

4. Now open phpmyadmin and open the Django's project database which you developed. Next, if you open auth_users table, you will see the details as it is except for the password.

You will also see that there are additional tables that you didn't create but are inside the database. Well, don't be worried, these are all Django.contrib package's imported models tables. The Django.contrib refers to a package for Django which is similar to the relation of standard library using Python. Almost all the functionality required is on the Django.contrib package.

5. Navigate to home in the Django Project.



6. Next, open the file `admin.py` of your app. The structure of the directory should look this way:

7. Open the `admin.py` file and enter the following two lines:

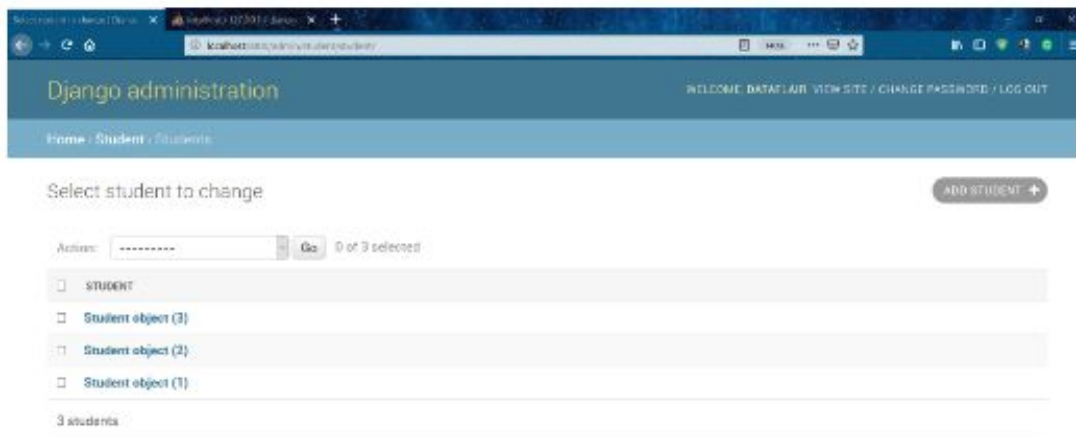
```
from .models import Student  
  
admin.site.register(Student)
```

The first statement is used to import class `Student` or the custom model created.

The second line passes the model or registers the model to the Django Admin interface.

8. Now, open admin again, and it won't request for a password.

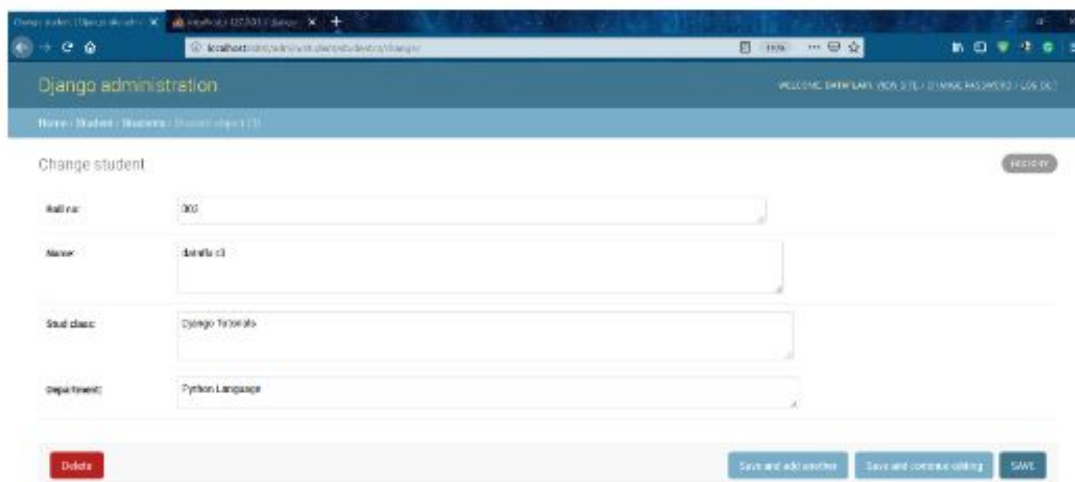
You have the student model in admin and now open that student field.



You'll see a list of students you have, and you can view a vital option there such as add a new student.

Adding a new student will create a new record in your database; all of it is the role of the Django Admin interface that combines with Django ORM.

9. Next, click on any student-object



Here, you can directly access and modify the information without even heading into the database.

Also, you should see various options such as save and delete.

See the power it grants you without creating the Django Admin interface itself. In other words, it's easy to use plus it's more secure than anything you've done for the

same.

If you have many developers working on the same task, then more interface functionality will be necessary. You can decide which developers get what privileges without typing anything, you only select.

In summary, the Django Admin Interface is a great tool but it is also true that certain websites may not require an admin interface because of their nature. However, that will not be too scalable or won't be for commercial purposes. If you want quick content management without the option of reinventing the wheel, then Django is the best framework to select, and the best admin you can get with many features as a developer.

This chapter has taken you through the steps of building a superuser and how you can integrate your project with the Admin interface. You should try to deploy various models with admin and then you'll become familiar with it.

Django Database

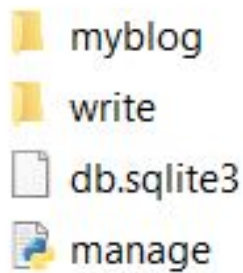
This section will look into how to connect databases with Django project.

Anytime you're building a web project or any type of project, you will need some form of input by the end-users or consumers. Now, all that data entered by the consumers take care of by databases. In the modern era, it's impossible to develop a website without building a database. Even if it's a blog site, you will still need a database.

To accomplish that you will require to have software. The function of this software is to store that data efficiently and also some middleware that can allow you to communicate with the database.

Connecting Your Database with Django Project

By default, when you created the first app and started the server, you probably saw a new file in your project directory. This file is called 'db.sqlite3'. The file is a database file where all the data that you will generate will be kept.



This file is automatically generated because Django has a default setting of the database set to the SQLite. While it is useful for testing and provides many properties, if you need a scalable website, you should change it to an efficient database.

In this case, you will be using MySQL, and this section will teach you how to integrate your project with MYSQL.

Databases Dictionary Indexes

```
# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

First, open the settings.py file in your web-application/project, and you will see the following part:

A database is a pre-defined dictionary in Django Framework having the 'default' as the index value for the significant database where all data is kept.

You can have multiple databases because you need data backups, but a default database is only one, although you will not be adding numerous databases now.

The default contains a dictionary where there are 2 indexes:

Engine

It describes the library to be used once connected to a particular website. The value should hold "Django.db.backends.sqlite3". This is a Python library for sqlite3 database and will transform your Python code to the database language.

Therefore, you don't need to master any new database language, and you can find every code in Python.

Name

In this part, you've got the name of the database you're using and the path of the database. This parameter shifts based on the kind of database you're using. Here you can test the database file.

You will also pass the name of the database file or if the file is not available, this will define the db.sqlite3 file. In case you change the name to db1.sqlite3 or anything of your choice, it will generate that file in your root directory every time you run the server.

As you can see, there is no big struggle when it comes to database creation using Django framework. Remember, every database has features that become the default dictionary indexes that you can create based on the database you are connecting to.

2. mysql and Django

MYSQL is a robust database that comes with multiple features and provides flexibility. Here's how you can integrate it into your project.

1. First, install Xampp

Xampp is a free opensource tool which comes with the Apache server and PhpMyAdmin which is the right source for beginner programmers to work with MySQL.

To download XAMPP, click the following url
<https://www.apachefriends.org/download.html>

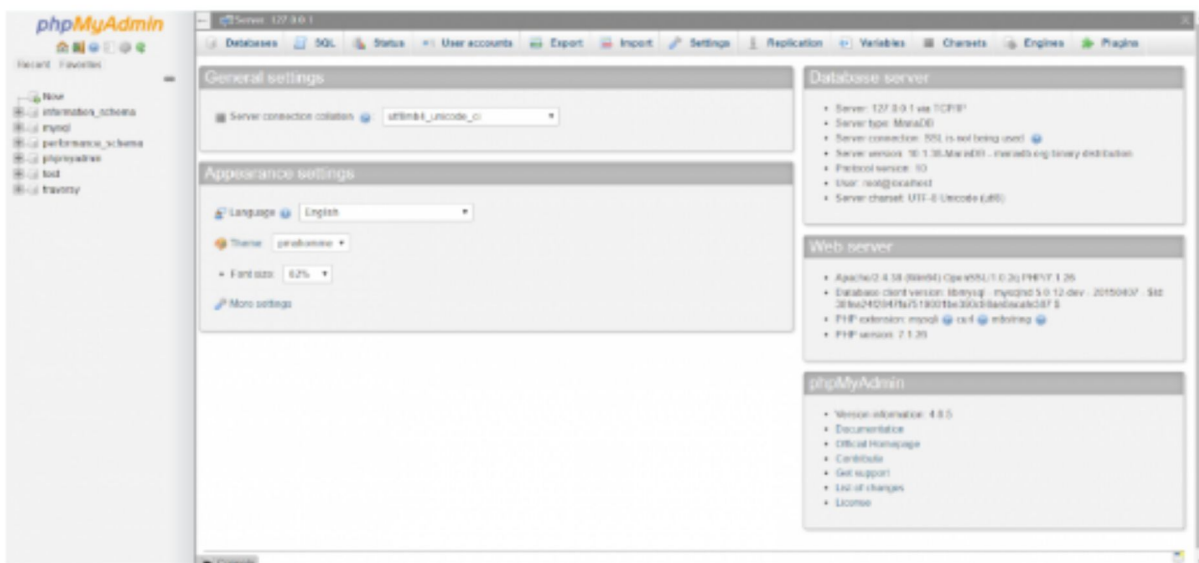
2. Open the Xampp control panel

After installation, you will need to open the Xampp Control Panel and start 2 services there: Apache and MySQL.

First, run the apache server, and then the MySQL server.

Modules								
Service	Module	PID(s)	Port(s)	Actions				
<input type="checkbox"/>	Apache	16256 16024	80, 443	Stop	Admin	Config	Logs	
<input type="checkbox"/>	MySQL	9660	3306	Stop	Admin	Config	Logs	
<input type="checkbox"/>	FileZilla			Start	Admin	Config	Logs	
<input type="checkbox"/>	Mercury			Start	Admin	Config	Logs	
<input type="checkbox"/>	Tomcat			Start	Admin	Config	Logs	

Simply click on the start button, and you'll see the following image:



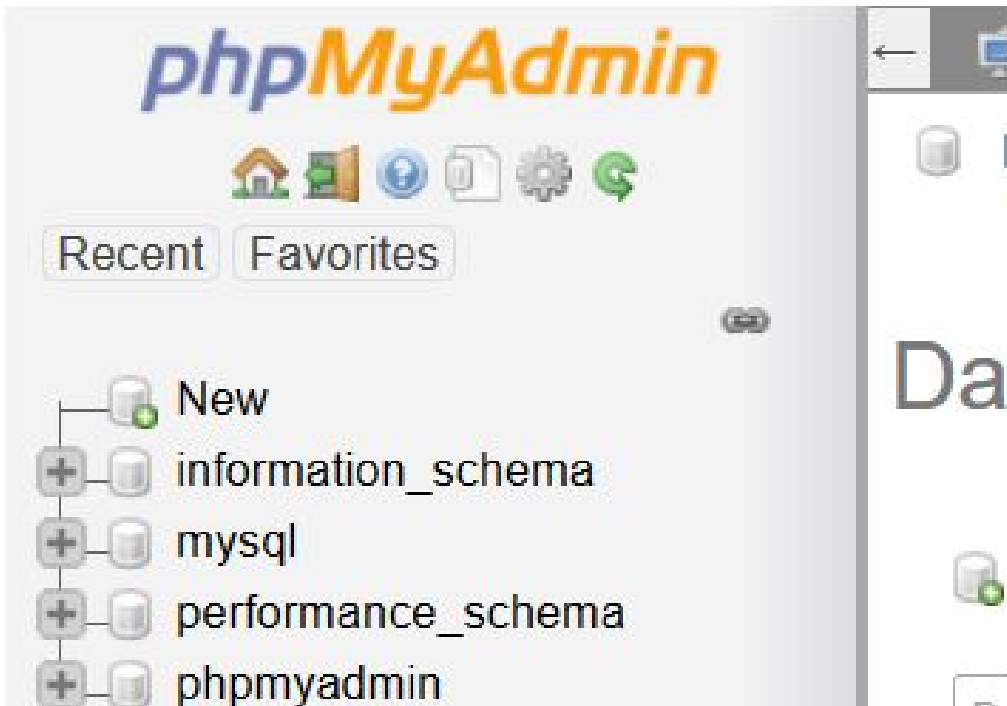
Next, click on the Admin of the MySQL Service, that should display a web page.

This is the main page where you will be monitoring your database.

The advantage of Xampp is that it has a powerful interactive environment, and the time of deploying your model, that too will take place here.

So, you'll get an efficient database for your project. That's quite easy.

Simply click on the new button as shown below. Then, fill the desired name of your database and click on the create button.



Once you're done with that, it will add your database in the list.

That's it, now you don't need to do anything here. You will only be working with Python and the models component of Django will prepare everything.

3. Changing settings.py

In the following last step, you will be changing the dictionary of the database in the main project settings.py.

First, you should install the following file by typing the following command in the command line:

Entering this command will install the Django code for connecting the MySQL Database.

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'dataflair',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
        }
    }
}

```

Next,

exchange this code with the DATABASE dictionary in settings.py.

In this case, the attributes are higher in number as Mysql provides additional features from the sqlite3. The engine, in this case, is “Django.db.backends.mysql” which has the same name as the Python library for MYSQL.

It's good to leave the password empty so that to avoid any errors.

In the above example, the HOST is the host server, but when left blank it takes the localhost as the default.

OPTIONS is an interesting feature. Here, you're merely passing the SQL as a string via Python, which later the SQL server parses itself.

```
SET sql_mode = 'STRICT_TRANS_TABLES'
```

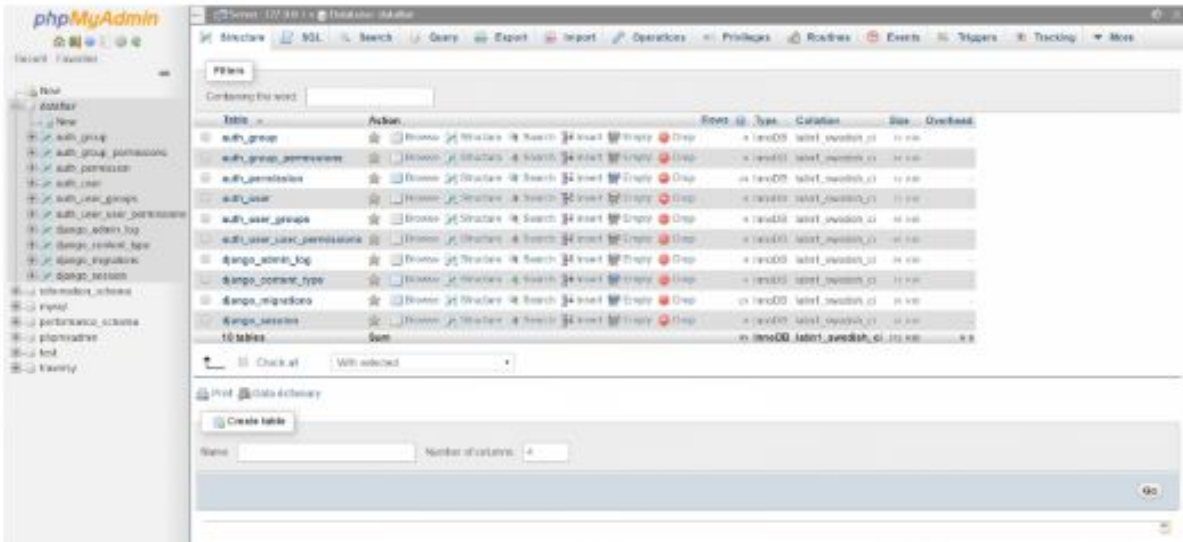
Here you need to write:

Basically, this is SQL being passed on as a string.

```
python manage.py migrate
python manage.py runserver
```

For the final steps, execute the following commands:

Next, refresh the phpMyAdmin page, and you will see some tables show up.



You're done! Now you have all the things ready for your website. The components and parts setup is finally over.



Chapter 15: Tips for Django and Python Beginners

So, you've made up your mind to learn Python or Django. Or maybe you're already learning the language and the framework. Here are tips that will help you speed up your learning.

1. Master the Rubrics

You must get your primary down. And thanks to the syntax of Python, you can use Django without being an expert in Python because it is amicable for new programmers. Python code resembles the English language.

Despite its simple syntax, you cannot ignore the basics. Mastering the basics, it the path to fully understanding Django. So, ensure that you learn the basics as your first step to becoming a Django pro.

2. Begin to Solve Problems

You've mastered the basics. You have the foundation of the Python language. Now is the time to start working out small issues.

Come up with a function that can compute taxes. Define a loop that asks the user to enter his or her age until the user stops the loop and computes the average of all ages

There are certain websites where you can take on small tasks. Then you earn points depending on the answer, and you start to graduate. One such website is the CodeWars.

3. Create small, simple projects

If you can complete small problems, why not start building fun projects?

If you have learned how to use Python to solve more significant problems, this is the time to give yourself a tougher challenge.

The idea is to build small projects in Python, and slowly increase the level of difficulty. However, don't try to create a Youtube kind of project on your own.

Just start to build something simple, something you can enjoy to work on. A game of tic-tac-toe or a guess the number game.

Once you're done, proceed to build another one. Or start adding new features on the one you've built.

I know it's going to take your time, you will need to spend a lot of time Googling, but that's how the best developers start.

Start with a Flask

If you have never learned any programming language or web development, and you want to learn Django, a great idea is to begin with a flask.

Flask is great option for Django. It's more straightforward and easier to understand.

Flask is good for the introduction, as it has more functionalities, but they need to be implemented by you.

With flask, what you see is what you get. Nothing else. And it's pretty damn easy.

Dive into Web Development with Django

Once you master the basics of Python language, and you completed your first small programs. Now, you're ready to learn Django.

Master New Django Concepts

You have learned how to create a project in Django. You understand how the Model View Template works. Well, but there is a lot more to learn. So much more.

The secret is to learn new concepts such as Signals, Logging, Testing, and so on.

Remember, this is not a race. Spend time to read the documentation. Try it. Don't even attempt to memorize. But learn how it works. What elements are moving, why should you apply that concept, and when should you use it.

And next....

Start Applying Those New Concepts

Immediately you learn something new, use it. There is no perfect way to internalize something than applying it right away.

Develop a small pet project where you can use it, or better, add it to the current one.

Read Code from Experts

If you want to become an expert Django developer, then you need to read code from professionals.

The internet is an excellent resource for information. Why don't you get Django Github projects to study?

You can still download the project and improve it. And to accomplish this, you need to read, read, read, and master the code.

Surround Yourself with People Learning

When you surround yourself with people who are learning the same language as you, it provides you with a sense of camaraderie. In other words, you're all moving towards the same goal, and each one can depend on each other. It could be that your problem was solved a few days ago by one of your friends. Why not find out from him how he arrived at the answer.

Surround Yourself with Experts

You need both your peers and experts to become the best. Your friends, workmates, and mentors that have been using Django or Python can solve complex problems that beginner developers can't.

A professional can solve your problems and provide you with a bigger picture.

A beginner may describe to you how to do something, but only an expert can tell if your current solution is optimal or not.

Read Lots of Django Books

Sometimes it can be hard to surround yourself with experts, or you may not find time to interact with them because of the nature of jobs they handle.

Thankfully, that is not the end. There are lots of books written by these professionals.

What is interesting about books is that you can learn something in hours, something that took many years for professionals to learn.

While these books may not turn you into a Django pro, it will set you a great path to becoming a good developer.

Write Code Every Day

You should code daily. How much, that depends on yourself.

Single mothers may have a harder time setting aside time compared to teenagers. People who spend most of the day outside working have little time to code. Those working as programmers also need time outside the job to code.

Whatever time you can create, code every day even if it's only for 20 minutes.

Creating a routine of coding every day will help you become a better programmer.

Take Some Breaks

After a long period of studying, you need to take breaks.

Taking breaks allows your brain, not only to rest but to organize what you've learned.

It's easy to lose your focus of the bigger picture, and lock yourself into a piece of code. It's easy to get disappointed because a certain concept isn't sinking into your brain when you've been studying for 2 hours non-stop.

So, you should take a break, come with your mind clear, and well-rested.

Ask Other People

When you're learning, you'll not know a lot of things. Or maybe you don't see how they work. You will have a lot of questions related to coding. And the best way to solve something is to ask.

There's a lot to learn from just asking people. Go to coding forums and post your question. Ask your workmates. And don't be afraid to ask. Remember, it's better to ask a dumb question than don't ask and put yourself in risk.

Ask, and you'll learn.

Teach Your Friends

Teaching is an excellent way to master something.

When you learn something, you understand how to use it. However, when you're teaching others, you need to know how to use it, and how it works operates internally. What it does in the backend, what happens if you do this, what happens if you skip this, why the developers made it work this way, and no other way.

There is so much to explore and going deep into the subject when you teach other people.

These are the tips you need to follow to become a good programmer in Python. Now it is up to you to follow them.

Conclusion

Now that you're done reading this book, we hope that it's been amazing, and you've learned plenty of Python concepts and its Django framework. If you prefer to learn more interactively, you can check out some online tutorials.

Remember that Django is a popular framework, but there are other frameworks for web development, and depending on the kind of project, you may consider using different frameworks.

However, if you feel you've attained a solid foundation of Python and Django fundamentals, you can take a new challenge by expanding your knowledge reading intermediate to advanced books.

