

RUSSEL R. RUSSO



COMPUTER PROGRAMMING: PYTHON

3 BOOKS IN 1

A CRASH COURSE TO GO DEEP INTO ARTIFICIAL INTELLIGENCE. TOOLS,
TIPS AND TRICKS TO IMPLEMENT YOUR NEURAL NETWORKS WITH
MACHINE LEARNING AND DATA SCIENCE

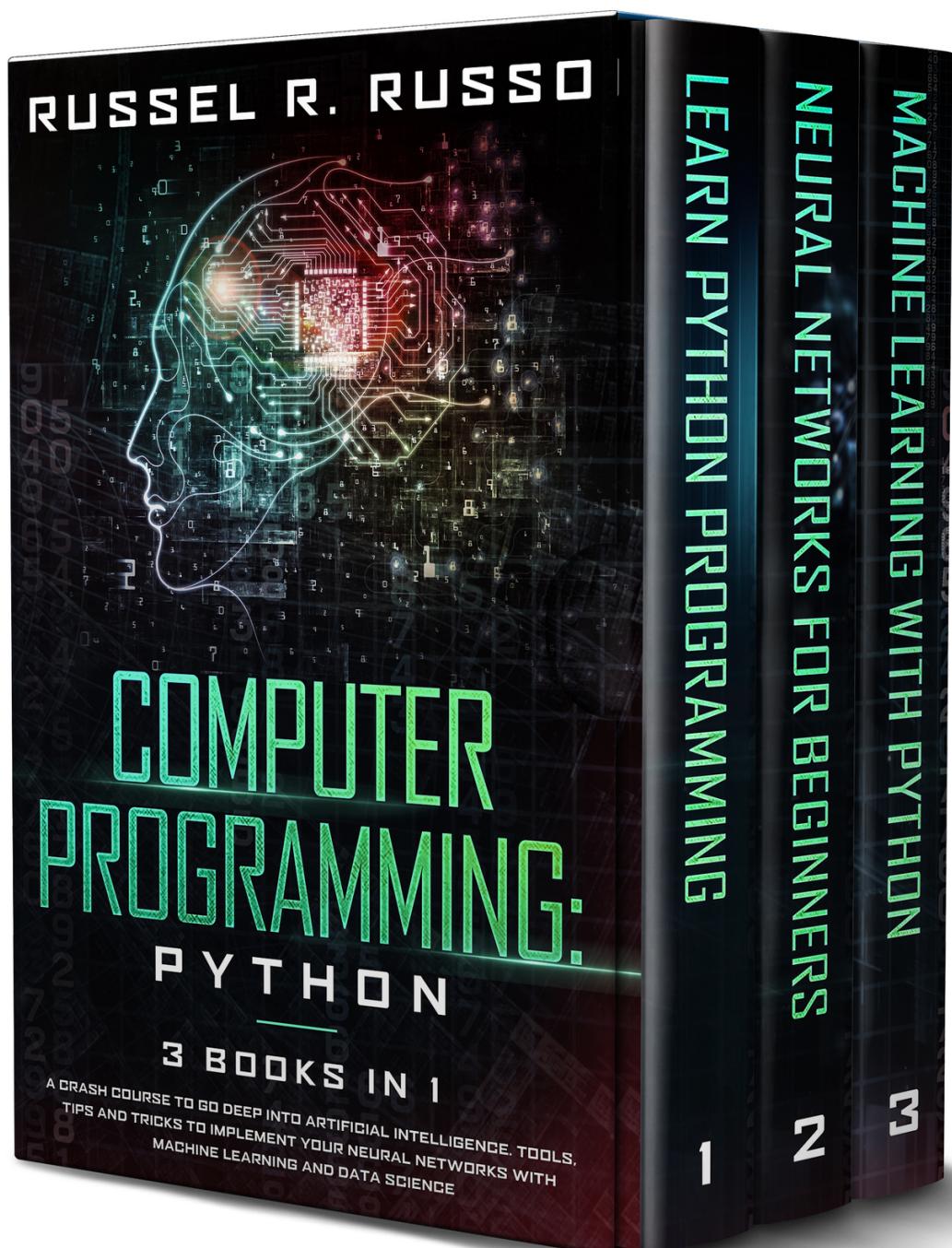
LEARN PYTHON PROGRAMMING

NEURAL NETWORKS FOR BEGINNERS

MACHINE LEARNING WITH PYTHON

1

3



Computer Programming: Python

3 books in 1

Learn Python Programming

+

Neural Networks for Beginners

+

Machine Learning With Python

A Crash Course to Go Deep into Artificial Intelligence. Tools, Tips and Tricks to Implement Your Neural Networks with Machine Learning and Data Science

RUSSEL R. RUSSO

To Enza and Angelomaria

© Copyright 2019 - All rights reserved.

The content contained within this book may not be reproduced, duplicated or transmitted without direct written permission from the author or the publisher.

Under no circumstances will any blame or legal responsibility be held against the publisher, or author, for any damages, reparation, or monetary loss due to the information contained within this book. Either directly or indirectly.

Legal Notice:

This book is copyright protected. This book is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part, or the content within this book, without the consent of the author or publisher.

Disclaimer Notice:

Please note the information contained within this document is for educational and entertainment purposes only. All effort has been executed to present accurate, up to date, and reliable, complete information. No warranties of any kind are declared or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice. The content within this book has been derived from various sources. Please consult a licensed professional before attempting any techniques outlined in this book.

By reading this document, the reader agrees that under no circumstances is the author responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to, — errors, omissions, or inaccuracies.

Table of Contents

General Introduction

Book 1: Learn Python Programming

[Introduction](#)

[Chapter 1:](#)
[Interacting with Python](#)

[Chapter 2:](#)
[Coding Your First Application](#)

[Chapter 3:](#)
[Basic Elements of Python](#)

[Chapter 4:](#)
[Number Data Types](#)

[Chapter 5:](#)
[Conditional Statements](#)

[Chapter 6:](#)
[Working With The Python Files](#)

[Chapter 7:](#)
[How To Work Classes And Objects](#)

[Chapter 8:](#)
[Strings](#)

[Chapter 9:](#)
[Making Predictions With Algorithms](#)

[Chapter 10:](#)
[Error Management](#)

[Chapter 11:](#)
[Functions](#)

[Chapter 12:](#)
[Walkthroughs](#)

[Chapter 13:](#)
[Building A Complete Program](#)

[Chapter 14:](#)
[Directories](#)

[Chapter 15:](#)
[Python - Dictionary](#)

[Chapter 16:](#)
[How To Use OOP](#)

[Conclusion](#)

Book 2: Neural Networks for Beginners

[Introduction](#)

[Chapter 1:](#)
[Types of Neural Networks](#)

[Chapter 2:](#)
[Components of a Neural Network](#)

[Chapter 3:](#)
[Advantages to Using a Neural Network](#)

[Chapter 4:](#)
[Downsides of Neural Network Computing](#)

[Chapter 5:](#)
[Profiting from Neural Networks](#)

[Chapter 6:](#)
[Neural Network Architecture Search](#)

[Chapter 7:](#)
[Neural Network Algorithms](#)

Chapter 8:
Machine Learning and Big Data

Chapter 9:
What is Predictive Analytics?

Chapter 10:
What is Data Mining?

Chapter 11:
Recurrent Neural Networks

Chapter 12:
Predicting Time Series

Chapter 13:
Neural Networks And Artificial Intelligence

Chapter 14:
Technical Fields and How They Operate

Chapter 15:
Neural Networks in the Future

Chapter 16:
The Application of ANN

Chapter 17:
Learning in Artificial Neural Networks

Chapter 18:
Recursive Neural Network in Theano

Conclusion

Book 3: Machine Learning WIth Python

Introduction

Chapter 1:
Machine Learning - Automation Within Learning

Chapter 2:

Building Blocks Of Machine Learning

Chapter 3:

Understanding The Syntax Of Python

Chapter 4:

Variables and Data Types Of Python

Chapter 5:

Manipulation of Data in Python

Chapter 6:

Understanding K-Means Clustering

Chapter 7:

Lists

Chapter 8:

Neural Networks With Scikit-Learn

Chapter 9:

Machine Learning And Big Data

Chapter 10:

Machine Learning and Support Vector Machines

Chapter 11:

Using The Neural Networks

Chapter 12:

Artificial Neuron Networks

Chapter 13:

Data Scrubbing

[Chapter 14:](#)

[Syntax](#)

[Chapter 15:](#)

[Auto Learning Myths](#)

[Conclusion](#)

General Introduction

Artificial intelligence is not just one of the elements behind the success of the Big Ones.

In fact, when you think at Google, Amazon, Facebook, and how they made it up to where they are now, with exponential growth from an inspired idea to a real empire, you know for sure that Artificial Intelligence is the one thing that made it possible.

A little too much isn't it?

I know, we are used to hearing this story told another way. The common storytelling about the building of these incredible success case histories is about the hero's journey of these visionary talents, struggling towards their goal, overcoming the everyday difficulties and the skepticism of everybody around them, stubbornly driven towards their aim, and finally succeeding against all odds.

And this, of course, is not false. We have, behind these success stories, extraordinary people, with extraordinary vision and extraordinary determination. And this is true for the aforementioned Big Boys, but also for some "minor" companies (just some billions of dollars) like Airbnb or Uber.

They are told as stories of great entrepreneurs achieving amazing results in different areas. And the greatness of these entrepreneurs is not questioned. Not in any way.

But...

What do all these extraordinary economical and social enterprises have in common? Is it just and only the undeniable determination

and ability of their leaders to get these companies such incredible results in such a short period of time, starting from zero?

Well, actually, there is another thing they all have in common.

Think about it, none of these companies could be where it is without an outstanding ability of elaborating enormous amounts of data, and of doing that in a very smart way. Not just the brute force of crossing billions of data, but the intelligence of doing that by processes that are incredibly similar to the human brain's.

Think about the success case histories in the pre-AI era. Of course you can find many of them, and they are generally led by entrepreneurs that have a lot in common with the ones we just talked about as for mindset, vision and determination. But they were the story of a lifetime, sometimes of generations, building greatness year by year. The incredibly fast raise that we see in these years was very rare at the time. And AI is one of the main (maybe The main) reasons for this difference.

Now, of course this doesn't mean that by learning about Python, Neural Networks, Deep Learning and Machine Learning you will have your billionaire company in a few years. Of course I wish that for you, if that's what you want for your life, but well, it may not happen. Also, truth be told, in most cases these entrepreneurs don't even know that much about computer programming.

The point is, though, that Artificial Intelligence has changed our lives, it's all around us, and we are so used to it that we don't even notice anymore. But this is the present and the future of the workplace, and it will be every day more important to be able to understand and work with this kind of technology.

Python Programming is The Skill you want to have these days if you are into computer programming and Artificial Intelligence, and the three books present in this collection will guide you from a

beginner level to an intermediate one moving your first steps on this never ending path.

You will go through the secrets of Python Programming, Neural Networks and Machine Learning opening your mind to the infinite possibilities they offer. You will have hands-on exercises, experimenting yourself with Python Programming.

The aim of these books is not to give you a complete formation (assuming that a complete formation does even exist when it comes to computer programming), but to get you started and make you want for more. The best thing I can wish you is that at the end of these books you will want to discover more, learn more and test more in this wonderful journey that is Machine Learning.

I could not be happier if this was for you the beginning of a lifelong path.

To your discovery!

Book 1:

Learn Python Programming

*A Beginners Crash Course on Python Language for
Getting Started with Machine Learning, Data
Science and Data Analytics*

To Enza and Angelomaria

Introduction

There are a lot of reasons why you will love working with the Python code. It is easy to use, easy to learn, has a lot of great frameworks and libraries to work with (and we will discuss at least a few of these as we go through this guidebook), and is still powerful enough to make machine learning easy for you.

While it is possible to work with other coding languages to help you get the results that you want, most people prefer to work with Python due to all of the benefits that we have discussed. Before we take a look at how to set up the Python environment so you are able to use it properly, let's take a look at a few of the different parts that come with the Python language, so you understand how a few of these codes can work for you.

The parts you should know about the Python code

First, we need to take a look at these important keywords in the Python language. Like with what you will find in other coding languages there is a list of keywords in Python that are meant to tell your text editor what to do. These keywords are reserved and you should only use them for their intended purposes if you want to be able to avoid issues with your code writing. They are basically the commands that will tell your compiler how to behave and they remain reserved so that you can execute the code without a lot of issues in the process.

Variables are important because they will save up spots on your computer's memory in order to hold onto parts of your code. When

you go through the process of creating a new variable, you are making sure that you are reserving some space on your computer for this. In some cases, such as when you are working with data types, your interpreter will do the work of deciding where this information should be stored because that speeds up the process.

When it comes to working with variables, your job will be to make sure that the right variables are lining up with the right values. This will ensure that the right parts show up in your code at the right time. The good news is that you are able to give the variable the value that you would like, but do check that it actually works inside of your code. When you are ready to assign a new value to a variable, you can use the equal sign to make this happen. Let's look at a good example of how this would work:

```
#!/usr/bin/python
counter = 100      # An integer assignment
miles   = 1000.0    # A floating point
name    = "John"     # A string
print counter
print miles
print name
```

With the above example, you will get the results to show up with the variable that you placed with the value. So, when the counter shows up, it will show you a 100 for the counter, 1000 for the miles, and John as the result of the name.

Next are the Python comments. These are helpful to leave a little note in your code and can make a difference in how others are able to look through the code and know which parts are supposed to work with. Working with the comments can be relatively easy when you are on Python. You simply need to add the # sign in front of any comments you would like to write. The compiler will know how to avoid that part of the code and will skip over it, without any interruption in the program.

One thing to note is how many comments you write. While you can technically write out as many of these as you would like or that you think the code needs, try to only keep with the ones that are absolutely necessary. You do not want to write in so many comments that it is hard to read the rest of the code. Just write in comments when they are needed, not all of the time.

Python statements are a simple part of the code that can make a big difference, so we are going to take some time to explore them real quick here. Statements are going to be the part that the compiler is going to execute for you. You can write out any kind of statement that you want, but make sure they are in the right place, and that you are not using any of the keywords with them either, or the compiler will get confused.

And the next thing that you need to take a look at here is the functions. Functions can be another part of your language that you need to learn about. This is basically a part of the code that can be reused and it can help to finish off one of your actions in the code. Basically, these functions are often really effective at writing out your code without having a lot of wasted space in the code. There are a lot of functions that you can use in Python, and this can be a great benefit to the programmer.

When you are working on your code, you will need to make sure that you are defining the function. Once you have been able to define the function and it is considered finalized, it is time to execute it to work in the code. You have the choice to call it up from the Python prompt or you can call it up from another function. Below is an example of how you are able to do this:

```
#!/usr/bin/python  
# Function definition is here  
def print( str ):  
    "This prints a passed string into this function"  
    print str  
    return;
```

```
# Now you can call printme function  
printme("I'm first call to user defined function!")  
printme("Again second call to the same function")
```

When this is placed into your compiler, you will be able to see the two statements that we wrote out inside of the code come up like a message. This is just one example of how you would call up a function and you can always change up the statements that are inside the code and figure out how you want them to execute later.

These are just a few of the basics that come with the Python code. We will take a closer look at doing these a bit more as we move through this guidebook, but these will help you get the basics of the Python language and how you can use it for your advantage in machine learning.

Getting that environment set up

Now that we have had a chance to look at machine learning, some of the ways that you can benefit from and benefit from machine learning, and some of the different types of machine learning that you are able to work with, it is time to introduce some Python into this. Python is a great coding language that you are able to work with, no matter what your skill level is when coding. And when it is combined with some of the ideas that come with machine learning, you are going to be able to see even better results in the long run as well.

That is why we are going to spend some time looking at how you can set up your own environment when working with the Python code. This will help you to make sure that Python is set up on your computer in the proper manner, and will make it easier to work with some of the codes that we will talk about later on.

You will find along the way that the Python code is going to be a really easy one to learn compared to some of the others that are out there, and it is often one that is recommended for beginners to learn because it is simple. But this isn't meant to fool you! Just because you see that it is simple to work with doesn't mean that you won't

be able to find the strength and the power that you need with this one. There are a lot of different parts that you can learn about the code, but first, we are going to make sure that the environment for Python is set up in the right way to help with the Python environment with the help of machine learning.

So, to help us get this done, we need to go to www.python.org and download the Python program that we want to work with. Then make sure that with the files you are working with, you will need to make sure the right IDE is present. This is going to be the environment that has to be there and will ensure that you are able to write out the codes that you want to work with. The IDE is also going to include all of the installation of Python, the debugging tools you need, and the editors.

For this specific section of machine learning, we are going to focus on the IDE for Anaconda. This is an easy IDE to install and it is going to have some of the development tools that you need. It is also going to come with its own command line utility, that is going to be so great for you installing any of the third party software that you need with it. And when you work with this IDE, you won't have to worry about doing a separate installation with the Python environment on its own.

Now we are on to the part of downloading this IDE. There are going to be some steps that you will need to complete to make this happen. We are going to keep these steps as simple as possible, and we are going to look at what we need to do to install this Anaconda IDE for a Windows computer. But you will find that the steps that come with installing this on a Mac computer or a Linux computer are going to be similar to this as well. Some of the steps that you need to use in order to help you download this kind of IDE to your computer include:

1.

To start, go to

<https://www.anaconda.com/distribution/>

2.

From here, you will be sent to the homepage. You will need to select on Python 3.6 because it is the newest version. Click on Download to get the executable file. This takes a few minutes to download but will depend on how fast your internet is.

3.

Once the executable file is downloaded, you can go over to its download folder and run the executable. When you run this file, you should see the installation wizard come up. Click on the “next” button.

4.

Then the License Agreement dialogue box is going to appear. Take a minute to read this before clicking the “I Agree” button.

5.

From your “Select Installation Type” box, check the “Just Me’ radio button and then “next”.

6.

You will want to choose which installation directory you want to use before moving on. You should make sure that you have about 3 GB of free space on the installation directory.

7.

Now you will be at the “Advanced Installation Options” dialogue box”. You will want to select the “Register Anaconda as my default Python 3.6” and then click on Install.

8.

And then your program will go through a few more steps and the IDE will be installed on your program.

As you can see with all of this, setting up the Python environment that you would like to work with is going to be simple. You just need to go through these steps to get the Anaconda IDE set up properly, and then you are able to use it for all of the codes that we will discuss in this guidebook, along with some of the other codes that you will want to write along the way.

Remember that there are some other options that you can work with when it is time to pick out an IDE that you would like to work with. If you are doing some other work than machine learning, or you like some of the features and more that comes with another IDE, you are able to download these IDEs to make it work with them as well. But we are going to spend time working with the Anaconda IDE because it is going to have all of the features that we need to get the machine learning algorithms working the way that you would like.

Chapter 1:

Interacting with Python



Python is built to provide users with several ways to improve interaction with the computer. Take the Integrated Development Environment (IDLE) for example. IDLE makes it easier for Python users to develop better applications. There are times, however, when you simply want to double-check an existing program for bugs or errors. If this is the case, it is recommended that you use the command-line version of Python. This version offers better user interface than other platforms because it requires less resources, fewer command-line switches and depends on minimal system requirements to run the code.

Starting Python

Depending on the platform you have installed in your work computer, there are multiple ways to open Python's command line. Here is a list of methods that you can use:

- Open a terminal or a command prompt. Simply key in Python, then press Enter. Programmers often use this option when they require better flexibility in configuring their Python environment.
- Locate the Python folder in your file manager and then select Python (command-line) option. This option makes you use a command-line version of Python configured in its default settings.
- If you are using Windows, you still have to install Python. Once installed, go to the Python folder. This is usually found in this address: C:\Python33. Double-click thePython.exefilename. This method opens a command-line configured in its default settings. Using this method, however, gives you the option to open the command-line using credentials that have upgraded privileges. This allows applications that utilize secured resources to be run by the system.

When running Python using a command prompt, you can use various options that increase its usability. Access these options by using the following syntax:

Python <options>

Replace<options> with any of the following case-sensitive options:

- b This command includes red flags or warning signals to the output when the program being run includes restricted Python features
- c This option uses the data and information taken from the cmd when starting the application. This also tells Python when to stop processing other data options.
- c
- m
- d

- d Locates errors from the program by starting the debugger option
- h Begins the program by displaying basic environment variables and helpful options in the screen.
- i This opens a Python interface that allows you to inspect the code being run.
- m This enables Python to run the library system specifically designated by the command mod as part of the script.
o
d
- q This stops Python from printing sensitive information like version and copyright data in an interactive startup.

Typing a Command

After opening the command-line version of Python, the precautionary first step is to check whether the options you need to run your application checks out. Commands are coded instructions that the computer uses in carrying out specific tasks or evaluate ideas and information.

Unlike the IDLE which hides the less fancy details of Python programming, command-line interface gives you first-hand experience and understanding on how Python works.

Commands tell the computer what to do. They are a series of steps in a coded procedure. This is done by typing commands that Python is familiar with. In response, Python translates these commands into instructions that can be understood by the computer.

For instance, the command `print()` displays the result of the program on the screen.

To give you a quick example of how commands work, execute the following steps using your command-line interface:

- At the command line, type the following:

```
print("This is an example of how command works.")
```

- Press Enter to signify the end of the command. The command-line will then display:

This is an example of how command works.

Exiting Python

If you want to close the command-line after several hours of playing with the programming language, there are two standard means for you to do this. There are other methods to close Python but it is recommended that you use the standard method to make sure that Python will behave normally the next time you use it. These methods are as follows:

`quit()` and `exit()`

These commands will shut down the current Python session. The good thing about said commands is that they can accept an optional instruction. For instance, you can use `quit(5)` or `exit(5)` to shut down Python when the ERROR LEVEL variable reaches a numeric value of 5.

To illustrate how to Exit Python properly, perform the following procedure:

1. Open a terminal or command prompt.
2. Key in Python then press enter.
3. Type `quit()`.

Chapter 1 Summary

- Python is built to provide users with several ways to improve interaction with the computer.
- Commands are coded instructions that the computer uses in carrying out specific tasks or evaluate ideas and information.
- Commands tell the computer what to do.
- Here is a list of methods for starting python:
 - Open a terminal or a command prompt.
 - Locate the Python folder in your file manager and then select Python (command-line) option.
 - If you are using Windows, you still have to install Python.

Chapter 2: **Coding Your First Application**



Beginners often see application development as either some form of rocket science or magic that only computer nerds can understand. What they don't understand is that they, too, can harness the power of computer programming.

Building applications follows specific stages of development. It's more than just a series of interrelated steps but it's not too complicated for beginners like you to understand. After you finish this chapter, you will be able to develop simple applications that don't require a magic wand.

As with other tasks, people use tools to build complicated applications. Fortunately, Python does not require tools to code applications. Using IDLE, however, makes it easier for Python programmers to simplify the coding process and process data and information faster.

Using IDLE

While you can use any text editor to code a Python application, IDLE provides a more efficient platform to write a Python code. The Interactive Development Environment is the most widely-used platform used by experienced programmers. IDLE can perform the following Python-related tasks:

- Write Python code,
- Save and run Python data and files,
- Perform simple editing like copy, cut and paste,
- Browse through and find specific Python files,
- Remove errors by performing debugging procedures, and
- Emphasize and recognize important codes and keywords from special text information.

Most of the above functions can be accomplished using a command-line interface but it takes time and effort. For instance, you can perform debugging procedures using command line interface but this is done in a manual and error-prone process. This makes using IDLE several times easier.

Creating the Application

Now that you are acquainted with the Python system, it is time to create your first Python application. When creating one, you can start by opening a new window to enter your commands. After typing the commands, you can save the file in a storage system that will serve as a database for running your application.

1.
Opening a new window

The Python Shell window is an interactive window that provides immediate feedback on every command entered in the system. Python is a static environment that allows you to key in the commands and save them for future use.

In order to open a new window, click on:

Choose File  New File

Using the Edit window is very similar to text editors. It also has access to basic commands such as Cut, Copy and Paste. Instead of executing the command, pressing Enter brings you to a different command line.

2.

Typing the command

Using the operative commands in your arsenal, simply key in the commands to the space provided by the platform.

3.

Saving the file

Before you are able to run the file, you are required to save the file. This is the system's way of freezing your code should you make an error in the process. This ensures the safety and the integrity of the code. In determining what went wrong, spotting errors, making necessary corrections, the usual first step is to save the file. This makes it easier for the system to run the application and spot possible error points.

When saving a file, follow this process:

Choose File  Save

This opens the Save As dialog box. By default, the Edit window chooses the destination folder connected to the installation folder of the application. Since this folder houses the Python codes, experienced programmers create folders where they will save application codes.

In the File Name portion, fill it in with the desired name for your application. For instance, if you want to name it as:MyFirstProject, Python will save it as MyFirstProject.py.

Running the Application

In order to determine the success of the coded application, you must be able to run them. Python will attempt to translate the codes and execute several computer functions from the translation.

One of the upsides of Python is that it provides many methods to run an application. In order to run your application, click on:

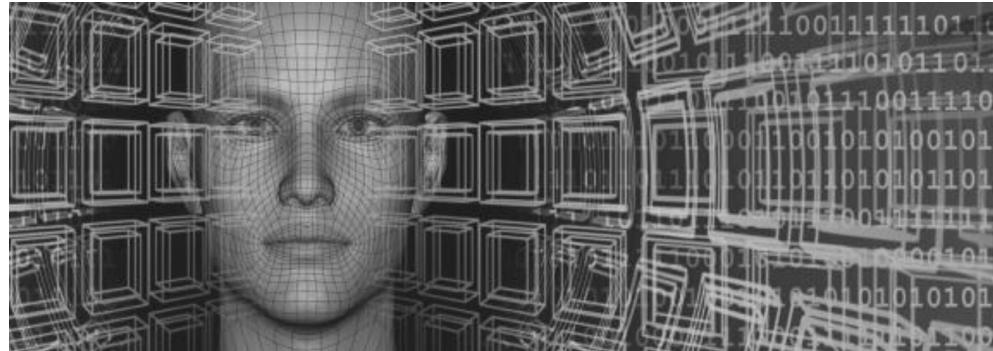
Run⇒ Run Module

A new Python Shell window opens to display the output.

Chapter 2 Summary

- Building applications follows specific stages of development.
- IDLE provides a more efficient platform to write a Python code.
- Python is a static environment that allows you to key in the commands and save them for future use.
- Python will attempt to translate the codes and execute several computer functions from the translation.

Chapter 3: **Basic Elements of Python**



Learning the ABCs of anything in this world, is a must. Knowing the essentials is winning half the battle before you get started. It's easier to proceed when you are equipped with the fundamentals of what you are working on.

In the same manner that before you embark on the other aspects of python let us level off the basic elements first. You need to learn and understand the basics of python as a foundation in advancing to the more complicated components. This fundamental information will greatly help you as you go on and make the learning experience easier and enjoyable.

Familiarize yourself with the Python Official Website <https://www.python.org> . Knowing well the website of python would give you the leverage in acquiring more information and scaling up your knowledge about python. Also, you can get the needed links for your work.

Learn from Python collections. Locate python collections such as records, books, papers, files, documentation and archives and learn from it. You can pick up a number of lessons from these, and expand your knowledge about Python. There are also tutorials, communities and forums at your disposal.

Possess the SEO Basics. Acquire some education on Search Engine Optimization so you can interact with experts in the field and improve your python level of knowledge. That being said, here are the basic elements of Python.

Language and programs

This is the phase where the program language is presented to make the user understand the type of language employed and knowing how to use it.

Interpretations and modules drafting

Python can be used as an active translator or transcriber by interaction through the web. It can also be employed to formulate lessons. In interaction, though, there is one serious concern: that is, it is impossible to keep a copy of what transpired. On the other hand, using lessons allows you to keep a record of the work done. In the interactive translator, you are allowed to open only one display page, while in lessons, you can open as many as you need.

Variables

Python uses information that are not constant, these are used to keep the data. When using these, be sure to put descriptions. These data could be names, age, addresses, gender and other similar material.

Outputs and Inputs

Any computer program requires interfacing between itself and the person using it. The user encodes and that is input, and the output is printing what has been encoded.

Mathematics

Numbers are the common language in computer programs including Python. Mathematical operations are used by Python as you will learn later on. Most of its language are represented by mathematical equations and symbols.

Loop

You need to understand the term loop in python. It is a symbol used to represent repeated word/s or sentence/s in python programming. Anything that is being repeatedly used can employ a loop.

Python categories

It is important to be acquainted with the types of python product categories for easy reference and understanding. Python categories are symbolized by A, B, C that signifies the shifts in language. Examples are 3.3.1 to 3.3.2. This means there are minor changes, but when it employs something like 2.xx to 3.xx it means there are major changes.

Cutting

This is a critical component of python which is used to copy the desired part of a data. It is a method of making programs simple by concentrating on items from a gamut of data. When you do that, you are actually removing components that are not relevant to the program.

Modules

Modules are files of descriptions and declarations of Python. It is a list of all the terminologies used by python with corresponding explanations of each. Python adopts a method of consolidating definitions into one folder called *module*. These modules can be introduced into different modules depending on the needs of the programmer or user.

This is created to allow users to have a better understanding and easy access to the standard library of Python. A programmer or even a beginner can make modules for his use.

Modules can be on: Indexing and searching, Audio and Music, Web development, Console and Database. Python provides an array of modules that you can use. You can also make your own.

Source codes

Generating Python source codes can be tedious, if you don't know how to derive your codes.

Program developers have now an application that converts your Python 2 codes to Python version 3 codes from AST.

One way of doing this is to use context managers.

These are the most basic elements in python, there are more but with the ones presented, one can already start using python and learn the others, as you go on in your programming.

Types of Python Statements

There are several types of Python statements that you must know. These statements will help you in creating your Python syntax/codes.

Here are the most common statements:

1.

Simple statements (simple_stmt)

These are statements that are composed of a single logical line. They may occur singly, or in several statements in one line. If there are several simple statements, they can be separated by a semicolon. The most common simple statement is the ‘print’ statement. You have also the ‘delete’ (del), ‘import’, ‘return’, ‘pass’, ‘continue’, ‘assignment’, ‘raise’, ‘break’, to name some.

Example:

```
print var1
```

This means that the values in variable 1 (var1) will be printed. Of course, var1 has to be defined first by assigning the values prior to executing the ‘print’ command.

2.

Assignment statements (assignment_stmt), (target), (target_list)

These statements are used when names are assigned to values, and when you want to modify mutable (can be changed) objects. The syntax is similar to that of the expression statements.

3.

Expression statements (expression_stmt)

These statements are generally used for computations and for evaluating an expression list. They are also useful in writing values. They usually return the (none) value, when used to call a procedure.

4.

Import statements (import_stmt)

These are used to import files, functions or modules. Python has packages (directories) containing modules (files). You can quickly import modules by using the key ‘import’.

Example:

I want to import my names1 and strings1 files; these are the statements:

```
import names1
```

```
import strings1
```

See image below:

```
>>>
>>> import names1
('Potter Richard', 'Walker Henry', 'Fell Don', 'Dean James', 20, 34, 41, 32)

>>>
>>> import strings1
Remember.911
>>>
```

If Python cannot find the file, it will return the ‘ImportError’. You cannot import your files if you have not saved them. Likewise, you cannot import a file or module that is not part of the Python program.

You can use `importlib.import_module()`, when you want to know more about the modules.

5.

Continue Statements (`continue_stmt`)

These statements indicate that the statement, usually a loop, continues with the next loop. It’s used with the ‘for’ or ‘while’ loop.

Example:

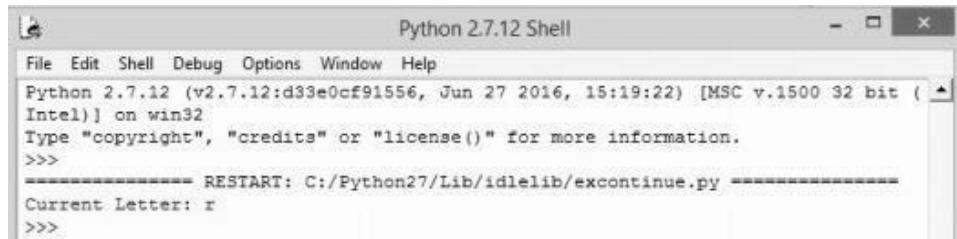
```
for letter in 'Walker':
    if letter == 'W':
        continue
    print 'Current Letter:', letter
```



The screenshot shows a Python IDLE window titled "excontinue.py - C:/Python27/Lib/idlelib/excontinue.py (2.7.12)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code area contains the following Python code:

```
for letter in 'Walker':
    if letter == 'W':
        continue
    print 'Current Letter:', letter
```

When ‘Run’ and ‘Run Module’ are clicked, the result that will appear in a new shell will be the rest of the letters of “Walker”.



A screenshot of the Python 2.7.12 Shell window. The title bar says "Python 2.7.12 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/Lib/idlelib/excontinue.py =====
Current Letter: r
>>>

6.

Break statements (break_stmt)

These are statements that may occur in the ‘for’ or ‘while’ loops. Their function is to ‘break’ the nearest enclosing loop, and resumes execution on the next statement. But the loop will finally ‘break’ when the ‘try’ statement and the ‘finally’ clause are executed.

Example:

For letter in 'Walker':

```
if letter == 'l':  
    break  
print 'Current Letter: ', letter
```

This ‘break’ statement will return these results:

Current Letter: W

Current Letter: a

This code will only return and print ‘W’ and ‘a’, which came before the break statement ‘l’.

7.

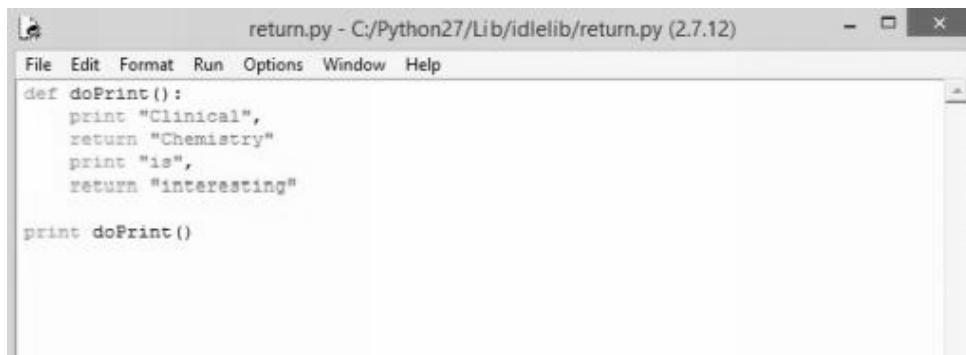
Return Statements (return_stmt)

These statements are usually used in evaluating an expression list and exiting a function, operation or method. There are two forms the ‘return’ and ‘return expressions’. They could be present in the definition of a function, but not in the definition of a nested class.

Example:

```
def doPrint():  
    print "Clinical",  
    return "Chemistry"
```

```
print "is",
      return "interesting"
print doPrint()
See image below:
```



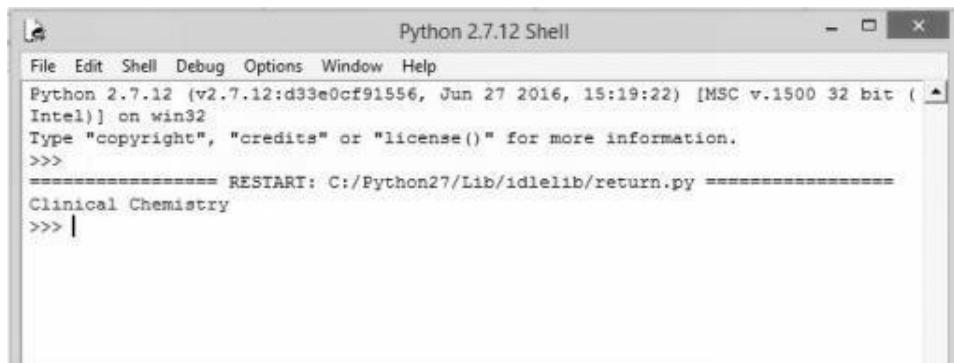
A screenshot of the Python IDLE editor window titled "return.py - C:/Python27/Lib/idlelib/return.py (2.7.12)". The code in the editor is:

```
def doPrint():
    print "Clinical",
    return "Chemistry"
    print "is",
    return "interesting"

print doPrint()
```

When ‘Run’ is clicked, the results printed are only ‘Clinical Chemistry’, because after the ‘return’ statement, the function will stop, so it won’t print/display the rest of the entries after the ‘return’ statement.

See image below:



A screenshot of the Python 2.7.12 Shell window. The command prompt shows:

```
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/Lib/idlelib/return.py =====
Clinical Chemistry
>>> |
```

8.

Else and Elif statements

Else Statement helps you to catch all of the remaining answers that the user could give you. If you don’t have a statement in the code to

handle the answer that the user gives, then the else statement will make sure to get you covered.

Example:

```
# Whether or not the stove is on.  
stoveOn = False  
  
# Is the stove on?  
if stoveOn == True:  
    print("The stove is on! Close it.")  
else:  
    print("The stove is off!")
```

Elif statements allow the user to look at a few options that you can present them with, and then, based on the kind of answer that the user gives, the program is going to execute the results that you added into the code for it.

Example:

```
# Whether or not the stove is on.  
stoveOn = True  
lightsOn = False  
  
# Is the light on? Just check the stove.  
if(lightsOn == True and stoveOn == True):  
    print("The lights are on, just close the stove.")  
  
elif (lightsOn == False and stoveOn == True):  
    print("Turn the lights on, close the stove!")  
  
else:
```

```
print("Both the lights and the stove is on.")
```

9.

If Statements

These are statements that give an ‘if’ argument. It’s often used with ‘else’ or ‘elif’ statements.

Example:

```
x= int(input("Please type a number: "))      # This is your base statement.
```

```
Please type a number:                      # This will appear, and you have to type  
a number .
```

the ‘if’ condition is:

```
if x > 0:
```

```
    x=0
```

```
    print 'You have a good number!'
```

```
You have a good number!          #If the number you typed  
is higher than zero(0), This will appear in the results.
```

The entire Python statement will be like this:

```
x= int(input("Please type a number: "))
```

```
if x > 0:
```

```
    x=0
```

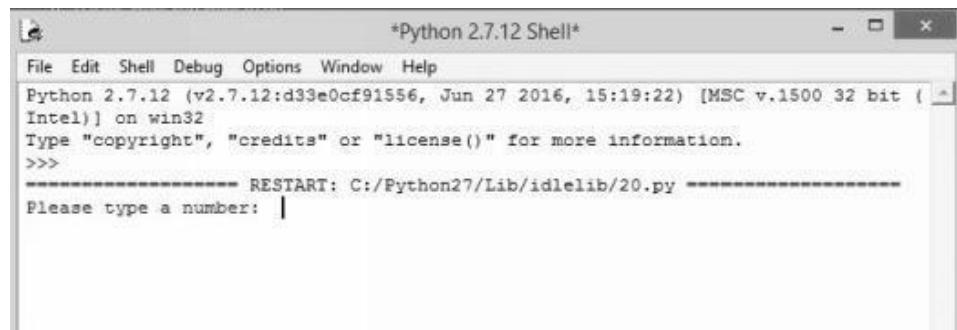
```
    print 'You have a good number!'
```

See image below:



```
20.py - C:/Python27/Lib/idlelib/20.py (2.7.12)
File Edit Format Run Options Window Help
x= int(input("Please type a number: "))
if x > 0:
    x=0
print "You have a good number!"
```

When you click ‘Run’, and then ‘Run Module’, the result will be a statement asking you to type a number:

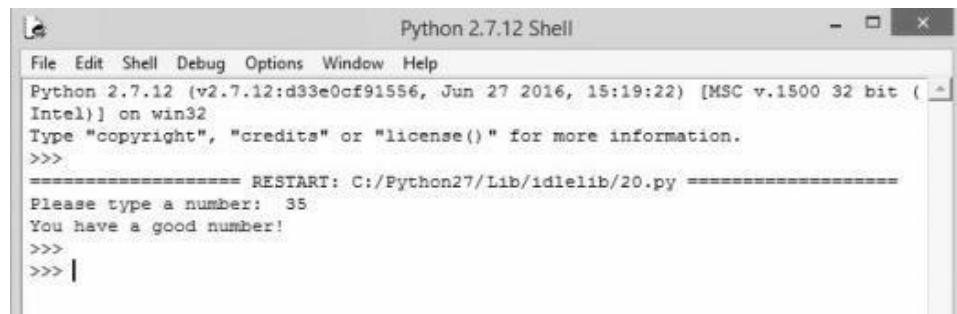


```
*Python 2.7.12 Shell*
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/Lib/idlelib/20.py =====
Please type a number: |
```

When you type 35, and press ‘enter’, this will be the result:
“You have a good number.”

This is because this was the ‘if’ statement specified for values more than zero (0).

See image below:



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/Lib/idlelib/20.py =====
Please type a number: 35
You have a good number!
>>>
>>> |
```

There are still various statements used in Python, but for now, these are some of the essential types that are good to know.

How to Start Using Python

Beginners may find it difficult to start using Python. It's a given and nothing's wrong about that. However, your desire to learn will make it easier for you to gradually become familiar with the language.

Here are the specific steps you can follow to start using Python.

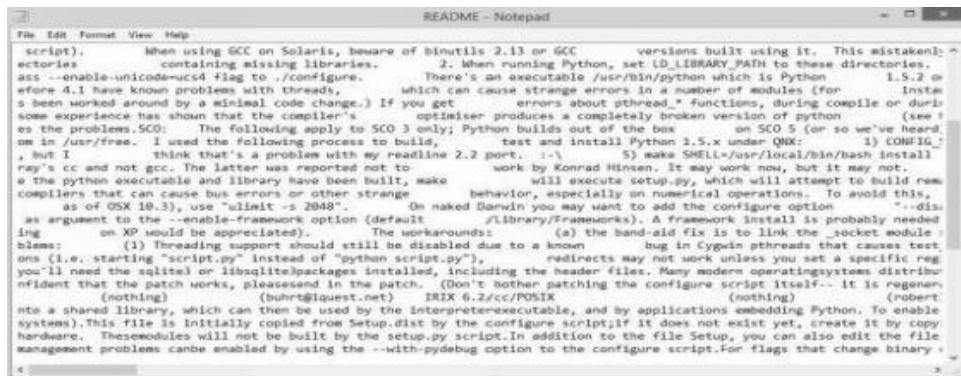
Steps in using Python

Step #1—Read all about Python.

Python has included a README information in your downloaded version. It's advisable to read it first, so you will learn more about the program.

You can start using your Python through the command box (black box), or you can go to your saved file and read the README file first by clicking it.

This box will appear.



You can read the content completely, if you want to understand more about what the program is all about, the file-setup, and similar information.

This is a long data that informs you of how to navigate and use Python. Also, Python welcomes new contributions for its further development.

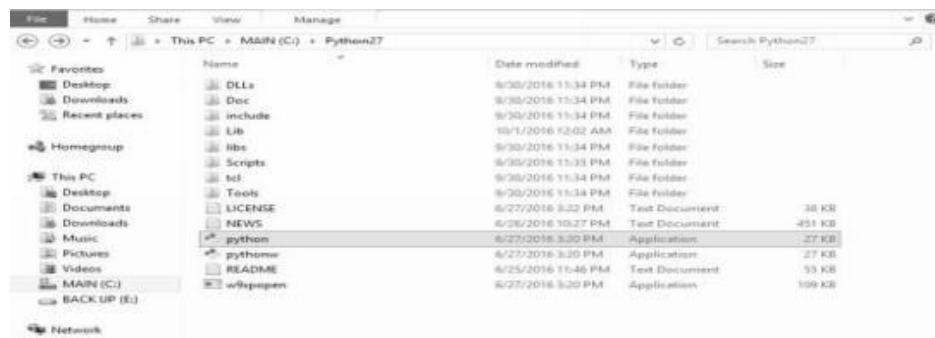
You can copy paste the content of the box into a Window document for better presentation.

If you don't want to know all the other information about Python and you're raring to go, you can follow these next steps.

Step #2–Start using Python.

First open the Python file you have saved in your computer. Click on Python as shown below. In some versions, you just click ‘python’ for the shell to appear.

See image below:



You can start using Python by utilizing the simplest function, which is ‘print’. It’s the simplest statement or directive of python. It prints a line or string that you specify.

For Python 2, print command may or may not be enclosed in parentheses or brackets, while in Python 3 you have to enclose print with brackets.

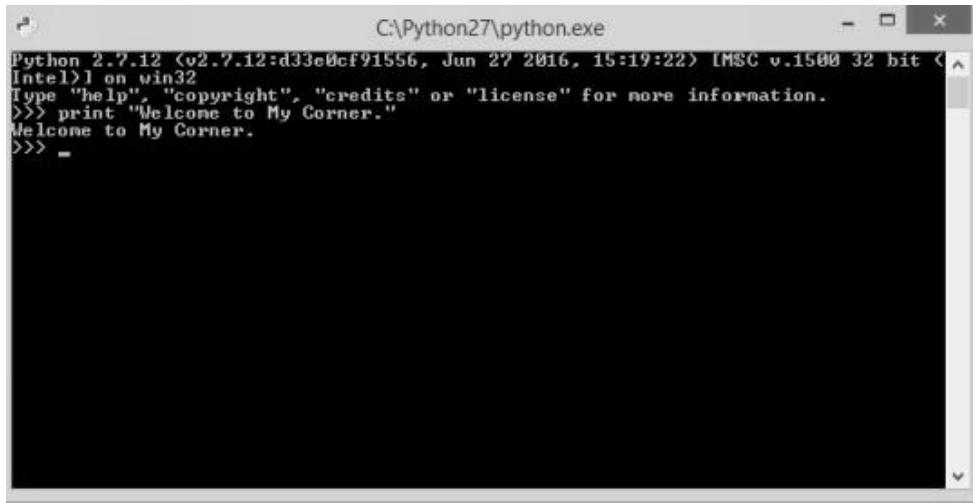
Example for Python 2:

print “Welcome to My Corner.”

Example for Python 3:

`print (“Welcome to My Corner”)`

The image below shows what appears when you press ‘enter’.



A screenshot of a Windows command-line window titled "C:\Python27\python.exe". The window displays Python version information and a simple "print" statement. The text in the window is as follows:

```
Python 2.7.12 <v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22> [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Welcome to My Corner."
Welcome to My Corner.
>>> -
```

You may opt to use a Python shell through idle. If you do, this is how it would appear:

In the Python 3.5.2 version, the text colors are: function (purple), string (green) and the result (blue). (The string is composed of the words inside the bracket ("Welcome to My Corner"), while the function is the command word outside the bracket (print)).

Take note that the image above is from the Python 2.7.12 version.

You have to use indentation for your Python statements/codes. The standard Python code uses four spaces. The indentations are used in place of braces or blocks.

In some programming languages, you usually use semicolons at the end of the commands—in python, you don't need to add semicolons at the end of the whole statement.

In Python, semicolons are used in separating variables inside the brackets.

For version 3, click on your downloaded Python program and save the file in your computer. Then Click on IDLE (Integrated Development Environment), your shell will appear. You can now start using your Python. It's preferable to use idle, so that your codes can be interpreted directly by idle.

Alternative method to open a shell (for some versions).

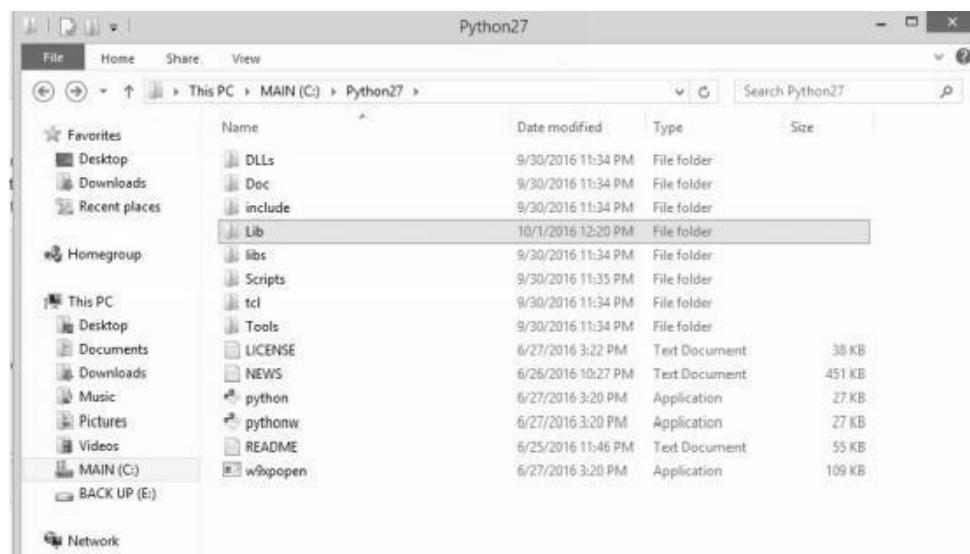
An alternative method to use your Python is to open a shell through the following steps:

Step #1– Open your menu.

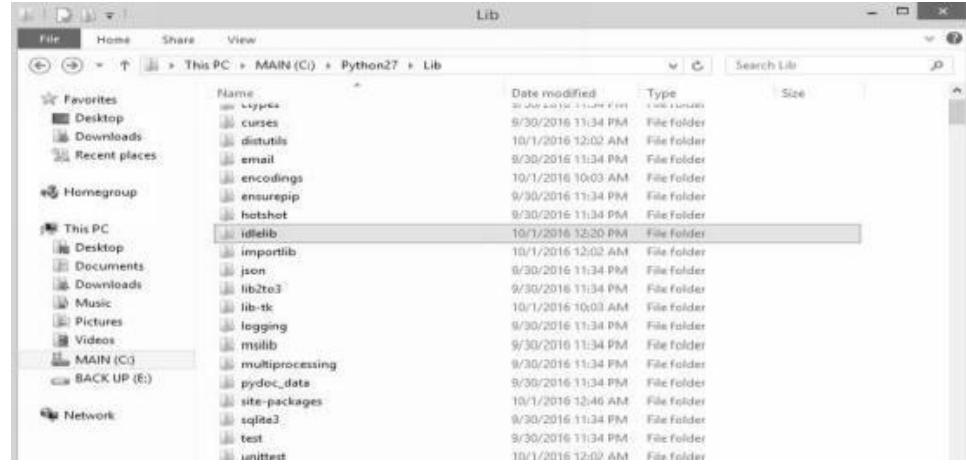
After downloading and saving your Python program in your computer, open your menu and find your saved Python file. You may find it in the downloaded files of your computer or in the files where you saved it.

Step #2–Access your Python file.

Open your saved Python file (Python 27) by double clicking it. The contents of Python 27 will appear. Instead of clicking on Python directly (as shown above), click on Lib instead. See image below.

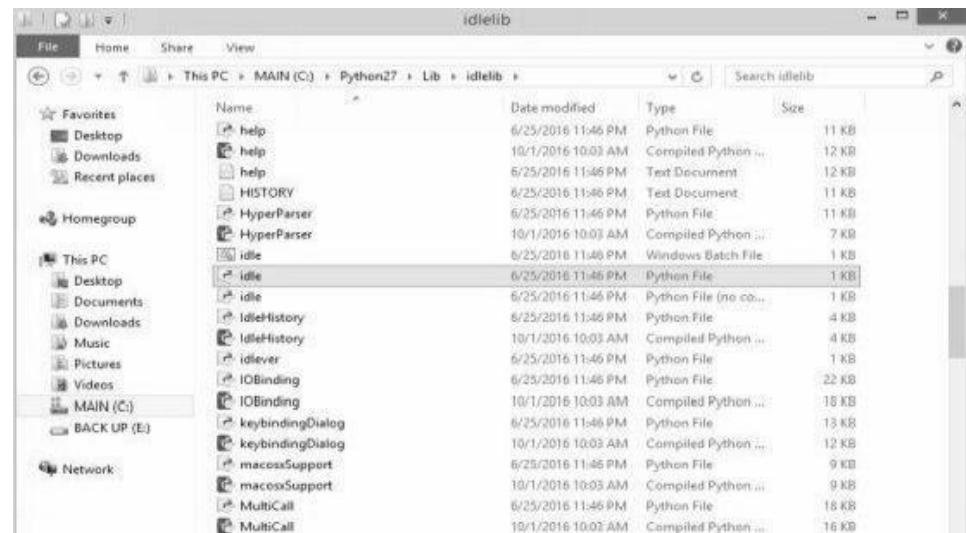


This will appear:



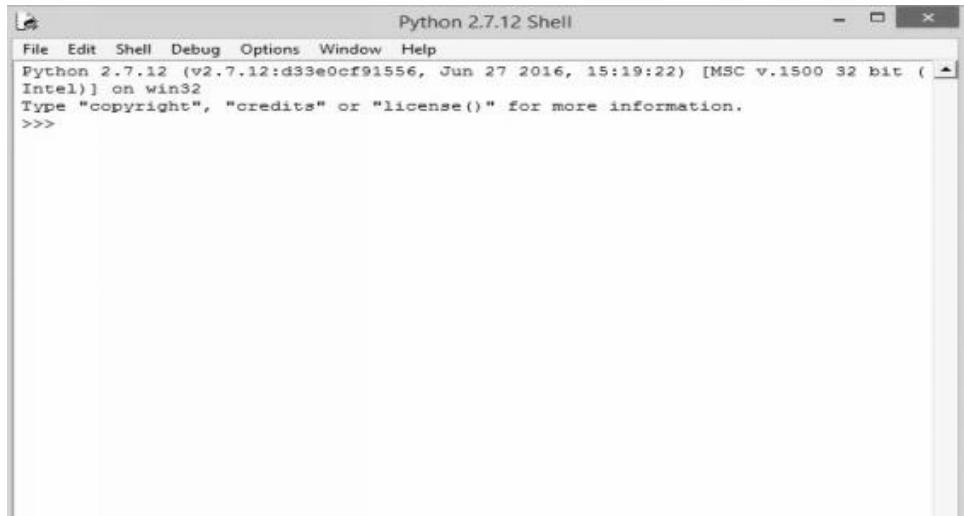
Step #3–Click on ‘idlelib’.

Clicking the ‘idlelib’ will show this content:



Step #4–Click on idle to show the Python shell.

When you click on any of the ‘idle’ displayed on the menu, the‘white’shell will be displayed, as shown below:



The difference between the three ‘idle’ menus is that the first two ‘idle’ commands have the black box (shell) too, while the last ‘idle’ has only the‘white’box (shell). I prefer the third ‘idle’ because it’s easy to use.

Step #5–Start using your Python shell.

You can now start typing Python functions, using the shell above.

You may have noticed that there are various entries to the contents of each of the files that you have opened. You can click and open all of them, as you progress in learning more about your Python programming.

Python is a programming language that has been studied by students for several days or months. Thus, what’s presented in this book are the basics for beginners.

Chapter 3 Summary

- You need to learn and understand the basics of python as a foundation in advancing to the more complicated components.
- Familiarize yourself with the Python Official Website.
- Python can be used as an active translator or transcriber by interaction through the web.
- Python uses information that are not constant, these are used to keep the data.
- Any computer program requires interfacing between itself and the person using it.
- Numbers are the common language in computer programs including Python.
- Loop is a symbol used to represent repeated word/s or sentence/s in python programming.

Chapter 4:

Number Data Types



Working with number data types means that you will be working with numeric values that have to be created. They also need to have a value assigned to them.

Example:

Var b = 4

var = 5

Both numbers are variables; therefore, they are obviously going to be numbers in order for them to fall into the numeric data type. In the event that you need to, you can always delete any reference that you have to a numeric object. All you need to do is use the “del” statement.

Syntax:

Del var 1 [, var 2 [, var 3 [...., varN]]]

Deleting doesn't stop there. You have the option to delete single objects as well as multiple objects that are inside the same statement. The only thing that you need to do is list every object that

you want to be deleted, so the program does not delete the entire statement, forcing you to restart your code.

There are four different numeric types that you will work with while using the numerical data type:

1.
Complex numbers

Example: 5.43a

2.
Integers that are signed

Example: 43

3.
Floating point numbers

Example: -34.5

4.
Long integers that are either Octal or Hexadecimal persuasion

Example: -2345643245A

You need to keep in mind that there are a few things that Python does not allow when you are trying to make the number types work as they should inside of your code.

1.
You have the option to use a lowercase “l” if you are working with long numbers. However, you should try and avoid doing this. Instead, use an uppercase so that you are not accidentally confusing it with the number “1”.
2.
Complex numbers are made up as ordered pairs from the floating-point numbers. This denotes that they are

complex numbers, since there will be both real numbers and imaginary ones.

Converting Integers into Strings

The biggest reason that you will need to convert an integer into a string is so that you do not have to deal with type error messages. While you convert a number into a string, it will then be able to align the results in a table; that way, you can view them easier. If you don't, then you have to concentrate on a number so that you are able to enumerate the number. While doing the conversion, you will be using the "str" function. The process is simple as long as you follow these steps:

Step 1: Open the Python editor.

Step 2: Enter your code with the "str" function.

Step 3: After you press "enter", the string function will be executed.

If you do not use the "str" function, you will see a type error message that informs you that it is not able to concatenate the "str" and the "int" objects that are located in the code.

Example:

```
A = raw_input()
```

```
M = []
```

For the inside range (3, int(a)):

```
L = raw_input()
```

```
D []
```

```
E, q = (int(u) for u in w.split (' '))
```

```
Y = pow (s, t)
```

```
Str()
```

```
z.append(e [4])
```

```
for e in q:  
    result a
```

Converting One Data Type to Another

As you saw previously, you are able to use functions while converting one data type into another. When it comes to operations like subtraction and addition, your integer has to be converted into a floating number automatically by the program that you are using; this is in case one of the operands that is being used is already a float when you begin your conversion.

Example:

3 + 4.2

7.2

As you can see, the integer is going to be 3 while the float is 4.2 because it contains the decimal point.

You will be able to use functions such as `float()` and `complete()` so that you can convert the numbers in your expression. This will also convert the numbers that are inside of a string.

Example:

`Int(3.4)`

3

`Int(-2.4)`

-2

`Float(3)`

3.0

`Complex(3+ 4d)`

3 + 4d

While converting, you should begin to see that you are going from floating point numbers to integers. The number is going to be truncated, which means that it will take the number that is closest to zero.

Operators for Arithmetic

Math is part of our daily lives even if we do not realize it. The programs that we use each day are going to incorporate math that we do not see because it is automatically programmed to do the math in the code that the developer wrote. Doing simple math is easy for everyone, but when it is not simple, that is when most people prefer to have some help from other tools. This is where Python comes in.

It is easy to place your equation into a program and read off the result, which is exactly what Python does for you! However, your result is dependent on you placing the correct equation into the program. If you are not placing the proper equation into the program, then you will get the wrong answer. However, it does not matter if you are working with simple math or complex equations: Python is able to do it all!

Here are some of the most common operators that you will use while you are working with Python:

1. Modulo (%)
2. Addition (+)
3. Floor division (//)
4. Square root (math.sqrt)
5. Negation (-x)

6. Subtraction (-)
7. Division (/)
8. Exponent (a^{**n})
9. Absolute value (abs())
10. Multiplication (*)

Keep in mind that, while using the square root function, you will be required to load the math module for that particular operation. While doing this, you will be entering a code that is listed at the top of the file being used.

A bad thing about math is that, with Python, there are some limitations that you will run into with floating point numbers, as well as the rounding of these numbers. A lot of users experience error messages, or even unexpected results. For example, while doing division on a floating-point number (6.0 / 2.0), you will receive the correct answer. However, while doing floor division, you will get an output of a number that does not make sense for the equation that was entered.

Floor division was first introduced in the 2x version of Python as a solution for working with integers and longs. However, true division has to be used while working with complex or float numbers because the results can be unexpected. As you move on and look at Python 3.x, the true division was created to make sure that it can be used on any number that is entered into the program. However, users still experience issues.

The issues can be solved by inserting a set of parentheses around the division sign so that whenever the program rounds, you will receive the proper answer.

Another thing that you should keep in mind is that math will always use the PEMDAS system. You most likely learned this in elementary school and thought that you'd never use it again. Sorry about this, but here it is again, and it will be used by the Python program while it executes the equations entered into its code.

Here is a little refresher of what the acronym PEMDAS stands for:

P - Parentheses

E - Exponents

M - Multiplication

D - Division

A - Addition

S - Subtraction

Example:

$$8 + (32 - 2) * 2 / 4 - 2 = ?$$

If you follow the PEMDAS rules, the first step will be the Parentheses.

$$8 + (\underline{32 - 2}) * 2 / 4 - 2 \rightarrow 8 + (\underline{30}) * 2 / 4 - 2$$

Next is the Exponent. If you examine the equation, you will notice that there are no exponents. Therefore, you can move on to the next step, which is Multiplication.

$$8 + (\underline{30}) * \underline{2} / 4 - 2 \rightarrow 8 + \underline{60} / 4 - 2$$

Next will be Division.

$$8 + \underline{60} / \underline{4} - 2 \rightarrow 8 + \underline{15} - 2$$

Then, Addition.

$$\underline{8 + 15} - 2 \rightarrow \underline{23} - 2$$

And finally, Subtraction.

$$\underline{23} - \underline{2} \rightarrow \underline{21}$$

As you can see, there are a lot of steps, but this is what Python does behind the scenes in order to provide you with the proper answer to

your mathematical equation.

Comparison and Relational Operators

Comparison operators examine the values that fall on each side and determine what the relationship is between them. You may also know these as relational operators.

The operators are:

1.

`<=`: The value that is found on the left side is less than or equal to the value that is found on the right side. If this is correct, then the condition is marked as true.

Example: `2 <= 4`

2.

`==`: The values are equal. If this is correct, then the value is true.

Example: `6 == 6`

3.

`>=`: The value that is found on the left side is equal to or greater than the value that is found on the right. If this is correct, then the condition is found to be true.

Example: `6 >= 2`

4.

`!=`: The values are not equal, and if they are not, then the condition is true.

Example: `1 != 4`

5.

<: The value on the left is less than the value on the right.

Example: $4 < 9$

6.

>: The value on the left is greater than the value on the right.

Example: $9 > 3$

Assignment Operators

Assignment operators explain where the value is assigned after the function is complete.

1.

//=: This is used with floor division. The operator is assigned to the value that is located on the left.

Example: $4 //= 2$ equals $4 =// 2$

2.

=: The value on the right is always assigned to the value found on the left.

Example: $4 = 2 + 2$

3.

**=: The exponent and this go before the exponent first, and then everything is assigned to the left side.

Example: $4 **= 2$ equals $4 = 4 ** 2$

4.

+=: “Add” and the right operator are added together before being assigned to the left side.

Example: $4 += 2$ equals $4 = 4 + 2$

5.

$\%=:$ Modulus and the modulus are done before it is all moved to the left side.

Example: $4 \%= 2$ equals $4 = 4 \% 2$

6.

$-=:$ Subtract and subtraction are done before moving it to the left.

Example: $4 -= 2$ equals $4 = 4 - 2$

7.

$/=:$ Divide and division are done before everything is moved to the left.

Example: $4 /= 2$ equals $4 = 4 / 2$

8.

$*=:$ Multiple and multiplication are placed on the result before it is moved over to the left.

Example: $4 *= 2$ equals $4 = 4 * 2$

Bill Calculator

Sometimes it can be hard to understand math without looking at it in a realistic way. Therefore, we are going to examine a problem that many people face, which is how much their bill will be after they go out to eat once they add together the cost of their meal, the tax, and the tip. A lot of people tip based on service. However, a traditional tip is 15%.

So, if your meal is \$45 and the tax is 7%, plus you add on a 15% tip, how much will your meal be in total?

Dinner = 45

Since you have another variable (tax), you will have to add it into your equation because the restaurant is not going to remove your tax. By doing this, you will have to divide the 7% by 100 in order to produce a decimal.

Tax = 0.07

The last variable is how much you're leaving for a tip. You'll have to create a third variable for the tip, and, just like with the tax, you'll want to work with a decimal. So, divide 15% by 100.

Tip: 0.15

Now that you have your three variables, the proper values have to be assigned to them. For your equation, you will need to assign the value for your dinner so that it is the value multiplied by the tax that is being paid for the meal.

$$\text{Dinner} = \text{dinner} + (\text{dinner} * \text{tax}) = 45 + (45 * .07) = 48.15$$

The result is \$48.15. So, your dinner costs \$48.15 with tax.

You're not done yet, though! You can't forget to tip the waiter!

$$\text{Final amount} = 48.15 + (\text{dinner} * \text{tip}) = 48.15 + (45 * .15) = 55.37$$

Adding in the tip, you will be spending a total of \$55.37 for dinner.

Chapter 4 Summary

- The biggest reason that you will need to convert an integer into a string is so that you do not have to deal with type error messages.
- While doing the conversion, you will be using the “str” function.
- While using the square root function, you will be required to load the math module for that particular operation.
- Comparison operators examine the values that fall on each side and determine what the relationship is between them.
- Assignment operators explain where the value is assigned after the function is complete.

Chapter 5:

Conditional Statements



Another fun topic that we get to spend some time working with when it comes to the Python language is the idea of conditional statements. These are also called the if statements in some cases, or even the decision statements. Learning how to use these will basically teach your computer how to react to the input from the user, even if you are not there to control what is going on.

There are going to be times in your coding when you will want the program to be able to make some decisions or do some actions on its own, based on what the user tells it, without you needing to go through and code in every piece of the puzzle. Any time that the user is allowed to put in an answer that is all their own, rather than having to pick out from a selection of answers that you provide, then you are working with the decision control statements, or the conditional statements, to make this happen.

There are a few varieties when it comes to working with these kinds of statements, and the one that you go with will be based on what you are trying to do with the code. You can work with the if statement, the if else statement, and the elif statements.

The first option here that we are going to take a look at is known as the if statement. These are going to be pretty simple to work with, and there is not necessarily a lot of power that is behind them. But they will work on the idea that the answer that you get from the user is either seen as true or as false. If it is true, the program will continue on, and if the answer is seen as false, then the program will stop.

You can imagine already that there is going to be a bit of a problem with using the if statement in a lot of cases, and this is why you may not see it all that often. But it is still a good option to start out with when you are learning how these conditional statements are going to work. A good example of how you will be able to use the if statement will be the following:

```
age = int(input("Enter your age:"))

if (age <=18):

    print("You are not eligible for voting, try next election!")

    print("Program ends")
```

Let's explore what is going to happen with this code when you put it into your program. If the user comes to the program and puts that they are younger than 18, then there will be a message that shows up on the screen. In this case, the message is going to say "You are not eligible for voting, try next election!" Then the program, as it is, is going to end. But what will happen to this code if the user puts in some age that is 18 or above?

With the if statement, nothing will happen if the user says that their age is above 18. The if statement just has one option and will focus on whether the answer that the user provides is going to match up with the conditions that you set with your code. The user has to put in that they are under the age of 18 with the if statement in this situation, or you won't be able to get the program to happen again.

You naturally want the user to put in their actual age when they join the program, rather than only putting in the right answers. And if nothing shows up in your program when the user puts in the wrong age, this is going to leave them with something that makes no sense, or even a program that ends. It's likely that this is not what you want to work with.

This is where the if else statements are going to start showing up, and you will quickly find that they are more useful to work with than the plain if statements. The if else statements are going to work with the idea that we just did and then takes it a step further. The point with the if else statement is to make sure that the program does something, no matter what answer they are able to give to the program.

Going with the idea of the example that we talked about above, you may want to go through and separate the people into two groups. You may have a group who is 18 and under, and a group that is over 18 years old. This is something that the if else statement is going to be able to help you work with, and will ensure that, no matter what answer the user adds in for their age, something comes up. A good example of the code that you are able to use when it is time to bring in the if else statements include:

```
age = int(input("Enter your age:"))

if (age <=18):

    print("You are not eligible for voting, try next election!")

else

    print("Congratulations! You are eligible to vote. Check out your local
polling station to find out more information!")

    print("Program ends")
```

As you can see, this really helps to add some more options to your code and will ensure that you get an answer no matter what results the user gives to you. You can also change up the message to say anything that you want, but the same idea will be used no matter the answer that the user gives.

The example above is going to be a pretty simple one to work with. You can add in as many possibilities to the if else statements as you would like and you do not have to limit yourself to just two options as we did above. If you

only need to work with two options, then this is fine to stick with. But there are lots of codes that need to expand to more, and the if else statement is going to help with that as well.

For example, maybe you want to split the individuals who come to your program into five different age groups rather than just the two from before. You can just go through and add an if part of the statement, along with a message that you would like to go with it and continue on with the if else statements. You can technically add in as many options as you would like based on the kind of code that you are trying to develop.

Another example of using the if else statement is when you are creating a program that wants the person to pick out their favorite type of drink. There are a lot of options out there when it comes to tasty drinks, and you can definitely expand out the if else statements. Maybe you pick out a few options like pop, milk, juice, and coffee. Then you can add in the else statement that is going to be your “catch-all” in case the user decides to pick water or something else as their favorite drink. When this is in place, no matter what answer the user gives to the program there will be some kind of result that shows up.

Adding a catch-all to the end of your code, or the “else” part of this, can be important. You can’t always think about all the different examples that the person may put in. You could put a hundred options into your code (which would take a lot of time and be messy and not really necessary), and then the user will name a color differently or pick the one color that you forgot. If you don’t have that as an option, then the program won’t know how to behave from there.

The else statement here is important because it helps you to catch all of the remaining answers that the user could give you. If you don’t have a statement in the code to handle the answer that the user gives, then the else statement will make sure to get you covered. Just make sure that you have that else statement in place to get it done.

The elif statements

We have spent some time talking about two of the conditional statements that are available for Python coding. The first ones, the if statements are a

good place to start to get some practice with the conditional statements. They are based on the idea of the answer the user giving you is true or false. If the user gives an answer that is seen as true, then the program will finish up what you have next in the code. If the answer is seen as something false, then the program is going to end. It is meant to be something that is simple to work with and can give you some practice with writing codes.

Then you can move on to the if else statements if your code needs something a bit more to it. With the if else statements, we took this a bit further and set up something that is going to make sure that the user is going to get some kind of results, no matter what answer they put into the system. We even looked at a few examples of how you would be able to use the if else statement so we can see how they are different compared to the if statements.

And the third type of conditional statement that you can work with when you want to code in Python is known as the elif statement. The elif statement is going to be great to work with because it allows the user to look at a few options that you can present them with, and then, based on the kind of answer that the user gives, the program is going to execute the results that you added into the code for it.

You will actually see a lot of different programs that are going to rely on the elif statements. One place where you may see this kind of conditional statement is when you play a game, and there is a menu that starts up on the program then this is a good sign that an elif statement is being used. These conditional statements are going to be useful when it comes to providing a few options to the user, rather than one or two.

When you use the elif statements, you get a bit of freedom with what you are going to add into the code. You can choose to add in just a few options, or you can add in quite a bit, as long as you write this code out in the right way and you double check that the right function is put in the right place. In addition, you may want to keep the number of these that you use to a minimum because having too many is going to add some complexity to the code. You have to decide if this is what you would like to do.

One of the best ways to determine how to use the elif statements and whether it is going to work in the code that you are trying to write is to look

at some examples of how the elif statement works. The following syntax can be used to write an elif statement:

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

This is a pretty basic syntax of the elif statement and you can add in as many of these statements as you would like. Just take that syntax and then place the right information into each part and the answer that is listed next to it. Notice that there is also an else statement at the end of this. Don't forget to add this to your code so that it can catch any answer that the user puts in that isn't listed in your elif statements.

To help you better understand how these elif statements work and how the syntax above is going to work, let's take a look at a little game that you can create using these statements:

```
Print("Let's enjoy a Pizza! Ok, let's go inside Pizzahut!")  
print("Waiter, Please select Pizza of your choice from the menu")  
pizzachoice = int(input("Please enter your choice of Pizza:"))  
if pizzachoice == 1:  
    print('I want to enjoy a pizza napoletana')  
elif pizzachoice == 2:  
    print('I want to enjoy a pizza rustica')  
elif pizzachoice == 3:  
    print('I want to enjoy a pizza capricciosa')
```

else:

```
print("Sorry, I do not want any of the listed pizza's, please bring a Coca Cola for me.")
```

When you add this into your code, the user is going to have the benefit of going through this part of the program and making the choice based on what they want. And if you get it set up the proper way, you will have the right answer coming up for them. So, if the user decides that they would like to choose the pizza rustica, they would need to go through and select number 2. If they want to have just a drink and none of the pizza options that are listed, then they would click on that.

Remember you are able to add in as many options as you would like to the conditional statements of the elif statement. It is all about what works best for your program. You may have four options, or you can have twenty, but try to keep these to just the ones that your program needs to function in the proper manner for your needs.

As you can see here, the conditional statements are going to work well to help you gain more power in your codes, and they will help you to get the program to work, even if you are not able to guess all of the answers that someone is going to give you in the program. It allows the program to make some of the decisions without you while ensuring that your code is going to behave while the user is interacting with it. Make sure to try out some of the examples of conditional statements in the compiler to see how these work and to get more familiar with the way that you are able to use these.

Chapter 5 Summary

- A good example of how you will be able to use the if statement will be the following:

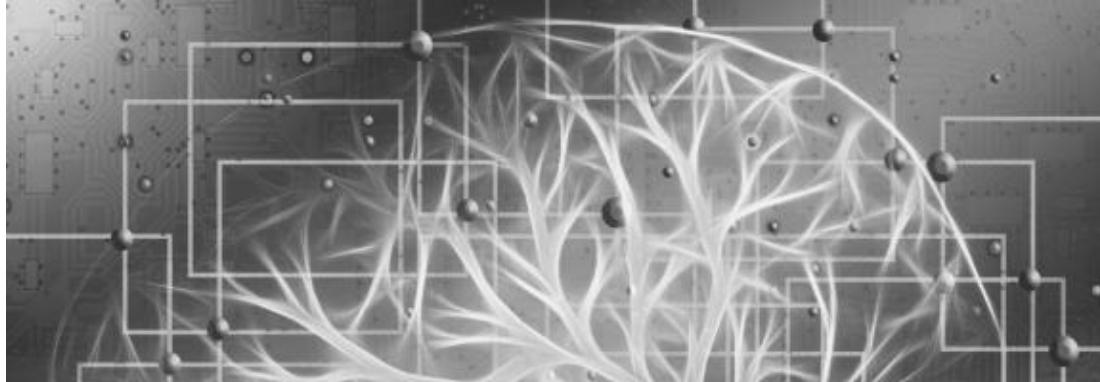
```
age = int(input("Enter your age:"))
if (age <=18):
print("You are not eligible for voting, try next election!")
print("Program ends")
```

- A good example of the code that you are able to use when it is time to bring in the if else statements include:

```
age = int(input("Enter your age:"))
if (age <=18):
print("You are not eligible for voting, try next election!")
else
print("Congratulations! You are eligible to vote. Check
out your local polling station to find out more
information!")
print("Program ends")
```

Chapter 6:

Working With The Python Files



The next thing that we need to focus on when it comes to working with Python is making sure we know how to deal with these kinds of codes is working with the Python files. There are going to be times when you are working with some data in these codes, and you will want to store it then while ensuring that it is accessible for you to pull up and use when the data is needed later. You do have some choices in the way that you save this data, how it is going to be found later on, and how it is going to react in your code.

When you work with the files, you will find that the data is going to be saved on a disk, or you are able to re-use in the code over and over again as much as you would like. This chapter is going to help us learn a bit more

about how to handle some of the work that we need to do to ensure the files behave the way that they should, and so much more.

Now, we are going to enter into file mode in the Python language, and this allows you to do a few different options along the way. A good way to think about this is that you can think about it like working on a file in Word. At some point, you may try to save one of the documents that you are working with so that it doesn't get lost and you are able to find them later on. These kinds of files in Python are going to be similar. But you won't be saving pages as you did on Word, you are going to save parts of your code.

You will find with this one that there are a few operations or methods that you are able to choose when it comes to working with files. And some of these options will include:

- Closing up a file you are working on.
- Creating a brand new file to work on.
- Seeking out or moving a file that you have over to a new location to make it easier to find.

Creating your new files

The first task that we are going to look at doing here is working on creating a file. It is hard to do much of the other tasks if we don't first have a file in place to help us out. If you would like to be able to make a new file and then add in some code into it, you first need to make sure the file is opened up inside of your IDLE. Then you can choose the mode that you would like to use when you write out your code.

When it comes to creating files on Python, you will find there are three modes that you are able to work with. The three main modes that we are going to focus on here includes append (a), mode(x) and write(w).

Any time that you would like to open up a file and make some changes in it, then you would want to use the write mode. This is the easiest out of the three to work with. The write method is going to make it easier for you to get the right parts of the code set up and working for you in the end.

The write function is going to be easy to use and will ensure that you are able to make any and all additions and changes that you would like to the file. You can add in the new information that you would like to the file,

change what is there, and so much more. If you would like to see what you are able to do with this part of the code with the write method, then you will want to open up your compiler and do the following code:

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

From here, we need to discuss what you are able to do with the directories that we are working with. The default directory is always going to be the current directory. You are able to go through and switch up the directory where the code information is stored, but you have to take the time, in the beginning, to change that information up, or it isn't going to end up in the directory that you would like.

Whatever directory you spent your time in when working on the code is the one you need to make your way back to when you want to find the file later on. If you would like it to show up in a different directory, make sure that you move over to that one before you save it and the code. With the option that we wrote above, when you go to the current directory (or the directory that you chose for this endeavor, then you will be able to open up the file and see the message that you wrote out there.

For this one, we wrote a simple part of the code. You, of course, will be writing out codes that are much more complicated as we go along. And with those codes, there are going to be times when you would like to edit or overwrite some of what is in that file. This is possible to do with Python, and it just needs a small change to the syntax that you are writing out. A good example of what you are able to do with this one includes:

```
#file handling operations  
#writing to a new file hello.txt
```

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

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

What are the binary files?

One other thing that we need to focus on for a moment before moving on is the idea of writing out some of your files and your data in the code as a binary file. This may sound a bit confusing, but it is a simple thing that Python will allow you to do. All that you need to do to make this happen is to take the data that you have and change it over to a sound or image file, rather than having it as a text file.

With Python, you are able to change any of the code that you want into a binary file. It doesn't matter what kind of file it was in the past. But you do need to make sure that you work on the data in the right way to ensure that it is easier to expose in the way that you want later on. The syntax that is going to be needed to ensure that this will work well for you will be below:

```
# write binary data to a file
# writing the file hello.dat write binary mode
F = open('hello.dat', 'wb')
# writing as byte strings
```

```
f.write(b"I am writing data in binary file!/n")
```

```
f.write(b"Let's write another list/n")
```

```
f.close()
```

If you take the time to use this code in your files, it is going to help you to make the binary file that you would like. Some programmers find that they like using this method because it helps them to really get things in order and will make it easier to pull the information up when you need it.

Opening your file up

So far, we have worked with writing a new file and getting it saved, and working with a binary file as well. In these examples, we got some of the basics of working with files down so that you are able to make them work for you and you can pull them up any time that you would like.

Now that this part is done, it is time to learn how to open up the file and use it, and later even make changes to it, any time that you would like. Once you open that file up, it is going to be so much easier to use it again and again as much as you would like. When you are ready to see the steps that are needed in order to open up a file and use it, you will need the following syntax.

```
# read binary data to a file
```

```
#writing the file hello.dat write append binary mode
```

```
with open("hello.dat", 'rb') as f:
```

```
    data = f.read()
```

```
    text = data.decode('utf-8')
```

```
    print(text)
```

the output that you would get from putting this into the system would be like the following:

Hello, world!

This is a demo using with

This file contains three lines

Hello world

This is a demo using with

This file contains three lines.

Seeking out a file you need

And finally, we need to take a look at how you are able to seek out some of the files that you need on this kind of coding language. We already looked at how to make the files, how to store them in different manners, how to open them and rewrite on them, and then how to seek the file. But there are times where you are able to move one of the files that you have over to a new location.

For example, if you are working on a file and as you do that, you find that things are not showing up the way that you would like it to, then it is time to fix this up. Maybe you didn't spell the name of the identifier the right way, or the directory is not where you want it to be, then the seek option may be the best way to actually find this lost file and then make the changes, so it is easier to find later on.

With this method, you are going to be able to change up where you place the file, to ensure that it is going to be in the right spot all of the time or even to make it a bit easier for you to find it when you need. You just need to use a syntax like what is above in order to help you make these changes.

Working through all of the different methods that we have talked about in this chapter are going to help you to do a lot of different things inside of your code. Whether you would like to make a new file, you want to change up the code, move the file around, and more; you will be able to do it all using the codes that we have gone through in this chapter.

Chapter 6 Summary

- When you work with the files, you will find that the data is going to be saved on a disk.
- You won't be saving pages as you did on Word, you are going to save parts of your code.
- Any time that you would like to open up a file and make some changes in it, then you would want to use the write mode.
- The write function is going to be easy to use and will ensure that you are able to make any and all additions and changes that you would like to the file.
- With Python, you are able to change any of the code that you want into a binary file.

Chapter 7: **How To Work Classes And Objects**



One of the things that you will really enjoy working on when it comes to Python is the organization that comes with it. This kind of coding language is going to spend time dividing everything up into classes and objects. This allows for more organization, ensures that every part of the code has a place and won't get lost, and really makes it easier for the beginner programmer to get things done without as much hassle.

Before we go through and create a new class, it is time to look a bit closer at what these classes and objects are all about. The objects are going to be anything that you are able to find in the real world. You will have a lot of objects that can work in the Python language, and these are going to help to power the code that you are working with, and will ensure that your code behaves in the manner that you would like. You can have any kind of object that you want, and they would match one of the objects that you want to do in the real world.

And then we have the classes. The best way to think about the classes in our Python code is as a little box that holds onto those objects that you created before. This helps to provide us with some organization when it comes to the code because we can place all of the objects that we want into each class, and then easily pull them back out at a later time when we want.

You can make as many classes as you would like in the code, and you get the benefit of being able to add in any objects, and as many objects, as you would like into each class. The biggest thing to remember here is that you need to make sure that the objects that belong to each class go together and make sense together. If someone else came in and looked at the coding that you did, and looked into one of the classes, would they be able to tell why all of those objects were found in the same class?

This doesn't mean that all of the objects in one class have to be exactly the same. But they need to make sense out of it as much as possible. You do not have to have a class that is just cars, for example, unless you would like to have it this way. But you could also go with a class that is going to contain different vehicles and this would be just fine as well.

How to Create a Class

With the information behind us about what these classes and objects are all about, it is time for us to take a look at the steps that we need to follow in order to create one of our own classes along the way. It is important to learn how to create our own classes because it is going to make sure that we have everything in place and we won't lose anything at the same time.

To make sure that you get the class ready to go, you need to put the right kind of keyword in place and then go through and name the class. You get the freedom here to give the class any name that you would like, but it is important to come up with a name that makes sense, and you have to place the name after the keyword so that the compiler knows what is going on.

After naming the class, it is time to name a subclass, which will be placed inside parentheses to stick with proper programming rules. Make sure that at the end of that first line, whenever you create a class, that you add in a semicolon. While this isn't technically needed with most newer versions of

Python and the code is going to work even if you forget this part, it is still considered part of coding etiquette to do this so make sure that you put it in.

Writing a class sounds more complicated than it really is, so let's stop here and look at an example of how you would write this out in Python. Then we can discuss what the parts mean and why they are important. A good example of creating a class with Python includes the following:

```
class Vehicle(object):  
    #constructor  
  
    def __init__(self, steering, wheels, clutch, breaks, gears):  
        self._steering = steering  
        self._wheels = wheels  
        self._clutch = clutch  
        self._breaks = breaks  
        self._gears = gears  
  
    #destructor  
  
    def __del__(self):  
        print("This is destructor....")  
  
    #member functions or methods  
  
    def Display_Vehicle(self) :  
        print('Steering:', self._steering)  
        print('Wheels:', self._wheels)  
        print('Clutch:', self._clutch)  
        print('Breaks:', self._breaks)  
        print('Gears:', self._gears)  
  
#instantiate a vehicle option
```

```
myGenericVehicle = Vehicle('Power Steering', 4, 'Super Clutch', 'Disk Breaks', 5)
```

```
myGenericVehicle.Display_Vehicle()
```

To see how this is going to work well for your needs and how the class can be set up, you should open up your compiler and type in the code that we have above. This will ensure that you are going to be able to test out the code and see which kind of class you have been able to create. This is a simple code that ensures we have the right objects that go into the different classes that we assigned, and will make sure that when we open up that class, it is all going to open up and work the way that we would like.

How to access some of the members of our class

As you look at the code that we wrote above, you can see that we took some time to go through and create a class and then added in a few objects to this as well. This is going to make it so that our objects are in the same place and can make it easier to pull them all out and make them easier to use in the long run. But now that the class is done, it is time to look at some of the objects that we put into that class and then learn how to pull them out.

In this part of the code, we have to learn the steps that are needed to access members of that class that we were able to create above. You want to make sure that your text editor and your compiler will be able to recognize all of the classes that you design later on because this is going to ensure that both of those are able to execute the code in the right manner.

Now, to set up the code that you want to write, you need to make sure that you set it up right, and there are a few methods that work for this. We are going to look at the accessor method to get this done because it is the easiest in most cases, and it is the one that most programmers are going to work with as well. To understand how you would use the method of the accessor function, look at the code below to help you get started:

```
class Cat(object)
```

```
    itsAge = None
```

```
    itsWeight = None
```

```
itsName = None

#set accessor function use to assign values to the fields or member vars

def setItsAge(self, itsAge):
    self.itsAge = itsAge

def setItsWeight(self, itsWeight):
    self.itsWeight = itsWeight

def setItsName(self, itsName):
    self.itsName = itsName

#get accessor function use to return the values from a field

def getItsAge(self):
    return self.itsAge

def getItsWeight(self):
    return self.itsWeight

def getItsName(self):
    return self.itsName

objFrisky = Cat()
objFrisky.setItsAge(5)
objFrisky.setItsWeight(10)
objFrisky.setItsName("Frisky")

print("Cats Name is:", objFrisky.getItsname())
print("Its age is:", objFrisky.getItsAge())
print("Its weight is:", objFrisky.getItsName())
```

Classes are not meant to be difficult to work with. They are perfect for helping you to take care of your information and keep it in order so that it makes the most sense. You can create any kind of class that you would like

and fill it up with any objects that you like, as long as those objects match each other in some way. Both the objects and classes are going to make a difference in your code to keep it organized, easy to read, and working properly.

Chapter 7 Summary

- The objects are going to be anything that you are able to find in the real world.
- The best way to think about the classes in our Python code is as a little box that holds onto those objects that you created before.
- After naming the class, it is time to name a subclass, which will be placed inside parentheses to stick with proper programming rules.
- Writing a class sounds more complicated than it really is.

Chapter 8:

Strings



A string is an ordered series of Unicode characters which may be a combination of one or more letters, numbers, and special characters. It is an immutable data type which only means that once it is created, you can no longer change it.

To create a string, you must enclose it in either single or double quotes and assign it to a variable. For example:

```
>>>string_one = 'a string inside single quotes'  
>>>string_double = "a string enclosed in double quotes"
```

When you enclose a string which contains a single quote or an apostrophe inside single quotes, you will have to escape the single quote or apostrophe by placing a backslash (\) before it.

For example, to create the string, 'I don't see a single quote':

```
>>> string_single = 'I don\'t see a single quote.'  
>>>
```

When you use the print function to print string_single, your output should be:

```
>>>print(string_single)
```

I don't see a single quote.

>>>

Similarly, you will have to escape a double quote with a backslash (\) when the string is enclosed in double quotes.

Hence, to create the string = "He said: "You have been nominated as honorary president of the Mouse Clickers Club."":

```
>>>string_two = "He said: \"You have been nominated as honorary president of the Mouse Clickers Club. \\""
```

>>>

```
>>>print (string_two).
```

He said: "You have been nominated as honorary president of the Mouse Clickers Club. "

>>>

Likewise, a backslash within a string should be escaped with another backslash.

```
>>> string_wow = "This is how you can escape a backslash \\."
```

```
>>>print(string_wow)
```

This is how you can escape a backslash \.

>>>

Accessing Characters in a String

You can access individual characters in a string through indexing and a range of characters by slicing.

String Indexing

- The **initial character** in a string takes zero as its index number and the succeeding characters take 1, 2, 3... and so on as

index numbers.

- To access the string backwards, the **last character** takes -1 as its index.
- A space is also considered a character.

To illustrate indexing in strings, define a variable named “string_var” and assign the string “Python String” with the statement:

```
>>>string_var = "Python String"
```

<u>Index #</u>	0	1	2	3	4	5	6	7	8	9	1	1	1
<u>String</u>	P	y	t	h	o	n		S	t	r	i	n	g
<u>Index #</u>	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Example #1:

To access the first character on the variable string_var (the first character of Python String is “P”), enter the variable name “string_var” and enclose the integer zero (0) inside the index operator or square brackets [].

```
>>> string_var[0]
```

```
'P'
```

```
>>>
```

In this example, the first character of the string “Python String” is “P”. Since the first character takes zero as it’s index number, Python gives you the letter “P” as an answer.

Example #2:

To access the character on index 8, simply enclose 8 inside the square brackets:

```
>>> string_var[8]  
't'
```

>>>

Since “t” takes 8 as it’s index number, Python gives you the letter “t” as an answer.

Example #3:

To access the character on index 6, an empty space:

```
>>> string_var[6]  
' '
```

>>>

Since an empty space takes 6 as it’s index number, Python gives you ''(a space) as an answer.

Example # 4:

To access the last character of the string, you can use negative indexing in which the last character takes the -1 index.

>>> string_var[-1].

```
'g'
```

>>>

A string is an ordered list, so you can expect that the penultimate letter takes the -2 index and so on.

Hence, -5 index is:

```
>>> string_var[-5]  
't'
```

>>>

The Len() Function

There is a more sophisticated way of accessing the last character and it will prove more useful when you’re writing more complicated programs: the len() function.

The len() function is used to determine the size of a string, that is, the number of characters in a string.

For example, to get the size of the variable ‘string_var’, you’ll use the syntax:

```
>>>len(string_var)
```

13

>>>

By using the len() function, Python is able to calculate the number of characters in your string “Python String”. Do not forget that Python calculates a space as a character. Thus you get 13 characters.

Since the last character in the string takes an index which is one less than the size of the string, you can access the last character by subtracting 1 from the output of the len() function.

To illustrate, type the following on the command prompt:

```
>>> string_var[len(string_var)-1]
```

'g'

>>>

Some important notes about accessing strings through indexing:

- Always use an integer to access a character to avoid getting TypeError.
- Attempting to access a character which is out of index range will result to an IndexError.

Slicing Strings

You can access a range of characters in a string or create substrings using the range slice `[:]` operator. To do this interactively on a random string, simply type the string within single or double quotes and indicate two indices within square brackets. A colon is used to separate the two indices. The slice operator will give you a string starting with S[A] and ending with S[B-1].

The syntax is: S[A:B-1]

S: The string you wish to use

A: The starting character of the substring you want to create

B: The ending character of the substring you want to create
Examples #1:

```
>>>"String Slicer "[2:12]
```

'ring Slice'

>>>

<u>Index #</u>	0	1	2	3	4	5	6	7	8	9	1	1	1
<u>String</u>	S	t	r	i	n	g		S	l	i	c	e	r
<u>Index #</u>	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

In this example, we want to create a substring using the range of characters from “r” to “e”.

Let's use the syntax S[A:B-1] to understand this command;

- S: refers to the string “String Slicer”.
- A: 2 is the index number which is associated with the letter “r”.
- B-1: $12-1=11$ is the index number which is associated with the letter “e”

Therefore, the command says that we are looking for a substring which includes the characters from “r” to “e” in the “String Slicer” string. Therefore, the answer is “ring Slice”.

Example #2:

```
>>>"Programmer"[3:8]
```

'gramm'

>>>

Index #	0	1	2	3	4	5	6	7	8	9
String	P	r	o	g	r	a	m	m	e	r
Index #	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

You can also slice a string stored in a variable by performing the slicing notation on the variable using the following statements:

Example #3:

```
>>>my_var = "Python Statement"
>>> my_var[0:12]
'Python State'
>>>
```

Index #	0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1
String	P	y	t	h	o	n		S	t	a	t	e	m	e	n	t
Index #	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	6	5	4	2	2	1	0	9	8	7	6	5	4	3	2	1

Example # 4:

```
>>>my_var = "Python Statement"
>>> my_var[7:11]
'Stat'
```

>>>

When slicing or creating a substring, you can drop the first number if the starting character of the substring is the same as the initial character of the original string.

For example:

```
>>> test_var = "appendix"  
>>>test_var [:6]  
'append'  
>>>
```

Index #	0	1	2	3	4	5	6	7
String	a	p	p	e	n	d	i	x
Index #	-8	-7	-6	-5	-4	-3	-2	-1

In this example;

- The starting character is “a”, as is the first letter in “appendix”.
- The starting character of the substring is also “a”. Therefore, we can drop the first number. In fact, instead of writing `test_var[0:6]` you can simply write `test_var[:6]`. Similarly, if your substring ends on the last character of the string, you can drop the second index to tell Python that your substring ends on the final character of the original string.

```
>>> test_var = "appendix"  
>>> test_var[3:]  
'endix'
```

>>>

Concatenating Strings

Several strings can be combined into one large string using the **+ operator**. For example, to concatenate the strings "I", "know", "how", "to", "write", "programs ", "in", "Python", you can type:

```
>>>"I" + "know" + "how" + "to" + "write" + "programs" +  
"in" + "Python."  
>>>
```

You should get this output:

'I know how to write programs in Python.'

Likewise, you can concatenate strings stored in two or more variables. For example:

```
>>> string_one = "program "
>>> string_two = "is "
>>> string_three = "worth watching"
>>> print("An excellent "+string_one[:7] +" "+ string_two + " " +
string_three +".")
```

>>>

When you run the program (when you press enter), the output will be:

An excellent program is worth watching.

Take note that since a string is immutable, the acts of slicing and concatenating a string do not affect the value stored in the variable.

For example, assign the string "concatenate" to the variable same_string and slice it to return the characters from index 4 to 6:

```
>>> same_string = "concatenate"
>>> same_string[4:7]
'ate'
```

>>>

Now, print the value on the variable same_string:

```
>>> print(same_string)
```

concatenate

>>>

Notice how the slicing did not affect the original string at all.

Repeating a String

To repeat a string or its concatenation, you'll use the operator * and a number. This instructs Python to repeat the string a certain number of times.

For example, if you want to repeat the string *<>* five times, you can type the string on the command prompt and specify the number of times it should be repeated with *5.

```
>>>"*<>*" *5
```

Here's the output:

```
>>>'*<>**<>**<>**<>**<>*'
```

You can store the above string in a variable and apply the * operator on the variable to achieve the same result:

```
>>>sign_string = "*<>*"
>>>sign_string * 5
'*<>**<>**<>**<>**<>*'
```

```
>>>
```

Using the upper() and lower() functions on a string

The upper() and lower() functions can be used to print the entire string in uppercase or lowercase.

To illustrate, define a new variable 'smart_var' and use it to store the string "Europe".

```
>>>smart_var = "Europe"
```

To print the entire string in uppercase letters, simply type:

```
>>>print(smart_var.upper())
```

The screen should display this output:

```
EUROPE
```

To turn things around, print the entire string on lowercase by typing:

```
>>>print(smart_var.lower())
```

You'll get the output:

```
europe
```

The use of the upper() and lower() functions does not change the string stored at smart_var. You can prove this by entering the command:

```
>>>print (smart_var)
```

```
Europe
```

Using the str() function

You may sometimes need to print non-string characters as string characters. For example,a program may require you to print a string

along with integers or other number types. Python's str() function allows the non-string character to be converted to string characters. To illustrate, you can create a variable to store the integer 246. The variable can then be used as a parameter for the str() function.

```
>>>my_number = 246  
>>>str(my_number)  
'246'  
>>>
```

To print the string "My employee number is 246", you can type the following:

```
>>>my_number = 246  
>>> print("My employee number is " + str(my_number))  
My employee number is 246  
>>>
```

Python String Methods

There are several Python methods that can be used with string to support various operations:

The replace() method

The replace() method replaces a substring within an existing string with a new substring. Since you can't actually change the string on account of its immutable nature, replacing values necessitates the creation of a new string.

The find() method

The find() method is used to search for a given character or a sequence of characters in a string.

Example #1:

```
>>> s = "A string is an immutable character or series of  
characters."  
>>> s.find("string")
```

>>>

Index #	0	1	2	3	4	5	6	7
String	A	s	t	r	i	n	g	
Index #	-8	-7	-6	-5	-4	-3	-2	-1

In the above example, the `find()` method returned ‘2’ which is the index of the first character of the string ‘string’.

Example #2:

In the following example, `find()` returns ‘5’, the index of the first occurrence of the string ‘i’.

```
>>> s = "A string is an immutable character or series of characters."
```

```
>>> s.find('i' )
```

>>>

Example #3:

There are several i’s in the string and if you’re looking for the next ‘i’, you’ll have to supply a second argument which should correspond with the index immediately following index ‘5’ above. This tells the interpreter to start searching from the given index going to the right. Hence:

```
>>> s = "A string is an immutable character or series of characters."
```

```
>>> s.find('i' , 6)
```

>>>

Example #4:

The search found the second occurrence of letter ‘i’ at index 9. Besides specifying an argument for the starting range, you can also provide an argument to indicate the end of the search operation. You can do it backwards by applying negative indexing. For example, if you want to find the third occurrence of the letter ‘i’, you can give

the index ‘10’ as a second argument and provide an end to the search range with a third argument, -10.

```
>>> s = "A string is an immutable character or series of characters."  
>>> s.find('i', 10, -10)  
15  
>>>
```

Isalpha()

The method `isalpha()` returns True if all characters of a non-empty string are alphabetic **and there is at least one character** and False if otherwise.

```
>>> s = ("programs" )  
>>> s.isalpha()  
True  
>>> print(s.isalpha())  
True  
>>> b = ("programming 1 and 2")  
>>> b.isalpha()  
False  
>>>
```

As you can see the first 2 strings had only alphabetic characters (only letters) whereas the 3rd string had both alphabetic and numeric (letters and numbers) characters.

Isalnum()

The method `isalnum()` returns True if ALL the characters of a non-empty string are alphanumeric and False if otherwise

Example #1:

```
>>> b = "Programmer2"  
>>> b.isalnum()  
True
```

>>>

Example #2:

```
>>> a = "Programmer 1"
```

```
>>> a.isalnum()
```

```
False
```

>>>

As you can see, in the first example all characters are alphanumeric (characters which include both numbers and letters). However, the second example includes letters, numbers and **a space** between “programmer” and “1”. This is why Python returns False.

Isidentifier()

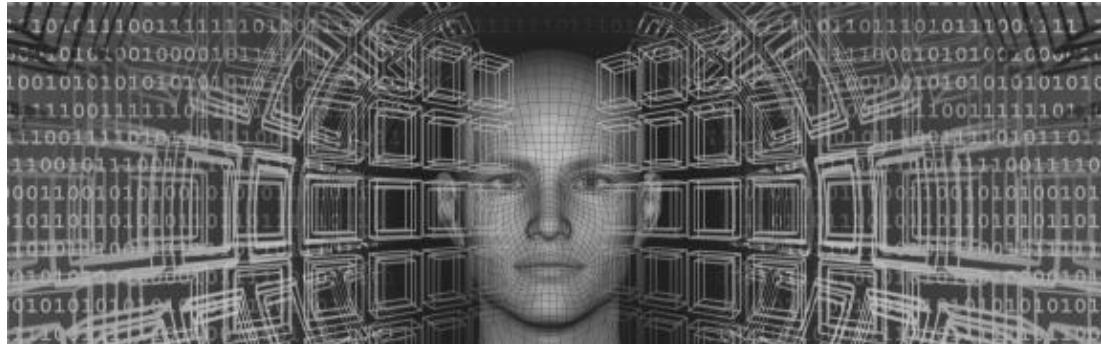
The isidentifier() method tests for the validity of a given string as an identifier and returns True if valid or False if otherwise.

Chapter 8 Summary

- A string is an ordered series of Unicode characters which may be a combination of one or more letters, numbers, and special characters.
- You can access individual characters in a string through indexing and a range of characters by slicing.
- There is a more sophisticated way of accessing the last character and it will prove more useful when you're writing more complicated programs: the len() function.
- You can access a range of characters in a string or create substrings using the range slice [:] operator.

Chapter 9:

Making Predictions With Algorithms



How are you feeling at this stage? Have you encountered any errors? Are you feeling as if you have gained some new knowledge?

Hopefully, things are going smoothly and you are grasping the concepts well at this point. Let's take a look at how to make predictions with algorithms in Python and what it means.

What is Predictive Analytics?

“Predictive Analysis” is regularly talked about with regards to building information, for instance, originating from instruments, specific sensors and associated frameworks in the real world. Business data, at an organization, for example, may incorporate exchange information, deals results, client dissensions, and promote data. Progressively, organizations settle on information-driven choices in light of this important aggregation of data.

With a significant growth in competition, organizations look for a competitive advantage in bringing items and administrations to open markets. Information-focused models typically enable organizations to take care of long-standing issues in creative and unique ways.

Manufacturers, for instance, often think that it is difficult to enhance just its equipment. Item designers can add prescient abilities to existing answers for increased incentives to the client. Utilizing prescient examination for hardware upkeep, or prescient support, can predict future product development disappointments, figure vitality accurately, and decrease working expenses. For instance, sensors that measure certain wave patterns and vibrations in car parts, and in turn, flag the requirements for upkeep before the car/automobile flops during use by an actual consumer.

Additionally, organizations utilize a prescient investigation to make more exact predictions, for example, estimating the increased demand for power on the electrical grids. These figures enable companies to do asset planning, like looking for other power plants, in order to be more efficient and effective.

To extricate an increase in value from all of the information, organizations apply calculations to vast information sets, utilizing new and upcoming technology tools, for example, Hadoop. The information sources may comprise value-based databases, hardware log files, pictures, audio/video, sensory details, or a number of other kinds of information. True innovation is often the result of using and combining data from a variety of different sources.

With this information, these technologies are of critical importance in the discovery of trends and patterns. Machine learning methods are utilized to discover commonalities and patterns in information and to estimate what the outcomes are going to be.

What does Predictive analysis do? What does it mean?

Predictable analytics allows groups in different job roles, ranging from financial, healthcare workers in pharmacy industries, and automobile. This particular analytical process is how we utilize the data that we have analyzed in order to make viable guesses which are largely based on the analyzed information.

Whew! Don't panic.

The great thing is that this process is a predictive model which allows for a systematic approach of delivering outcomes based on a certain set of common criteria.

To define what “predictive analytics” means, this process involves applying a certain statistical approach based on Python machine learning strategies and models which creates realistic and measurable estimations and predictions about future outcomes. Regularly, Python machine learning techniques are used in real-world problem-solving. For example, it is commonly used to estimate the value of something in the near future such as “How long can my word processor run before needing it to be replaced or require routine maintenance?”

Constructed on a set of criteria, it can also be used to guess certain customer behaviors. A great deal of banks and financial institutions use this to determine the creditworthiness of their customers, how likely they are to default on their mortgage or car loan, or the probability of excessive overdrafts each month. It’s pretty amazing.

Predictive analytics is primarily used in helping companies and organizations make future predictions and meet certain goals. Think about the most common goals of any business: stay in business, make money, and reduce excess waste through the analyzing of data, methods to decrease expenses and ability to offer employee bonuses if goals are met. To do something of this scale does require an extensive amount of various data types and inputting them into pre-built models that will ultimately generate concise, measurable, and most importantly—achievable outcomes to maintain a positive bottom line and support growth.

In order to make this click, let’s look back at what we said: “predictive analysis” is and what it’s for, as it relates to some real-world examples. These are not all inclusive by any means, and more can be found using a simple Google search and research.

Real World Examples of Predictive Analytics:

The Car Industry –Breakthrough technology in cars, designed to gather specific details and information regarding how fast the car is going, how far the car has traveled, its emission levels and the behaviors of drivers are now used with an extremely sophisticated predictive analysis model. This allows

the analysts to release extremely beneficial data for car manufacturers, insurance companies, and the racing circles.

Aviation –Determining the viability and health of an aircraft is an application developed by an aviation engineer, it helped improve the performance of aircraft speed and reduce costs to maintain and repair them. This particular application is used to test performance in every critical function of the plan from the take-off, to the control systems, all the way to the efficiency of the fuel and maximum take-off conditions.

Production of Energy –Electricity companies use predictive analytics in order to determine the cost and demand for electrical supplies. There are a ton of extremely sophisticated models that forecast access, patterns (future and past), the different changes in weather and many other factors.

Accounting and Financial Services –The Development of credit risk models is a prime example of predictive analytics in the real world. Nowadays, banks, credit unions, and many other financial institutions use these models and applications in order to determine a customer or potential client's credit risk.

Equipment and Machine Manufacturing —Testing and determining future machine weaknesses and failures. This particular application is used in helping to improve the efficiency of assembly lines and production of large equipment and machines and at the same time optimizing its operations and workforce.

Modern Medicine –This is last on the list, but certainly not least. Predictive analysis has been used in modern medicine to detect infections and common diseases and even pre-existing conditions. It's also a great way to bridge the communication gaps between those in the medical profession.

Pretty cool, huh? Can you find more ways that predictive analysis is used in real-world situations to improve our life, our economy, and our businesses?

Workflow in Predictive Analytics

You may or may not be familiar with predictive models at this stage of your learning, but you can think of a real-world example as to what meteorologists use in day to day weather forecasting.

A basic industry utilization of prescient models identifies any circuit that consumes power and allows a prediction to be made about the demand for power, as it relates here—energy. For this model, network administrators and brokers require precise conjectures of each circuit load to make important choices for integrating them into the electrical grid framework. Huge amounts of information are easily accessible and utilizing these prescient analytics, allowing matrix administrators to transform this data into noteworthy bits of knowledge that can be used to make important decisions and predictions.

Typically, a simple workflow for a predictive analytics model will follow these basic steps outlined here:

- Import information from changed sources, for example, web chronicles, databases, and spreadsheets.
- Information sources incorporate energy load data and information in a CSV record and national climate information demonstrating temperatures and varying dew points.
- Clean the information by evacuating anomalies and joining information sources.
- Distinguish information spikes; especially pinpoint missing information, or even bizarre outputs to expel from the information.
- Make a solitary table including energy load, temperature and dew point.
- Build a precise data model in light of the accumulated information.
- Predicting any type of energy source is a perplexing procedure with numerous factors. You may utilize neural systems to assemble and prepare a prescient model.
- Practice training through your data index to achieve diverse strategies. At the point when the preparation is finished, you can attempt the model against new information to perceive how well it performs.
- Coordinate the model into a front gauging framework in a production type of environment.

- When you locate a model that precisely gauges the outcomes, you can move it into your creation framework, making the examination accessible to programming projects or gadgets, including web applications, servers, or smartphones.

Your aggregated data tells a tale that is certainly complex. To withdraw the insights, you are going to need an extremely accurate design which is predictive. This might not be the best step as a beginner; nonetheless, it is here for reference to the entire picture of Python capabilities.

Predictive analysis is being modeled after major mathematical models to predict a conference or result. These designs forecast the desired outcome at some future time based on modifications placed into data inputs. Using a repetitive procedure, you create the models by choosing a training information set where you will proceed to test and further validate the information. After you examine it to ascertain its reliability and accuracy in predicting forecasts, play around with different methods until you find one that is comfortable for you. The important thing is that you choose one that you can understand, learn and apply without much effort.

To give you an idea of some good examples of such a method, you can try a time-series reversal model for predicting low and even high levels of flight traffic or fuel. Of course, this is certainly predicting based on a linear approach to speed compared to upload and continuing to be extremely beneficial in real-world estimation models for conjecture.

What is the Difference between Predictive Analytics & Prescriptive Analytics?

Businesses that have been able to successfully implement predictive analytics have a competitive advantage to problems, situations and good things in the future. Predictive analytics is a process that creates an estimation of what will happen next—literally. It also gives you tips about how to be able to make high-level decisions in a way that maximizes the information you wouldn't have access to.

Prescriptive analytics is just a branch of data analytics that makes use of designs that are predictive guesses to make for the most ideal outcomes. Prescriptive examination depends on advancement and tenets-based

procedures for decision-making on the most basic of levels. Anticipating any issues or strains on the framework is absolutely essential in the decision-making process. It means what is to be done is based on the prediction.

Chapter 9 Summary

- With a significant growth in competition, organizations look for a competitive advantage in bringing items and administrations to open markets.
- To extricate an increase in value from all of the information, organizations apply calculations to vast information sets.
- Predictable analytics allows groups in different job roles, ranging from financial, healthcare workers in pharmacy industries, and automobile.
- Predictive analytics is primarily used in helping companies and organizations make future predictions and meet certain goals.
- Real World Examples of Predictive Analytics:
 - *The Car Industry*
 - *Aviation*
 - *The Production of Energy*
 - *Accounting and Financial Services*
 - *Equipment and Machine Manufacturing*
 - *Modern Medicine*

Chapter 10:

Error Management



It should not be surprising that errors occur - applications are written by humans, and humans make mistakes. Most developers call application error exceptions, which means they are the exception to the rule. Because exceptions occur in applications, you need to detect them and fix them as much as you can. Detecting and dealing with an exception is called error handling or exception handling. To correctly detect errors, you must know the sources of error and know why the errors occur first. When you detect the error, you must handle it by capturing the exception. Capturing an exception means looking at it and possibly doing something about it.

Sometimes your code detects an application error. When this happens, you must throw or raise an exception. You discover that both terms are used for the same thing, which means that your code has encountered an error that can not be manipulated, and then passed the error information to another piece of code to manipulate (interpret, process, and, with a bit of luck, correct exception). In some cases, you make use of custom error message objects to convey information. Although Python has a large number of generic message objects that cover most situations, some are special. For instance, you may want to provide special support for a database application, and Python will not normally cover this event with a generic message object. It's essential to know when to control exceptions locally, when to give them to the code that requested your code, and when to create unique exceptions so that each part of the application understands how to handle the exception - all topics covered in this chapter.

You may also need to make sure that your application normally handles an exception, even if it means terminating the application. Fortunately, Python provides the final clause, which is always executed even in case of exception. You can put code to close files or perform other essential tasks in the code block associated with this clause. Even if you do not do this task all the time, this is the last topic in the chapter.

Know why Python does not understand you

Developers are often frustrated with programming languages and computers because they seem to do everything to cause communication problems. Of course, programming languages and computers are inanimate - we do not want anything from them. Programming languages and computers do not think either; they accept what the developer has to say literally. There is a problem.

Neither the computer nor Python "will know what you mean" when entering instructions as a code. Both follow the instructions you provide to the letter and as you provide them. You may not want to tell Python to delete a data file unless an absurd condition occurs. However, if you do not

clarify the conditions, Python will delete the file, whether the condition exists or not. When such an error occurs, people often say that the application contains a bug. The bugs are program errors that can be removed using a debugger. (A debugger is a unique kind of tool that allows you to pause or pause running applications, examine the contents of variables, and often dissect the application to see what makes it work.)

Errors occur in several cases when the developer makes assumptions that are not true. Of course, this includes assumptions about the user of the application, who probably does not care about the extreme level of attention you received when creating your application. The user will enter incorrect data. Again, Python will not know if the data is incorrect and will care and treat it even if its purpose is to prevent an incorrect entry. Python does not understand good or bad data concepts; It simply processes the received data according to the defined rules, which means that you need to define rules to protect users against themselves.

Python is neither proactive nor creative - these qualities only exist at the developer. When the user does something unexpected, or a network error occurs, Python does not create a solution to the problem. It only deals with the code. If you do not provide the code to handle the error, it is likely that the application will fail and fail incorrectly, possibly resulting in the transfer of all user data. Of course, the developer can not anticipate all possible error situations. This is why the most complex applications contain errors - omission errors in this case.

Some developers think they can create a code foolproof, despite the absurdity of thinking that such a code is possible. Smart developers assume that several bugs will go through the process of sorting the code, that users will continue to carry out unexpected actions, and that even the smartest developer cannot predict all error conditions possible. Always assume that your application is prone to errors that may cause exceptions. This way, you will have the necessary state of mind to make your application more reliable.

Chapter 10 Summary

- Applications are written by humans, and humans make mistakes.
- Errors occur in several cases when the developer makes assumptions that are not true.
- Python is neither proactive nor creative - these qualities only exist in the developer.
- Always assume that your application is prone to errors that may cause exceptions.

Chapter 11:

Functions



Although Python makes it easy to write expressions that fit a lot of logic into a single line of code, this isn't always the best way to build a program. In many cases, as you increase the number of steps performed by a single line, you decrease its readability. Fortunately, you can implement functions in order to clear up your code as well as perform the same series of operations multiple times, improving clarity and efficiency.

How Do I Use A Function?

Whether you are just using Python without any additions or you are using code written by other people, you will have access to many already-made functions that you can use in your code. You simply need to call the

function you wish to use by typing the function name followed by any parameters you need to pass in, like so:

```
function_name(parameter1, parameter2, etc.)
```

A parameter is simply variable information you give a function to work with, like the length and width values. Parameters don't have to be numerical values. They can also be different variable types, like strings or objects. The function then manipulates these values in order to perform certain actions, display desired information, and tell your program what to do next. Take a look at the following function:

```
def happy_birthday(name, age):
```

```
    print("Happy birthday ", name, "! You're ", age, " years old. Have a good year!"
```

```
    return 0
```

This function, which is named `happy_birthday`, accepts two parameters when it is called -- `name` and `age`. You would call it in your program with a line similar to this:

```
happy_birthday("Geraldine", 32)
```

using whatever name and age you'd like. Notice that the first parameter is a string, so it requires quotation marks. The second parameter is an integer, so no quotation marks are necessary.

The `happy_birthday` function then uses this information that you passed into it when you called it to display the following line:

```
Happy birthday Geraldine! You're 32 years old. Have a good year!
```

It then exits the function with the line `return 0` and proceeds with the remainder of your program.

You can probably already understand from this simple example how much you can improve the efficiency of your code by using functions. Imagine if you wanted to write this Happy Birthday message to six different people. Would you rather write this:

```
print("Happy birthday Amanda! You're 18 years old. Have a good year!")
```

```
print("Happy birthday Jose! You're 34 years old. Have a good year!")
```

```
print("Happy birthday Peter! You're 41 years old. Have a good year!")  
print("Happy birthday Lynn! You're 22 years old. Have a good year!")  
print("Happy birthday Amelia! You're 30 years old. Have a good year!")  
print("Happy birthday Jack! You're 17 years old. Have a good year!")  
or this:
```

```
happy_birthday("Amanda", 18)  
happy_birthday("Jose", 34)  
happy_birthday("Peter", 41)  
happy_birthday("Lynn", 22)  
happy_birthday("Amelia", 30)  
happy_birthday("Jack", 17)
```

By incorporating functions into your code, you eliminate the need to rewrite redundant parts of your program. While the happy_birthday function only saves you from manually writing out a single line, more complex functions can save you from much more, like having to manually perform complicated calculations repeatedly.

A Little More About Parameters

So far, you've been able to take a look at some functions that use one or more parameters to perform desired actions. You also learned in the last section that a function can have any number of parameters, including different variable types, and how to call those functions. However, what if you have a function with no parameters, or a function with default parameters? Take a look at the examples below to see how to call these types of functions.

No parameters necessary: Some functions don't require the user to input any information in order for them to run as desired. For instance, consider the following function:

```
def hello_funtion()  
    print("Hello!")  
    return 0
```

As you can see, `hello_function` doesn't require any parameters to be passed in and does not use any external information to perform any actions. You can simply call functions without parameters in the same way as any other function, but without putting any values in parentheses, like so:

```
hello_function()
```

The function will then run and produce the following output:

```
Hello!
```

Just because a function doesn't accept parameters, however, doesn't mean that the output will always need to be the same when that function is run. In the `hello_function` example that was the case -- the function will always only display `Hello!` when it is run -- but other functions may perform actions like generating a random number or choosing a random option from a predefined list. In those examples, even though there is no user-defined information passed in, the function will produce a varying output when it is run multiple times.

Parameters optional: In certain circumstances, you may want to be able to call a function that **can** accept a value as a parameter but doesn't **need** to. In order to do this, you can write your function using one or more default values. A simple function using a default value as a parameter might look something like this:

```
def happy_birthday(name=None):
    if name :
        print("Happy birthday ", name, "!")
    else:
        print("Happy birthday!")
```

In this version of the `happy_birthday` function, you can either specify the name of the person receiving the message or not, and the function will display the birthday message accordingly. You can actually call this function in three different ways. The first way is by passing in a string value for the name, like so:

```
happy_birthday("Stephanie")
```

which will output

Happy birthday Stephanie!

Alternatively, you can call the function without passing in any values, and the function will assign the default value of None to the name variable when it runs. So, if you call the happy_birthday function like so:

happy_birthday()

you'll receive the output

Happy birthday!

A third option is to pass in the value of None when you call the happy_birthday function:

happy_birthday(None)

which will output

Happy birthday!

Since the default value of name is None, this will result in the same output as if you called the function without passing in any parameters. Why? When the happy_birthday function checks the value stored in the name variable to determine whether or not to execute the if statement, it sees a value of None -- the same value that would have been assigned by default if you had left the parentheses blank when calling the function. So, the program doesn't execute the if statement and produces the same output whether the value of None is assigned manually or by default.

For instance, the following version of a happy_birthday function uses both standard and default parameters:

```
def happy_birthday(name, age=None):
```

```
    print("Happy birthday ", name, "!")
```

```
    if age:
```

```
        print("You're ", age, " years old!")
```

In this example, you can call the happy_birthday function like this:

happy_birthday("Amanda")

which will output

Happy birthday Amanda!

Or, you can call the happy_birthday function like this:

```
happy_birthday("Claire", 27)
```

which will output

Happy birthday Claire!

You're 27 years old!

The name parameter in this version of the happy_birthday function is required, but the age parameter is not. As you can see, the function only displays the second line about the person's age if a value for age was passed in when the function was called. Otherwise, it only displays the first line wishing the person a happy birthday.

When using a mix of standard and default parameter values, it is important that they are in the proper order in the function prototype -- standard values must go first, then default parameter values. In the version of the happy_birthday function using both a standard value and a default value, you may have noticed that the name value came first, and this is for good reason. Imagine if the order had been reversed, like so:

```
def happy_birthday(age=None, name):  
    print("Happy birthday ", name, "!")  
    if age :  
        print("You're ", age, " years old!")
```

Why won't this work? Since the function isn't guaranteed to receive a value for age, it has no way of knowing if the first parameter it receives is a value for age or for name. In the correct order -- standard parameter values first, default values last -- the program knows that the first parameter value should definitely be assigned to name, and if there **is** another value, that one should be assigned to age. If you have a function with multiple default values, like so:

```
def some_function(w, x, y=0, z=0):  
    print("w = ", w)  
    print("x = ", x)  
    print("y = ", y)
```

```
print("z = ", z)
```

```
return 0
```

a value for y must be defined in order to define a value for z, even if that just means manually setting y to None. For example, the following call to `some_function`:

```
some_function(2, 4, 6, 8)
```

will output

```
w = 2
```

```
x = 4
```

```
y = 6
```

```
z = 8
```

as expected, but what if you want to assign a value of 6 to z without assigning any value to y? You may be tempted to do the following:

```
some_function(2, 4, 6)
```

and just skip over the y parameter entirely; after all, it is optional. However, this will output

```
w = 2
```

```
x = 4
```

```
y = 6
```

```
z = 0
```

since the program has no way to know that you wanted to skip y and assign the value of 6 to z instead. In order to get the desired output, you would need to call `some_function` like so:

```
some_function(2, 4, 0, 6)
```

This will manually assign the value of 0 to y, which is its default value, and let the program know that it should assign 6 to the z variable since y has now been accounted for:

```
w = 2
```

```
x = 4
```

y = 0

z = 6

Where can I find premade functions to use in my code?

Before you start writing your own functions, you should familiarize yourself with functions that are already available -- why take your own time to rewrite something that already exists? Python actually has a variety of its own built in functions, or you can search for modules to import and use.

Using Python's built in functions: Once you have Python downloaded, you automatically have access to a long list of functions that you can use immediately in your code. This list includes -- but is not limited to -- the following:

- `abs(x)` is a function that is used to return the absolute value of a number `x`
- `bin(x)` is a function that is used to convert an integer `x` to a binary string
- `bool(x)` is a function that is used to convert a value `x` to True or False
- `chr(x)` is a function that is used to convert an integer `x` to a string
- `dict(x=2, y=5)` is a function that is used to create a dictionary
- `float(x)` is a function that is used to convert a string (containing decimal points) or a number to a floating point number
- `format(x, fs)` is a function that is used to format a value `x` to a format specified by `fs`
- `hex(x)` is a function that is used to convert an integer `x` to a hexadecimal string
- `id(x)` is a function that is used to return the unique integer identity of an object `x`
- `input(prmt)` is a function that is used to read a line from input (after displaying the optional string `prmt`) and return it converted to a string

- `int(x, base)` is a function that is used to convert a string or number `x` with an optional base of `base` into an integer object
- `isinstance(x, info)` is a function that is used to check whether or not an object `x` is a subclass or instance of a class defined by `info`
- `list(x)` is a function that is used to create a list from object `x`, which can be a sequence, set, dictionary, or iterator object
- `len(x)` is a function that is used to find the number of items in an object `x`, which can be a sequence or a collection
- `max(x, y)` and `min(x, y)` are used to find the largest and smallest elements in an iterable, or the largest and smallest of 2 or more parameters
- `object()` does not accept any parameters and is used to create a featureless object
- `open(file)` is a function that is used to open a file and return a file object
- `pow(x, y)` is a function that is used to determine `x` to the power of `y`
- `print(x)` is a function that is used to display an object `x` to the screen or a file
- `round(x, n)` is a function that is used to round a number `x` to `n` digits after the decimal point
- `slice(start, stop, step)` is a function that is used to create a slice object specified by `start`, `stop`, and `step`
- `sum(x)` is a function that is used to find the sum of the elements of an iterable `x`
- `type(x)` is a function that is used to determine the type of an object `x`

and many others. You can find a complete list of Python's built in functions -- as well as information on what they do and how to use them -- in the official Python documentation at DOCS.PYTHON.ORG .

Using functions defined in external modules: If the built in functions provided by Python aren't enough for you to accomplish everything you need to in your program, you can also import modules to use functions written and stored in external files. A module is simply a `.py` file containing

Python code, and it can define variables, classes, and functions. You can reference any Python file as a module.

In order to access the variables, classes, and functions defined in a Python module, you'll first need to import it for use in your program file. You can do this by adding a line at the top of your code consisting of the word import followed by the name of the module you wish to use. For example, you can import the Python math module by writing the following at the top of your Python program file:

```
import math
```

Then, you can use any of the functions and variables defined in the math module as you write your code by calling the module name followed by a full stop and the function or variable name. For example, your program file might look like this:

```
import math
```

```
print("pi: ", round(math.pi, 2))  
print("3!: ", math.factorial(3))
```

which will output the following when it is run:

```
pi: 3.14
```

```
3!: 6
```

using the pi constant and factorial() function found in the math module. If you hadn't included the import math line at the top of your code, however, you would receive a NameError when you run the code. Just as you would receive an error if you tried to reference a variable you hadn't previously defined, math.pi and math.factorial() don't have any real meaning unless you reference the math module first.

Note: in the previous example, we used the math module, which is installed automatically when you install Python. If you wish to use a module that is not already installed, you'll have to install it before you can import and utilize it within your code. You can install a module via the command line using pip like so:

```
pip install module_name
```

replacing module_name with the name of the module you want to install. You can find a list of Python modules in the official Python documentation at DOCS.PYTHON.ORG .

How do I create a function?

If the built in functions provided by Python aren't adequate for the program you're creating, you can opt to write your own functions that will perform whatever actions you decide. You can do this by starting a new function like so:

```
def new_function_name():
```

where new_function_name is whatever you'd like to call your function. This first line of your function must begin with the word def and end with a colon. You can also tell your function to accept values as parameters by writing one or more variables within the parentheses in this line, separated by commas. For example, you might want your function to accept two standard parameter values and a third optional value:

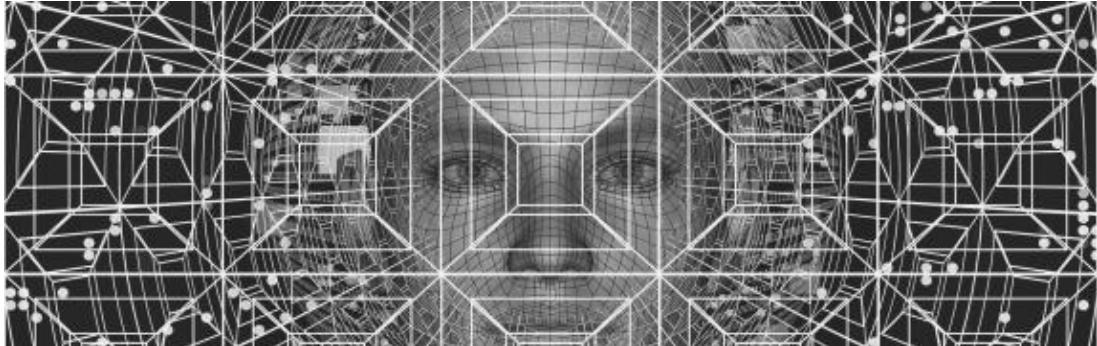
```
def new_function_name(x, y, z=0):
```

Chapter 11 Summary

- A parameter is simply a variable information you give a function to work with, like the length and width values.
- By incorporating functions into your code, you eliminate the need to rewrite redundant parts of your program.
- Just because a function doesn't accept parameters doesn't mean that the output will always need to be the same when that function is run.
- When using a mix of standard and default parameter values, it is important that they are in the proper order in the function prototype.
- Before you start writing your own functions, you should familiarize yourself with functions that are already available.

Chapter 12:

Walkthroughs



You know how to write a very simple counting program, so the next step is to learn a few others that are pretty easy. In this section, we'll walk through how to write a program that can determine whether a number is prime and one that lets you send texts and emails. You'll also learn how to draw with Turtle and how to start creating simple games.

Prime number program

As you probably know, a prime number has no other factors except one and itself. 2, 3, 5, and 7 are the first examples, but once you get into larger numbers, it becomes harder to know if it's prime or not. For this program, the user will be able to type in a number and find out if it's prime or not. We're going to make use of the if...else statement, loop, and break and continue. On Programiz, here's what that code looks like:

Remember, anything following a # is an in-line comment, and is not actually executed by Python. They are there to help you understand the code. The first actual code is simply assigning a variable. In this example, it's 407. When the program is being used, your user would input a number.

The next step is to have Python check if the input number is divisible by any number from 2 to the input number subtracted by 1. If a number pops up, we know the number is *not* prime. In the code, you told Python to tell the user the number is not a prime number and to show them why. For example, we know 24 is not a prime number because it's divisible by 2. If you ran the code with 24 as num, you get this output:

24 is not a prime number

2 times 12 is 24

Python also knows to stop hunting for numbers at this point because of **break**. It already knows the number is not prime, so there's no need to spend any more time searching. However, if the number is indivisible, Python knows to print that it is a prime number.

Additionally, any numbers less than or equal to 1 are not prime numbers. At the beginning, you set the perimeters if num> 1, then the code checks within the range and has two options - either the number is prime or it's not - *but* if that original if statement is not true, then Python will automatically print that the input number is not a prime number.

Sending texts

In this walkthrough, you'll learn how to send out SMS text messages from a Python app. SMS stands for "short message service." To add SMS capability your Python app, here's what you need:

Python

os module

pip and virtualenv

Open source Twilio Python helper library (at least version 6.0.0)

A Twilio account

Step 1: Get a Twilio account

For now, sign up for a free trial account with Twilio. This lets you send text messages to your own validated phone number; if you want to send SMS to *any* phone #, you'll need to upgrade your account. Once you've signed up, you'll get a free phone number that you can use to send outbound text messages.

Step 2: Install the virtual environment

To make this whole texting thing easier, your code is going to use a helper library, but first you need a `virtualenv` (virtual environment) for the library. If you haven't installed it yet, head over to the `virtualenv` website, then click on PyPI, which will direct you over to the installation page. Get the latest version. To activate it, go to your terminal and first create a directory, just to keep everything organized. We're assuming you have the `os` module installed, which gives you the ability to create, view, rename, and delete directories.

Python Directory is basically working with operating system directories. To work with Python Directories, we need to import `os` modules. The `os` module contains functions to create, view, delete, rename directories. You can use the `mkdir(name)` function to create a new directory. Matt Makai's code over at RealStackPython looked like this:

```
# the tilde "~" specifies the user's home directory, like /home/matt
cd ~
mkdir venvs
# specify the system python3 installation
python3 -m venv venvs/flaskproj
```

What does that `flaskproj` mean? In that article, Matt was making an app with Flask, but we aren't doing that, so you can name it `textproj` or something. Then you activate the `virtualenv` like so:

```
source ~/venvs/textproj/bin/activate
```

Step 3: Install Twilio Python helper library

Time to install the library. Make sure you're using the 6.0.0 version or higher. In your terminal, type in:

```
pip install twilio>=6.0.0
```

The library is now installed and ready to be used for your code!

Step 4: Getting the Twilio code

In your terminal, type the python command or create a new file named something like send_SMS.py. You need the account credentials from Twilio, so go to the console on the site and copy the Account SID and Authentication Token. First, you'll be importing the Twilio client, and then typing SID and token like so:

```
# we import the Twilio client from the dependency we just installed
from twilio.rest import Client
# the following line needs your Twilio Account SID and Auth Token
client = Client("ACxxxxxxxxxxxxxx", "zzzzzzzzzzzz")
```

As you can see from Matt's comments, the "x's" stand for the SID numbers, and the "z's" are the authentication token. Make sure to write in the code with quotes and commas, exactly as they appear above.

Step 5: Sending the text

The "to" number is the one you used to sign up for Twilio with, while the "from" number is the number Twilio gave you. The numbers here are just placeholders. In the body, you put your text message.

```
client.messages.create(to="+1922644157",
    from_="+17023351279",
    body="This is a test!")
```

To run the Python script, type in SEND_SMS.PY . If you're using the interpreter, just entering the code will run the program. Wait a few seconds to see if it worked!

Sending plain-text email

In this walkthrough, you'll learn how to send basic, plain-text emails using Python. There won't be attachments or HTML content, since adding that capability is a bit more complicated. Right now, let's keep things simple. Before getting started, Joska de Langen over at RealPython says it's a good idea to create a new gmail account where you can send all the code you're testing. If you use your regular one, the inbox will get full right away and you might accidentally expose your login details. Setting up a new google account is straightforward. Next, you want to toggle the "ON" for "Allow less secure apps."

Step 1: Securing an SMTP connection

Now, you want to establish a secure SMTP connection, which stands for simple mail transfer protocol. This ensures encryption, so your emails and logins aren't insecure and vulnerable. Joska describes two ways to do this, but we're only going to explain the first one, which is starting an SMTP connection that's secure right from the beginning. It's a good choice because it's concise. Here's what that code looks like in your editor:

```
import smtplib, ssl  
port = 465 # For SSL  
  
password = input ( "Type your password and press enter:  
")  
  
# Create a secure SSL context  
context = ssl.create_default_context()  
  
with smtplib.SMTP_SSL( "smtp.gmail.com" , port, context=context)  
as server:  
  
    server.login( "my@gmail.com" , password)  
  
    # TO_DO: Send email here
```

What is **smtplib** , **ssl**? It's Python's built-in module for encryption. When using this method, you connect to port 465. The above code gets you a secure connection with Gmail's SMTP server and uses the module to encrypt it with TLS (transport layer security) protocol, which basically just

ensures there's privacy between two communication applications. In this case, that's your code and your new gmail account.

Another note: when you're typing in the code, instead of writing my@gmail.com, use the gmail account address you just set up. Also, you'll notice the `input()` function, which prompts you or whoever is using the code to type in the password when running the program. It's safer than storing the password in the code itself. If you don't even like the idea of typing in your password and having it show up on the screen, you can import the `getpass` module, which lets you blind input (those little stars) your password.

Step 2: Sending the email

You have a connection, so now you can send emails. Here's Joska's full code:

```
import smtplib, ssl  
port = 465 # FOR SSL  
smtp_server = "smtp.gmail.com"  
sender_email = "my@gmail.com" # ENTER YOUR GMAIL ADDRESS  
receiver_email = "your@gmail.com" # ENTER ADDRESS OF PERSON YOU'RE  
SENDING EMAIL TO  
password = input ("Type your password and press enter: ")  
message = """  
Subject: Hi there  
  
Hello, this is my first email with Python.  
  
context = ssl.create_default_context()  
  
with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:  
    server.login(sender_email, password)  
    server.sendmail(sender_email, receiver_email, message)
```

As you can see, much of the code repeats what we have before, you're just filling bits around the connection. The actual sending of the email happens

in the last line, but above, we've defined what `sender_email`, `receiver_email`, and `message` are. This way, you can send as many emails as you want just by changing what's in green and leaving the rest of the code alone.

Drawing with Turtle

Way back in 1966, two guys created a programming language called Logo. They invented Turtle graphics along with Logo as a way to teach it to kids, who could learn to write code that allowed them to manipulate a turtle on the screen. Like a pen, the turtle would move around creating colored lines, shapes, and more. Because of how engaging Turtle is, you'll see it used for other languages. There's even a standalone Python install that's just Turtle. In our normal Python 3 we've installed, there's a `turtle.py` module built in, so that's what we're going to be using. Another note: We're summarizing a walkthrough from GitHub, where they use IDLE, so open that up first. Then, click on File > New File. We're ready to enter code.

Drawing a square

The easiest drawing we can make is a simple square. In IDLE, type in:

```
from turtle import *
```

```
t = turtle.Turtle()
```

```
t.FORWARD( 100 )
```

```
t.LEFT( 90 )
```

```
t.FORWARD( 100 )
```

```
t.LEFT( 90 )
```

```
t.FORWARD( 100 )
```

```
t.LEFT( 90 )
```

```
t.FORWARD( 100 )
```

Save the file, and then run the program. If you want to test it out online first, head over to the Trinket site. You will see a square being drawn, ending with an arrow pointing down into the square's left-hand corner. How did this happen?

We're going to refer to that arrow as a "turtle." In the original Logo Turtle graphics, it actually was a tiny turtle. It basically functions as a pen. Also, whenever you create a new file and want to draw, don't forget you need to import turtle first. It's automatically installed, but it needs to be imported when you're making a new program. You don't need to assign `t` to `TURTLE()`, you can name it whatever you want, like a person's name.

The forward, left, forward, etc are the instructions for the turtle. The reason you type "forward" is because the turtle starts out facing right. The number after "forward" is the number of steps, while the "90" after "left" is degrees. So, in normal English, the code above is telling the turtle to:

Move forward 100 steps.

Turn 90 degrees to the left.

Move forward 100 steps.

Turn 90 degrees to the left.

Move forward 100 steps.

Turn 90 degrees to the left.

Move forward 100 steps.

The "forward" and "left" are functions, and there are many more you can use, like "backward," "right," "circle," "dot," "stamp," and so on. The numbers that come after it set the parameters for that function. The number in a backward() or forward() function is always distance, while in left() or right(), it's an angle.

Adding color

Your pen doesn't have to be black. It can be any color you want using the RGB color system. RGB stands for red, green, and blue, because these are the three primary colors of light. By adjusting the intensities of these three primaries, you get all the colors. This is done using float values. Float value 0.0 in red, green, or blue means that none of that color shows up. 1.0 adds the full brightness. As an example, here's what pink looks like: (1.0, 0.0, 1.0). You can make any color you want with the proper RGB float values. This website <https://rgbcOLORcode.com> has the full spectrum available to you, and you can see the values for any of the colors, as well as their fancier names, like "AntiqueWhite" and "DarkSalmon."

When you're using the imported turtle mode, you don't have to use float values, you can just use the color name, like so:

You can also add and change colors using tuples. Remember tuples? They're enclosed in parentheses and can't be changed. Until you're comfortable with tuples, maybe stick with more simple designs, so you don't have to use them.

You can fill in shapes with color, too, using the `FILL()` function. You would first define what `T.FILLCOLOR` you want. Let's stick with a shade of pink, like fuschia. Then, you write in `T.BEGIN_FILL()`, so the code knows whatever shape is created next will be filled. Tell the turtle to move around how you want - we're going with a square again. When the square is completed, you write `T.END_FILL()`, and that square will be filled with the color you chose. We end up with this:

Your pen color, which becomes the shape's outline can have its own color. Before the `T.FILLCOLOR()` line, simply add `T.PENCOLOR()`.

The only limits on shape-drawing in Turtle is your imagination. You can create intricate snowflakes, hexagons, and more, all in a rainbow of colors, using loops and other coding shortcuts. A tool like Trinket is a great way to practice Turtle basics and get results right away.

Chapter 12 Summary

- Python knows when to stop hunting for numbers because of break.
- Python Directory is basically working with operating system directories.
- The only limits on shape-drawing in Turtle is your imagination.

Chapter 13:

Building A Complete Program



Open a new document in your text editor and name it **atm.py**

Let's start by welcoming the customer to our virtual bank machine.

```
print("Welcome to Sky Bank\n") # \n adds a new line
```

We will need to create a prompt for the user to enter his pin number. We will assign the user input to a variable called **pin** . (Don't forget to convert the text from the user keyboard to a number using "int" before the input function)

```
pin = int(input("Please enter your PIN number "))
```

We now need to create a **while loop** in case the user fails to enter his pin number correctly.

If you wished, you could create a **for loop** instead giving the user a set number of times to enter his pin. However, we will use a **while loop** :

```
while pin != 1234: #not equal to  
    pin = int(input("Incorrect, Enter your PIN number "))
```

When the user enters his correct pin; in this case 1234 he is given a menu with a choice to withdraw, lodge money or display the balance on the screen.

(We use the newline (\n) to display each menu item on a new line)

```
if pin == 1234: #is equal to  
    option = int(input("Select:\n 1 Withdrawal\n 2 Lodgement\n 3 Balance\n\n"))
```

If the user selects option 1: Withdrawal

We need to know how much he wishes to withdraw. We also confirm with the user that he wishes to withdraw that amount, in case he had made a mistake. To do this we create a new variable called **confirm** which displays text to the screen and takes the users input as a string. We will need to concatenate the string with the variable **withdraw** which contains an integer. To do this we convert **withdraw** to a string using str

```
if option == 1:  
    withdraw = int(input("Enter amount to withdraw "))
```

```
confirm = input("Confirm withdrawal of " +  
str(withdraw) + " Y or N ")
```

We need to create another **if statement** to handle the variable **confirm**. We will nest this statement by tabbing once. This **if statement** only becomes live if the user initially selects **option 1**, otherwise it will be ignored.

```
if confirm == "y":  
    print("Please take your money")  
else:  
    print("Transaction cancelled. Please take your card")
```

If the user instead selected option 2; Lodgement

Firstly, we prompt the user to lodge their money. We create a variable called **confirm_2** to handle this **if statement**, which we will nest.

```
elif option == 2:  
    print("Please insert your cash ")  
    confirm2 = input("Proceed with lodgement Y or N ")  
    if confirm2 == "y":  
        print("Your money has been lodged")  
    else:  
        print("Transaction cancelled. Take your card")
```

If instead the user selected option 3; view Balance:

If we were doing this for real, the balance would be read in from a user's account.

For this basic program we will just display a fictitious balance via a print statement to the screen.

```
if option == 3:  
    print("your balance is 214.12")
```

Once the balance is displayed to the screen the program finishes and automatically closes. But we want to keep our program open and have the user control when it closes...

To do this we create an arbitrary variable called **close**. Even though it seeks an input, we don't use it for anything other than print a prompt to the screen. It will continue to do so until we hit enter or in fact, any key at all.

The variable **close** is not indented so is not controlled by any of the **if statements**. Therefore, **close** will be triggered and display its prompt no matter where the program finishes.

```
close = input("Have a nice day. Press Enter to Exit")
```

Have a look at the complete program, especially taking note of how the **if statements** are nested.

```
print("Welcome to Sky Bank\n")  
pin = int(input("Please enter your PIN number "))  
while pin != 1234:  
    pin = int(input("Incorrect, Enter your PIN number "))  
if pin == 1234:      # main if statement  
    option   =   int(input("Select:\n    1    Withdrawal\n    2  
Lodgement\n    3 Balance\n\n"))
```

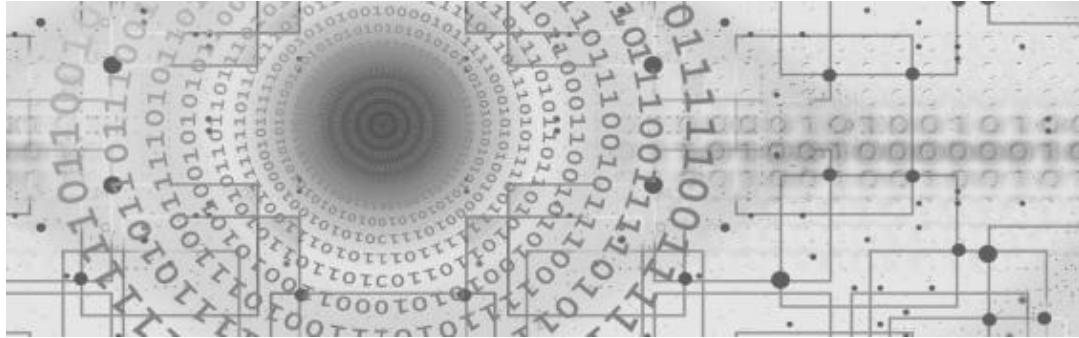
```
if option == 1:
    withdraw = int(input("Enter withdrawal amount "))
    confirm = input("Confirm " + str(withdraw) + " Y or
N ")
    if confirm == "y":
        print("Please take your money")
    else:
        print("Transaction cancelled. Please take
your card")
elif option == 2:
    print("Please insert your cash ")
    confirm_2 = input("Proceed with lodgement Y or N "
)
    if confirm_2 == "y":
        print("Money has been lodged")
    else:
        print("Transaction cancelled. Please take
your card")
if option == 3:
    print("your balance is 214.12")
close = input("Have a nice day. Press Enter to Exit")
```

Chapter 13 Summary

- Welcoming a customer: `print("Welcome to Sky Bank\n")` # \n adds a new line.
- We use the newline (\n) to display each menu item on a new line.
- For this basic program we will just display a fictitious balance via a print statement to the screen.

Chapter 14:

Directories



The `mkdir()` Method

Use this to make directories in your current directory. However, you have to supply the argument that has the name of the directory you want to create. The syntax of the `mkdir()` method is as follows:

```
os.mkdir("newdir")
```

The following example shows how to create the directory *harlequin* in your current directory:

```
#!/usr/bin/env python  
  
import os  
  
# This creates the directory harlequin.  
  
os.mkdir("harlequin")
```

The `getcwd()` Method

It displays your current working directory. Its syntax is as follows:

```
os.getcwd()
```

Take a look at the following example. It shows you how you can see your current directory:

```
#!/usr/bin/env python  
import os  
  
# This shows the location of your current directory.  
os.getcwd()
```

The chdir() Method

You can use this to change your current directory. It takes an argument -the directory name of the one you intend to be your current directory. Its syntax is as follows:

```
os.chdir("newdir")
```

Consider the following example. It shows you how to go into the */home/newdir* directory:

```
#!/usr/bin/env python  
import os  
  
# This changes the directory to /home/newdir.  
os.chdir("/home/newdir")
```

The rmdir() Method

It deletes your directory, and then passes that as an argument. Take note that before you remove your directory, make sure that you remove all of its contents. The syntax of the rmdir() method is as follows:

```
os.rmdir('dirname')
```

The following example shows you how to delete the */tmp/test* directory. You need to indicate the full name of the directory in order to prevent your current directory from being searched.

```
#!/usr/bin/env python  
import os
```

```
# This removes the /tmp/test directory.
```

```
os.rmdir("/tmp/test")
```

Utilities For Files and Directories

In Python, there are vital sources that provide a variety of tools for handling and manipulating files and directories on both the UNIX and Windows operating systems. These are the following:

- OS Object Methods. They provide the methods for processing files and directories.
- File Object Methods. The *file* object is the one that provides the functions for manipulating files.

Chapter 14 Summary

- Use The mkdir() Method to make directories in your current directory.
- You can use the chdir() Method to change your current directory.
- Tools for handling and manipulating files and directories on both the UNIX and Windows operating systems are OS Object Methods and File Object Methods.

Chapter 15:

Python - Dictionary



Python dictionary is a mutable key-value pairing feature. It is mutable because it can be changed. It pairs a key to a value, in each pairing. The keys in the Python dictionary are immutable, like numbers or strings and cannot be changed. They also have to be unique. The dictionary in Python can contain unordered items. In other programming languages this is referred to as a hash table or an associative array.

Creating a dictionary in Python can be done in more than one way. One method is using literal codification, where the key-values are contained in curly brackets and the pairs are separated by commas.

```
months = { "Jan": "January", "Feb": "February" }
```

The first value, "Jan" is the key, followed by a colon and then the value, in this instance the value is "January."

Python dictionaries can also be created using dict (), the dictionary function.

```
vals = dict (uno=1, dos=2)
```

Another method of creating a dictionary in Python is by first creating an empty dictionary, followed by one or more instances. In this case, 3 pairings are added.

```
clothing { }
```

```
clothing["hat"] = "bonnet"
```

```
clothing["shoes"] = "hush puppies"
```

```
clothing["jacket"] = "leather bomber"
```

The keys are assigned first, inside the square brackets. The values are on the right side of the equals sign. The dictionary is put together using a comprehension, which is defined with 2 parts. The first part, as shown, is known as the object. The second part is referred to as the, for i in range(4) loop. 4 pairs have now been created in the example above.

```
dic = { i: object() for i in range(4) }
```

Operations can be performed using dictionaries. Values can be added or removed from one or more dictionaries using specific functions. The pop function will remove pairs from the dictionary. It is also possible to create new dictionaries from lists using the fromkeys() function. Using the setdefault() function, if a key is available then a value is created. Dictionaries can be joined using the update() function.

The clear() function can remove all items from a dictionary.

Python - Date & Time

When you are planning to create date and time stamps, you need to remember that there are many modules for creating them. When you use these modules correctly, you can put the stamps anywhere you want. Also, you will be able to create time and date stamps that people can use themselves. Consider each way of using the module in your programming work.

A Clock

You can create a clock that fits on the screen easily. These clocks can carry the date and time, and you can populate them with a calendar that people can check for many years to come. Also, these clocks can be designed in a number of styles. Some of them look better in certain colors, and you can change the colors to create a special clock that is bespoke for your current project.

Stamps

You can create time and date stamps that will go on a number of applications. They can be used to mark word processing documents, and they can be used to mark messages in a messenger service. Also, these stamps can be used when people are editing documents.

Timers

When you want to create a timer, you will be able to offer people a stopwatch and timer. These timers can count down from any number you want, or you can create a stopwatch that counts down to the hundredth of a second. People use these timers to work, and they can use stopwatches to measure processes that they are working on.

The timers and stopwatches are wonderful for people who are timing athletic events, and they can be used for people who work on their computers. Also, these people need to make sure that they are setting up the clocks to work for athletics or work purposes.

The Clock Setup

The time and date module allows people to set up the clock to run on 12 hour or 24 hour timing. These times are easier to read for many people, and they allow people to read a clock that they are used to reading. This will help with European users, and it will help with people who prefer to read modern American military time.

The Python time and date module is going to help people create clocks and calendars that will work well on all the projects they work on. These

modules can provide a time and date for each item on the computer.

Chapter Summary

- Python dictionary is a mutable key-value pairing feature.
- Python dictionaries can also be created using dict (), the dictionary function.
- Operations can be performed using dictionaries.
- When you are planning to create date and time stamps, you need to remember that there are many modules for creating them.
- The Python time and date module is going to help people create clocks and calendars that will work well on all the projects they work on.

Chapter 16:

How To Use OOP



Now that you know what OOP is and how it works in accordance with your Python programming, it is important that you look at all of the concepts that come along with OOP and the way that they can be used.

In this chapter, you can expect to learn more about the way that OOP works and the way that you can make sure that you are using different options for your python experience. It will give you a chance to try new things and to add different elements to the programming that you created. It is also something that will give you the ability to make major changes to the way that things are done in your Python experience. Your programs will give you the chance to see how you can make the coding process easier and how you can include various options with the codes that you write.

Inherit

The “inheritance” factor is one of the concepts that you will learn when you are using OOP. It is the simplest and most widely used concept that you can write with on Python. It also gives you the chance to learn new things about

the programs that you are writing and gives you the ability to make major changes with the experience that you have on Python.

By looking at vehicles again, you can have a better understanding of how inheritance works.

All SUVS are vehicles. Not all of the vehicles that are available are going to be SUVS though. All vans are vehicles, and again, not all vehicles are vans. When you are looking at the different types of vehicles, you can clearly see that not all vans are SUVS and there are actually a lot more vehicles than just those two that you can use for different things.

The concept behind OOP is that things can be broken down into the smallest part possible, like this:

Vehicles -> SUVS -> Four Wheel Drive -> Class (seating) -> Make -> Model -> Trim

By looking at this, you can see that the vehicle is the actual parent that is used in the situation while the Trim becomes the child for it. There can only be one parent for a certain set of classifications, but there can be many children that the parent has.

If you were going to use the same sequence to be able to convert the information into your own Python codes and then be able to put the information out on your site or wherever you are using your programming for, it would look like this:

To make sure that you would have everything that you need in the code, you would actually need to write it:

By doing this, you will give yourself the chance to see that there are many different options that can go into the codes, and that will give you the chance to see the different aspects of each of the codes that you have. You will need to make sure that you are doing what you can with the codes you are working with so that you will be able to do more with it and so that you can include everything that is necessary in the codes that you are using with your Python.

You can test it all out and use the exact code that is included there to be able to bring it up on the program that you have created with Python.

Polymorph

With the same idea behind inheritance, polymorphism allows for the changes to be made to the different aspects that are included with the fares on all of the different changes that you are making. When you are able to use polymorphism, you will make sure that you are getting the best experience possible and that you can make some changes without ever having to go into the code. If you are going to use it for Python, you will need to use method overloading or overriding.

When you are able to use overloading to change the way that coding appears, you will need to write it so that there is a lot of code in one area. This will “overload” the system and will cause it to stop with the changes that you want to make. There will be too much for it to process, and this is something that can completely change the system. Either it will not pick up on the information that you have put into the code, or it will cause it to become completely deleted within the way that it is done.

If you are going to use overriding, you will need to write a code that is capable of overriding the entire system. Just because you need to get rid of one instance of the code does not mean that you will be able to choose the area that you want to get deleted. It is important that you try to make sure that you are using what you can to be able to make those changes and that each of the changes that you make will depend on the entire system. After you have overridden the system, you need to add additional information to it and put it into play so that you can change the way it functions.

Now that you are aware of all of the changes that you can make after you have written your code, you will be able to make the changes that you want. It won’t necessarily mean that you are going to have those changes, but it will give you a chance to try more and to do more with the areas that you are using.

You should feel less pressure when putting your code together and trying the different files out since you now know that it is all able to be fixed through either polymorphism or inheritance.

Chapter 16 Summary

- The inheritance factor is one of the concepts that you will learn when using OOP.
- The concept behind OOP is that things can be broken down into the smallest part possible.
- Polymorphism allows for the changes to be made to the different aspects that are included with the fares on all of the different changes that you are making.

Conclusion

Thanks for making it through to the end, let's hope it was informative and able to provide you with all of the tools you need to achieve your goals whatever it may be.

The next step is to go on to more complicated topics. You've started the long and often arduous journey of programming in this book. And the best thing about it? There's no finite end point. There's never going to be a point in programming where you say enough is enough, or where you reach some kind of "peak" in your knowledge. Well, technically speaking, maybe, but only if you quit trying will you have hit a peak.

Programming is one of the most liberating tasks known to man, because it's the ultimate art form. It's the most interactive art form too. When you program, what you're doing is literally talking to the computer, and thereby making the computer talk to the user. Every single program you make is an extension of the effort that you put into it and the time and the code that you've dedicated to it.

Programming, too, is not easy. In fact, it's rather difficult. And there are topics that are sadly too esoteric to cover in this book. For example, we didn't get to the bulk of file operations, nor did we get to things like object-oriented programming. But I hope what I've given you is a very solid foundational understanding of Python so that you can better service yourself to learn about these things.

I told you I wasn't going to hold your hand, and I didn't. And now I won't, either. But what I can say is that you have worked hard to be a programmer and you have worked hard throughout the course of this book, likewise. Not all of the concepts within this book are easy to understand, even with more in-depth explanations.

My goal here wasn't explicitly to teach you Python or object-oriented programming or any of that: my goal was to teach you the computer. The

way it thinks, and the way programs are written. Anybody can learn Python keywords. But to learn to program, and to write solid effective code regardless of which programming language that you're using, that's another skill entirely.

I sincerely hope that this book has helped you to get on the road to accomplishing everything that you want to accomplish in the world of programming. Remember going forward that without a doubt, it will not always be easy. It won't be even remotely easy. Programming is made difficult by its very nature. Humans, we just don't think like computers. But hopefully I've helped you to understand programming at least.

Book 2:

Neural Networks for Beginners

*An Easy Textbook for Machine Learning
Fundamentals to Guide You Implementing
Neural Networks with Python and Deep
Learning*

To Enza and Angelomaria

Introduction

We are all aware that computers are better than humans at analyzing a series of numbers. However, what about the tasks which are more complex? How would you teach a computer what a cat seems to like or how to play a complex strategy game or even how to drive a car? Can it make predictions regarding the stock market? These are some of the more difficult tasks involving artificial intelligence and they are far more complex for the capabilities of machine learning and artificial intelligence. In these cases, the scientists go to neural networks for help.

The thing that sets neural networks to one side from the other ML algorithms in a wide range of industries and disciplines. Let's cut through the buzzwords and look at what the neural networks actually are and how they are different from other machine learning algorithms and how they are applied in modern scenarios.

This concept of underpinning a neural network has been here for many decades. However, during recent times its computing capability has caught up with the requirement. Distributed systems such as Hadoop's MapReduce paradigm clearly means that there is no need for a supercomputer to handle large computations. Neural networks need you to spread the work item across the clusters of cheap hardware.

There have been many reports in the media suggesting artificial neural networks working like a human brain but in reality they are being too

simplistic. For one there is a massive difference in scale. Although the neural networks have increased a great deal in size they still consist of a few million neurons. This pales into insignificance compared to the huge 85 billion neurons present in any normal human brain.

The other major difference in the two lies in how the neurons are connected. Inside a human brain all the neurons are connected to several neurons close by. In a typical network the information flows only in one direction. In neural networks the information is spread across 3 layers:

- *Input Layer*: The input layer contains neurons that only perform the function of receiving and passing on the data. The number of neurons inside the input layer will be equal to the features in the data set.
- *Outside Layer*: The outside layer contains a number of nodes which depends on the kind of model you are building. For a classification system there will be a single node for every type of label you are applying. Similarly there will be a single node which puts out a value for a regression system.
- *Hidden Layer*: Between these two layers the things start to get more interesting as there lies what is called as a hidden layer. It contains a number of neurons. How many neurons it contains will depend on the number present in the input and output layers. These nodes in the hidden layer will apply transformers to the input before passing it on. As this network gets training these nodes become more predictive and accurate and the output begins to get more weight.

Training the Models

One of the ways of thinking of neural networks is by imagining a black box having several knobs on its side. Yann LeCun who is a pioneer of neural networks describes them this way. Let's take the example of a neural network in which we are trying to train the network to predict whether a picture is that of a cat or not. This is a real-life example. Training a model involves fiddling with these knobs until the output layer can accurately identify a cat's picture. It goes without saying that when many knobs are involved it is not possible to turn them all manually. This is where the sophisticated machine learning algorithm comes in. The algorithms adjust

these knobs automatically until the model fits the input data. This means they are adjusting the weighting functions of different neurons in the hidden layers.

Another important thing to take into consideration is, similar to this example we are adjusting the knobs until we have one great dog detector or we could tweak them a little until there is a great cat detector. Or we can re-adjust the knobs until we have one great big submarine detector. The point of the discussion is that the structures are generalized meaning that the same fundamental structures can be trained to reply to a number of queries. This is the reason why neural networks are more powerful. The key to the whole thing is in what machine learning algorithms and weighing functions we employ.

Skills & Tools Required

Neural networks add a cutting edge to the machine language technology and also artificial intelligence. Implementing this technology needs expertise in statistical analysis, large data processing, distributed systems, and other related fields. Luckily there are a lot of libraries available which make designing and implementing the neural networks relatively simple. Here are the more popular options:

TensorFlow

It is a high profile entry in the field of machine learning and is developed by Google as an open-source successor to DistBelief which was their former framework for training the neural network. This framework makes use of a system of nodes which is multi-layered to allow you to set up quickly, train, and install artificial neural networks by using huge datasets. This allows Google to find objects in photographs or understand the words in their voice recognition app.

Scikit-learn

It builds on the foundational Python libraries such as SciPy and NumPy by adding a set of algorithms required for the common data mining and

machine learning tasks. These include support for supervised and unsupervised neural networks both. Scikit-learn as a library has a lot of things going for it. It has tools which are well documented and its contributions are prepared by several ML experts. Significantly it is a curated library meaning, the developers don't have to select between various versions of the algorithm. The ease of use and power of the option makes it popular among many data-heavy startups including OKCupid, Birchbox, Spotify, and EverNote.

Theano

This is a machine learning library from Python that makes use of syntax similar to that of NumPy to evaluate and optimize mathematical expressions. What sets it apart is the use of the machine GPU to make data-intensive calculations that are 100x quicker than the CPU. The speed of Theano makes it extremely valuable for deep learning other complex computational tasks.

DeepLearning4j

This is a Java-based library used for implementing neural networks that are widely used for anomaly detection, recommender systems, and image recognition. It comes with APIs which allows the software to be used with the more data-oriented languages such as Python, Clojure, and Scala.

Historical Background

Although neural networks were developed before the invention of computers, they suffered a setback since people were not fully adept at using these networks. It is due to this reason that the simulation of neural networks is a recent development. Improvements made to computers have boosted many advances in the development of neural networks. People threw themselves into research when the concept began to gain importance. They were unable to obtain any information regarding how they could use these networks to improve the accuracy and efficiency of machines, which led to a dip in enthusiasm. There were some researchers who continued to

study neural networks and worked hard to develop technology that people in the industry would accept.

Warren McCulloch and Walter Pitts developed and produced the first artificial neural network in 1943. This neuron is known as the McCulloch-Pitts neuron. Since the technology to develop the network further was not available to them in that year, they were unable to use the neural network to perform complex tasks.

Why Use Neural Networks?

Every industry today utilizes large volumes of data to improve functions. These data sets have many variables that make it difficult for human beings to understand the patterns within them; neural networks make it easier to identify these hidden patterns in the data set. Some computers also find it difficult to identify the trends in the data set if they do not have neural network architecture. An engineer can train the neural network using large data sets to help it become an expert in the field. They can then use the network to predict the output for any future input. This gives the engineer a chance to answer some important questions about the future.

Some advantages of neural networks include:

- They use a machine learning technique called supervised machine learning to adapt and learn new tasks.
- They can represent any information provided to it during the learning stage.
- Since neural networks can compute some data in parallel, machines with neural network architecture work faster to provide results.
- If a neural network is partially damaged, it can lead to a dip in performance; however, the network can retain some of its properties even when damaged.

Neural Networks vs. Conventional Computers

Conventional computers and neural networks do not both use the same approach to solve any problem. The former only use algorithms to solve a given problem. Conventional computers can also solve problems if they know what steps to follow to identify the solution. This means that conventional computers often solve problems that human beings can solve, though computers are definitely more useful when they can solve problems that human beings cannot solve.

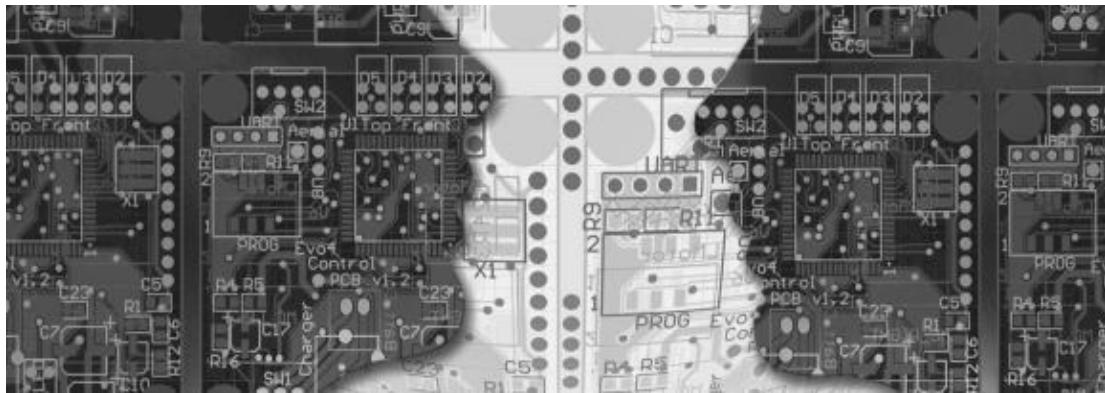
The neurons in the network are interconnected. The neural networks always work together to solve most problems. Since the network learns by example, the engineer cannot teach the network to perform a specific task. This makes it important for the engineer to choose the training data sets carefully. Otherwise, it's difficult for the machine to learn how it should solve a problem correctly. Since the neural network learns from the data set, it can solve a variety of problems, including those that the engineer does not train the machine to solve. This makes them quite unpredictable.

Computers also solve problems using cognitive approaches. The computer should know the process it needs to follow in order to solve the problem, and this will only work when the engineer provides the machine with the correct instructions, which can be provided to it using a high-level programming language. The computer will then decode the instructions into a language that it can understand. This process helps the engineer predict how a machine will work. If there's an issue with the result, the engineer can be certain that it's due to a hardware or software problem.

Neural networks and conventional computers complement each other. Some tasks like arithmetic calculations are better suited for conventional algorithmic computers, while various complex tasks are more suited for neural networks; however, most require a combination of these approaches to ensure that the machine performs at maximum efficiency.

Chapter 1:

Types of Neural Networks



Convolutional neural networks

For humans, image recognition is done without thought. It happens almost instantly. We see a photo of our grandmother and we don't have to analyze it to determine who it is, we just know. Every person can identify a bird when they see one. In our brains, this process is second nature but teaching a computer to do it is no easy task.

Computers can do this now with the use of a convolutional neural network. This is a neural network that uses identical replicas of the same neuron. This makes it possible for the network to learn the characteristics of a single neuron and then use it in a variety of different places.

The architecture of a convolutional neural network is based on the biological process that happens when a person identifies something with their sight. In the human eye, individual cortical neurons respond to

stimulation from light as it enters the person's field of vision. This is known as the 'receptive field', which is almost entirely covered by neurons. In a computer, this process is accomplished with convolutional neural networks (CNN).

These are a special class of networks that are enhanced to be extremely powerful in selected fields like image recognition and classification. They can identify objects, signs, and are the primary tool used in modern inventions like self-driving automobiles.

Each CNN has several sub-sampling layers as well as a number of additional layers that are all interconnected. When data input is gathered it is based on a very specific formula that tells the computer the height and the width of the image as well as the number of colors and any other characteristics that will allow the computer to make an identification.

To create a CNN, the computer must be programmed to recognize an object. Any object. Just like with other types of neural networks, computers can solve a number of different problems by linking a series of neurons together. This way, all the neurons work together to calculate the relevant features of the problem. With a CNN, you can use the same algorithms but adapt them to different types of problems. By adapting the algorithm to recognize an object rather than coming up with a numerical calculation you can simplify the task.

To that end, researchers have already created a dataset of handwritten numbers called the MNIST, which contains more than 60,000 images of handwritten figures. However, computers only recognize numerical values not images, so you will have to transform those images into a matrix of numbers, which symbolizes its different characteristics.

To input this image into a neural network it needs to be transformed from a visual image to a matrix that consists of more than 300 numbers. In order to manage all of these inputs, the neural network must be expanded so that it has at least 2 outputs instead of 1. The first output will calculate the probability rate that the picture is a given number say 7 while the other output will do the opposite and calculate the probability that the image is not. The system will then categorize all the images in the database with a separate output for each group of items it will have to identify.

While the neural network is not capable of matching the recognition skills of the human eye, it can recognize objects when properly programmed to do so. This type of function can be very useful in a wide range of industries that need recognition skills in areas where humans cannot function at an extremely rapid pace.

Perceptron and backpropagation

The most basic and simplest form of a neural network is the perceptron. It has a simple binary function that can produce only two possible results. If the solution it produces is positive the function will produce a 1 but if it is negative the result will be a 0.

This type of network is designed for a very specific class of problems, such as the identification of objects. A single perceptron can identify a dividing line and can tell whether any given point is located above or below that line.

Multilayer Perceptron

Multilayer perceptrons are often used to analyze a large amount of data that exists in table format. They can decipher rows and columns and use them as variables. This system works best when the rows and columns are interchangeable. They can switch them around without risk of changing the meaning or the value of the data.

In an MLP, the input for the classifier is first changed by using a non-linear transformation formula. This places the input data into an area where it can then be separated linearly. This is usually done in one of the hidden layers of the network.

To train an MLP, parameters must be set. These parameters must represent the qualities of the classifications that need to be made. The good news is that in the last few decades, guidelines have been created that make selecting these parameters much simpler. Several of these guidelines already exist and can be found in programs like the Efficient BackProp.

Chapter 1 Summary

- In the human eye, individual cortical neurons respond to stimulation from light as it enters the person's field of vision.
- To create a CNN, the computer must be programmed to recognize an object.
- Neural networks can recognize objects when properly programmed to do so.
- The most basic and simplest form of a neural network is the perceptron.
- Multilayer perceptrons are often used to analyze a large amount of data that exists in table format.

Chapter 2: **Components of a Neural Network**



The final thing that we are going to take a look at in this guidebook is the idea of neural networks and how they will work with Python, and with machine learning. Neural networks are going to be one of the frameworks that you can do in machine learning, one that tries to mimic the way that the natural biological neural networks in humans are going to operate.

The human mind is pretty amazing. It is able to look at a lot of different things around us and identify patterns with a very high degree of accuracy. Any time that you go on a drive and see a cow, for example, you will recognize it as a cow. This applies to any animal or other things that you are going to see when you are out on the road. The reason that we are able to do this is that we have learned over a period of time how these items look, and what can differentiate one item from another.

The goal of an artificial neural network is to be able to mimic, as much as possible, the way that the human brain is able to work. It is a complex architecture to make this happen. But when it is in place, it allows the system to do a lot of amazing things that it would not be able to do in other

situations. Let's take a look at how these neural networks can work, and why they are so amazing for our work with machine learning.

A neuron is going to be made up of the body of a cell, with a few extensions that come from it. The majority of these extensions are going to be in the form of a branch that is known as a dendrite. A long process, or a branching, is going to exist and this part is known as the axon. The transmission of signals begin at a region in this axon, and that is going to be known as the hillock.

The neuron is going to have a boundary that we like to call the cell membrane. A potential difference is going to exist between the outside and the inside of this cell membrane, and that difference is called the membrane. If you are able to get the input to be big enough, then some action potential is going to be generated. This action potential will then travel all the way down the axon and head away from the body of the cell.

A neuron is going to come next. This neuron is connected up with another neuron, and on down the line, with the help of a synapse. The information is going to head out of the neuron through the axon and then it is passed on to the synapses and to the neuron which is going to receive the message. Note that the neuron is only going to fire once the threshold has gone above the amount that is specified. The signals in this process are going to be important because they are going to be received by the other neurons in the chain. The neurons are going to use signals to help them communicate with one another.

When it comes to the synapses, they can either be excitatory or inhibitory. When a spike or a signal arrives in one of the excitatory synapses, the receiving neuron is going to be caught on fire. If the signals are going to be inhibitory, then the neuron is not going to be fired onward.

The synapses and the cell body are going to be able to work together to calculate the differences that are there between the excitatory inputs and the inhibitory inputs. If there is a big difference here, then the neurons are told to fire the message on down the line.

These types of networks are going to be used a lot because they are great at learning and analyzing patterns by looking at it in several different layers. Each layer that it goes through will spend its time seeing if there is a pattern that is inside the image. If the neural network does find a new pattern, it is

going to activate the process to help the next layer start. This process will continue going on and on until all the layers in that algorithm are created and the program is able to predict what is in the image.

Now, there are going to be several things that are going to happen from this point. If the algorithm went through all the layers and then was able to use that information to make an accurate prediction, the neurons are going to become stronger. This results in a good association between the patterns and the object and the system will be more efficient at doing this the next time you use the program.

This may seem a bit complicated so let's take a look at how these neural networks will work together. Let's say that you are trying to create a program that can take the input of a picture and then recognize that there is a car in that picture. It will be able to do this based on the features that are in the car, including the color of the car, the number on the license plate, and even more.

When you are working with some of the conventional coding methods that are available, this process can be really difficult to do. You will find that the neural network system can make this a really easy system to work with.

For the algorithm to work, you would need to provide the system with an image of the car. The neural network would then be able to look over the picture. It would start with the first layer, which would be the outside edges of the car. Then it would go through a number of other layers that help the neural network understand if there were any unique characteristics that are present in the picture that outlines that it is a car. If the program is good at doing the job, it is going to get better at finding some of the smallest details of the car, including things like its windows and even wheel patterns.

Now, when you are working on this, you may notice that there could potentially be a ton of layers that work with it. The more details and the more layers that you decide to find, the more accurate the prediction with the neural network is going to be. When there are more layers, there are more chances that the algorithm is going to be able to learn along the way. From then on, the algorithm is going to get better at making predictions and will be able to do it with more accuracy, and faster, from now one.

This is a good algorithm to use when you want to have it recognize pictures, or even with some of the facial recognition software that is out there. With

these, there is no possible way that you can input all of the possible information that is needed. So working with the neural networks, where the program is able to learn things along the way, can really make a difference in whether the program is going to work or not.

Backpropagation

For you to work on training a neural network to do a certain task at the right time, the units of each part need to be adjusted. This ensures that there is a reduction in the amount of error that happens between the target output and the actual output. What this means is that the derivatives of the weights need to be computed by the network. So, this means that the network has to be able to monitor the changes in error as the weights are decreased or increased based on the situation. The backpropagation algorithm is the one that you will choose to use to help you figure this part out.

If you plan on having the network units you work with becoming linear, then this is an easy enough algorithm to understand. With the linear options, the algorithm is going to be able to get the error derivative of the weights by determining the rate at which the error is changing as the unit's activity level is being changed.

When we are looking at the situation with the output units, the derivative of the error is going to be obtained when we can figure out the difference between the target output and the real output. To help us find any change rate in error for the hidden unit in a layer, all the weights between the hidden unit and the output unit which it has been connected to has to be determined ahead of time.

To help us with this part, we then need to go through and multiply the weights by error derivatives in the weights and then the product that is added together. The answer that you are able to get here is going to be equal to the change rate in error for your hidden unit.

Once you have this error change rate, you can then go through and calculate the error change rate for all of the other layers that you want to work with. The calculation that you get for these is going to be done from one layer to the next and in the opposite direction from where you would like the message to head through the network.

Chapter 2 Summary

- The goal of an artificial neural network is to mimic the way the human brain is able to work.
- The neuron is going to have a boundary that we call the cell membrane.
- For you to work on training a neural network to do a certain task at the right time, the units of each part need to be adjusted.

Chapter 3: **Advantages to Using a Neural Network**



The idea behind a neural network is that it will be able to function similar to how the human brain functions. As we have already discussed, this is something that will be able to help people have a better understanding of the way that things work and of the different approaches to learning that can be done with computer systems. With neural networks, you will be able to get a multitude of benefits that will provide you with a system to do many different automated tasks.

The benefits of neural networks are extensive and will be able to help you no matter what you are using the neural network for.

Decreased Time

The speed at which a neural network is able to get things done is much higher than what a human brain is able to accomplish. There are many more chances for the computer system to get things done than there are for people to do it. Because of this, the amount of time that it takes to complete a specific task is decreased to a very low amount.

The faster that things get done with a neural network, the better. A computer system that is set up with a neural network can work nearly twice as fast as someone who is a human doing the same amount of work.

Increased Production

Because of the faster time that can be accomplished with neural networks, the amount that is produced in that time is increased. The higher levels of production are great for people who have a lot of things and businesses who have a lot of tasks that need to be accomplished. When production increases, profits go up.

Even for small business owners, neural networks can mean a huge change in profits. These are great for people who want to make more money and for people who want to be able to get things done the right way.

Businesses that have switched over to a neural network way of processing things have seen a huge increase in the amount of production that they have. They are able to get more done, produced, and begin profiting from it in a shorter amount of time than what it would have taken them to do half of the work while using humans.

Technological Outlook

In a world that has a huge technological influence, it is important for businesses who want to stay on the right track to keep up with all of the technology that is available to them. Because of this, using a neural network can be a big help and can bring them into the 21st century while they are doing different things with their business.

It is important to note that neural networks are not the only way to bring technology into production but they are the most efficient way to do it. When business owners utilize neural networks for the things that they are doing, they will have a better chance at being able to make important changes to their business.

With neural networks, the technology that is used is able to make things much better and create a better environment for the people who are using them. When it comes to the way that neural networks work, most business owners and production companies are able to do the tasks that they previously relied on humans for.

One of the biggest benefits of neural networks is that they can produce the slow and often faulty work that was previously performed by people. Those who the neural networks are replacing will then be able to control the network, but at a much faster speed than what they were able to work at in the past.

Additional Locations

Since neural networks function over a network instead of in a physical location, they are able to be controlled from nearly anywhere. As long as someone has access to that network, they will be able to control the network. It is something that will make it much easier for even the most advanced businesses to get things done. When people are using neural networks, they don't have to worry about where they are doing things – they just have to worry about getting them done from that location.

The neural network that businesses run depends on the different things that are going on with the business and the production. Now that you are aware of the different types of neural networks and what they are capable of, you can make sure that you are choosing the right one for your business.

No matter where your business is located or what you want your neural network to do, there is a chance that you will be able to get the neural network that you want. It will give you an idea of what can be done and the different things that the neural network can do for you.

Higher Level of Accuracy

Humans make mistakes. Perhaps it is because of their intuitive brains or because they have so much self-awareness that they can make their own choices. Free will can always dominate what humans do and they will have a different chance to make a mistake. Because they are working with different things that are related to technological advances, these are the problems that come along with humans and using them for these tasks.

Computers, though, do not make mistakes. Your network is not going to make a mistake for the functions that you are using. Whether you are using the network to help you get the trading options that you want or you are using it to produce the latest product that your company has developed, it will not make a mistake in producing them. The problems that are seen with humans will not happen with neural networks.

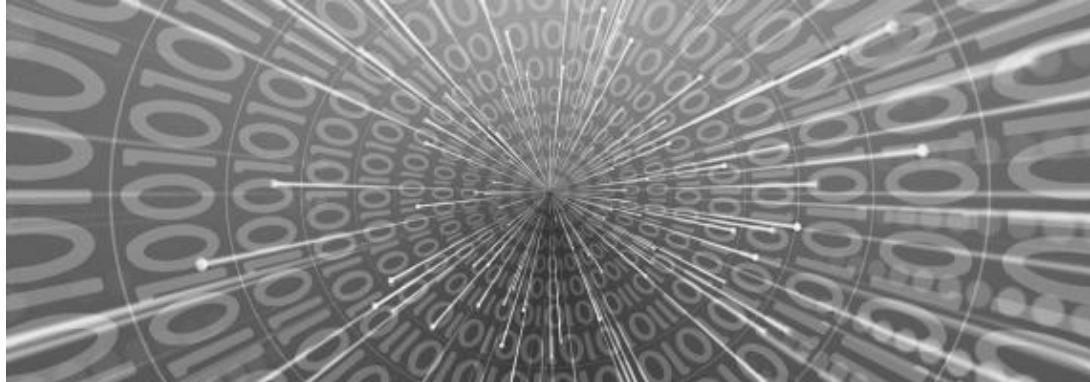
Businesses and individuals who choose to use the neural networks for their functions will see a significant decrease in the mistakes that are made – neural networks should prevent nearly every mistake that could happen with automated responses.

Chapter 3 Summary

- The idea behind a neural network is to function similar to the human brain.
- The speed at which a neural network is able to get things done is much higher than that of a human brain.
- Businesses using a neural network have seen a huge increase in production.
- Neural networks function over a network instead of in a physical location, hence they are able to be controlled from nearly anywhere.
- Neural networks should prevent nearly every mistake that could happen with automated responses.

Chapter 4:

Downsides of Neural Network Computing



It is very simple to see that neural networks can be beneficial to the world but what happens when there are different problems that cannot be solved as a result of the neural networks that are created for their purposes? What happens when the network misfires just like the nerves do with your own body? Neural networks are not completely foolproof but they can be beneficial to the way that you do different things and the way that you are able to change the different processes.

Are the benefits that come with neural networks enough to outweigh the problems that could happen with them? Here, we'll break down the various disadvantages that come along with neural networks in the different areas that they may be used in.

System Overload

The system that the neural networks are a part of will always have the chance of being overloaded with the different problems that come along with it. There is always the chance that it could become as overloaded as what it was in the past. It is important to make sure that you are doing things the right way so that it doesn't happen but there is always the possibility that it could happen.

When the system gets overloaded and if there are major problems with it, there is a possibility that the whole neural network could crash. This would not only be detrimental because there would be a loss of all of the different outputs that have come from it but there will also have been a loss from all of the time that was spent building up the neural network.

“Misfires”

Similar to how the mind is able to misfire, there is always a chance that the neural network could misfire. While it is not as likely to happen with the computer system as it is to happen with a human, there is still a chance that it could happen. If you look at all of the problems that are associated with nerve misfires, you can see that it would also cause problems with the artificial neural network.

The biggest problem with misfires is that, since neural networks put out so much data after it has been processed, it is almost impossible to find the problem with the misfire until it has made a very catastrophic mistake. It will have to have many different problems that are associated with it long before it is able to be caught. Most of the time, misfires are small and are of no real concern to the way that things are done with the data input but they are sometimes bigger problems that need to be taken care of.

A computer does not make mistakes but it can have instances where it misfires.

Technology Concerns

Right now, technology is booming. It has been for years and it is expected to continue. But, what happens when technology is no longer prominent and there is not an infrastructure to uphold all of the technology that is available today?

People who have relied on technological advances will have nothing to lean on and will actually have to go back to the way that things are going. When technology is no longer supported, there is a chance that people who are using neural networks will have to rethink their strategies and could end up losing time (read: capital) because of that.

While this is not something that you necessarily need to be prepared for right now, it is important to think about it in the fact that you could have problems in the future that would come as a result of technology that is currently used for neural networks.

Increased Energy Usage

A human does not need to be plugged in to be able to function. They do not require wireless Internet and they certainly don't waste huge amounts of energy just from being in the "on" mode. The neural networks, however, do. Whether you are running the neural network from a remote location or you are doing it in the same physical location that you are in, you will still need to use energy to be able to use the neural network. The energy usage that is used in most neural networks is negligible but when there are thousands of neural networks being run at one time, you will start to see an increase in your usage of energy.

While there are plenty of ways that you can solve this problem, it is one more thing that you will need to do. For example, you can simply switch to a "greener" method of running the networks or you can try to find a more sustainable source of energy to be able to run the network from. If you are doing all of this, it can have a negative impact on the energy that you use but it will be far less than if you were using traditional energy sources.

Consider the benefits that come with the neural networks over the energy usage problem.

Possibility of Hacking

Anytime that you do anything online or “over the waves,” you have the possibility of being hacked. This does not necessarily mean that someone is going to “come” through your neural network and find out all of your credit card information to use to make things harder for you. It actually means that someone who is not authorized to do so will actually come into your neural network and find out information that they are not entitled to. They can figure out the way that you produce things, the way that you are able to do different things and even the way that your neural network functions.

Because neural networks are not necessarily proprietary, you cannot prevent this from happening if someone gets into your network. The only thing that you can do is setup safeguards that will prevent people from getting into your network. This will help to safeguard it but there is nothing that you can do that will 100% reduce all of your chances of being hacked by the people who want to access your neural network.

Chapter 4 Summary

- There is always a chance that the neural network could misfire.
- A computer does not make mistakes but it can have instances where it misfires.
- Anytime that you do anything online you have the possibility of being hacked.
- Setup safeguards that will prevent people from getting into your network

Chapter 5:

Profiting from Neural Networks



Neural networks are great if you want to be able to create different things that are automated within a computer or a process. They are even better, though, when you are able to take the time that you need and start to profit from them. Profiting from your neural networks is so much more than simply being able to add the different options to your computers. You can now use neural networks to be able to make huge profits within the stock markets and even forex.

Now that neural networks are becoming more popular and they are able to be trained to be self-taught, you can use them for more things than what you were ever able to do in the past. Your neural networks need to be adjusted to be able to reach different levels and you should make sure that you are

going to be able to include all of the different money-making skills with the information you learn about them.

When you know how to make a neural network, you have more than just a skill – you have a profitable skill that you can use to build capital and create financial freedom for yourself.

Computers over Brains

Human brains are great because they are intuitive but neural networks are able to become nearly as intuitive as the brain. When they are trained to be self-taught in the right way, they will be able to make the best decisions when it comes to the options that are included with neural networks.

In the past, humans were the preferred method that were traditionally used for trading. While some computers were able to help trade on an intuitive level, they were not able to pick up on the same things that humans can.

With neural networks, people do not really need to be involved in the trading process.

Specific Money-Making Definition

The ability of a neural network to make money is completely dependent on its ability to be able to learn the different things that the administrator has taught the network to be able to do. It is important to make sure that you are getting the best experience possible with your neural network by including everything that you want it to do. If you set up your network to trade and learn how to interact with the various aspects of the stimuli that are going into it with the trading sector, you will be able to make money from it.

Training Through Data

The best way to get your network to make money when you are doing different things with it is to push various types of data through it. This will allow the computer to essentially “learn” the data and learn how to interact with it. It is a training process that the neural network needs to be able to go through in different areas that it is in. You should make sure that you are

doing what you can to be able to get the neural network adjusted to your trading abilities.

Rules Created for Networks

When you are working with the neural network and you are pushing the data through, you will need to set up some rules at the beginning of the trading process. This will mean that you will need to pay close attention to the different things that it is doing and the way that different types of stimuli are pushing through it. While you are watching it, you should change the rules that you set for your neural network.

Working them for Monetary Gain

As you are learning more about neural networks and the different things that go on with it, you will learn the way that you can make money from them. Money can be had in many different ways through neural networks. The most popular ways are through creating them for other entities and through trading with them. You can also invest in neural networks but the return on that takes quite a long time and is often not worth the time that you have to wait to see on it.

Trading System

There is an entire trading system that is associated with neural networks. People can buy, trade, sell and move around the different networks that they have. The different entities that are a part of the neural network trading field will help to determine the amount that is included with the trading. When you are working in neural networks, it is worth looking into the trade field that comes along with it – some people will pay a lot of money for the creation of these networks while others will pay to trade the different aspects of the network that they have created. While the trading plays a huge role in the buildup of capital with your neural network, it will also help to create a sense of community within the network.

Investing Through Networks

The investment that you make into a neural network while building it is usually made up of your time. There are generally not very high costs that are associated with neural networks but you should always be prepared for the costs that you might incur while you are using the neural network. You can, instead, invest in the actual neural network business. You can do this if you want to get returns on your trading and if you are going to be able to increase the amount of money that you have for neural investing.

Chapter 5 Summary

- Neural networks are becoming more and more popular and they are able to be trained to be self-taught.
- Neural networks are able to become nearly as intuitive as the brain.
- Neural network needs training to be able to go through different areas that it is in.

Chapter 6:

Neural Network Architecture Search



“**N**eural Architecture Search: A Survey” focuses on building such a neural network that can evolve on its own, create offspring neural networks and find the most optimal neural network layout for any given problem. The neural architecture search (NAS) is in essence automated machine learning that involves three parameters: search space, search strategy, and performance estimation. The only human involvement is setting these three parameters and assessing the results, though the paper laments the fact that humans introduce their bias by limiting the neural network’s search space. Search space refers to the number of architecture models we’ll let the neural network assess and recombine to reach something genuinely new and better than any of them. A novel proposition when it comes to search space is the

introduction of “cells”, hand-crafted neural network components that can preserve the data multidimensionality or be made to reduce data to a vector. Cells can be transplanted to another neural network or stacked on top of one another for unparalleled performance gains, but the notion is to let the neural network combine them in arbitrary ways until there is significant progress.

Any number of search strategies can be employed by the neural network, though the performance of some of them declines as the network scales, making them unfeasible. For example, the reinforcement learning search strategy used by Zoph and Le in 2017 required 800 computer graphics cards running for three-four weeks. However, evolutionary methods were used all the way back in the early 90s and allowed considerable search space exploration with low-key resource consumption. The main idea behind the evolutionary method is that the parent neural network runs for a while until it reaches a certain architecture, notes its efficiency on the task and then passes the layout on to its offspring, a brand-new neural network that now has some idea on how well certain cells perform on the task and can weight them accordingly. By gradually weeding out certain cells and their layouts the offspring can eventually reach the ultimate neural network architecture for the given task. Now, all that’s left is performance estimation.

Despite cutting corners with cells and using the evolutionary method to increase scalability NAS still incurs significant resource drain when it’s time to test the proposed neural network architecture, prompting the authors to resort to *performance estimation* that is, in essence, lo-fi measurement. The trick is that the estimate can’t oversimplify or the results will be no good, so the authors resort to merely ranking resulting neural networks based on their performance, with the idea that we don’t need to know how good the best one is, only that it *is* the best one of the bunch.

The authors conclude that NAS fared well, but ranking performance is nonetheless difficult due to the lack of common benchmark standards in deep learning. Another remark is that NAS doesn’t really reveal why a certain architecture performs in a certain way and that understanding cell groupings (also called “motifs” in the paper) would provide insight into how neural networks work.

Noise recognition

The 2018 research paper “Noise Adaptive Speech Enhancement Using Domain Adversarial Training” looks at speech recognition neural networks trained using deep learning and their ability to handle audio sources with additional background noises not encountered during the training phase. Rather than compiling the list of every noise ever created, the paper suggests using domain adversarial training (DAT) to train two additional subroutines of the neural network: a discriminator that tries to determine if the noise is coming from the original audio source and a feature extractor that tries to produce the best noise to confuse the discriminator. By pitting a discriminator against a feature extractor over an arbitrarily long period of time, the DAT technique allows the evolution of a specialized discriminator module that can later be merged with the audio recognition neural network for 26-55% better performance in devices such as cochlear implants and speech-to-text software. This kind of adversarial training is typical for neural networks as it allows scientists to leverage the blazing speed of computers for efficient learning compared to what would happen if they were manually fed examples.

Scene recognition

“From Volcano to Toyshop: Adaptive Discriminative Region Discovery for Scene Recognition” looks at an image classifying neural network that is first trained to label objects in the scene, then label the scenery itself, and finally contextualize both to arrive at a description of the location, for example, “art school” or “campsite”. The idea is to train the neural network to recognize certain objects as highly deterministic of the location, for example, finding a tent in the image strongly suggests it’s outdoors. Such neural networks already exist but are computationally demanding both in training and in operation; deep learning allows the operator to set an arbitrary number of signifiers in the image to achieve scalability.

Decrypting hidden messages

Steganography is intentionally hiding information in other data, and steganalysis is a way to unveil such hidden messages, both of which neural networks do much better than humans as revealed in “Invisible Steganography via Generative Adversarial Network”. The idea is to train two separate neural networks, one in hiding the information and the other in unveiling it, by pitting them against one another. The test given was to have one neural network analyze the cover image, find the most suitable pixels, successfully hide a gray image into a color one, and then send it to the other neural network for analysis. When working in tandem the two networks allow, for example, the headquarters to transmit a secret message encoded in a plain image through public channels to operatives in the field who can decode it using the other part of the duo.

Automatic question generation

High school quizzes taught us many facts, such as that mitochondria is the powerhouse of the cell. Preparing the quizzes required a lot of delicate but hasty work that could go wrong in any number of places, making students feel cheated out of a good grade due to a malformed question. With the help of deep learning, we might be on the brink of an age where questions and answers are unmistakably created from swaths of text by a neural network, the powerhouse of the learning process.

“Improving Neural Question Generation using Answer Separation” looks at automatically creating questions and answers out of any amount of text, ranging from single sentences to large paragraphs. The goal isn’t to have the neural network create a rough draft of the quiz but an actual workable version from scratch that doesn’t need any proofreading or editing. This is done by having the neural network identify and mask the answer with a token signifier and semantically conclude the correct word sequence and pronouns.

For example, the sentence “John Francis O’Hara was elected president of Notre Dame in 1934” contains three questions: who (John), what (president) and when (1934). By masking either of the three answers, we teach the neural network how to pose a question and test it against the

masked answer; by adding an attention mechanism we give higher weight to keywords and information tidbits, mimicking the way humans parse questions and retain knowledge. This approach allows extraction of maximum value out of the same text but also acts as an anti-cheat measure during the quiz itself since students can no longer copy an answer from someone else.

The neural network was tested using 23,215 text samples originating from 536 text sources with about 100,000 questions and answers created manually for actual quizzes. The results showed that only 0.6% of questions created by this neural network erroneously revealed the entire answer and 9.5% gave a hint as to what the answer was, both being common weak points of neural networks dealing with creating quiz questions. “What”, “how” and “who” were the three most common pronouns that the neural network guessed correctly, though the average precision for other question types wasn’t so stellar; the authors attributed this to 55.4% of all questions having “what” and other pronouns not being nearly as represented in the training dataset.

3D visual reconstruction of 2D objects

Great painters know how to use shades and perspective to make the canvas a genuine window into another world. Even though on some level we know that this kind of painting is an illusion, our brain snaps the 3D picture together and presents it as real, filling in the blanks using what it knows about the outside world. It turns out neural networks are capable of something similar and can reconstruct the unseen side of an object based off of one of its 2D views.

“Deep Learned Full-3D Object Completion from Single View” is a joint USA-Italy research paper looking at how pixels can be turned into *voxels*, best described as volumetric pixels. The main goal of this study is to minimize the number of perspectives needed to complete a 3D object, possibly to be used with robots on a tight computational budget moving through an environment and interacting with real objects. Since the authors needed at least one view, they decided to go with that and ended up actually succeeding.

The neural network was trained using 5,000 model presets from CAD, a popular modeling program, each model having eight snapshots taken from different angles to arrive at 40,000 challenges. All models were at a 30x30x30 resolution, which presented a challenge when it came to preserving all the nuanced features of a model, but the neural network managed to achieve impressive results nonetheless, restoring 92% of the original model.

Rain streak removal from an image or video feed

Setting up a remotely accessible outdoors camera can sound like a fun experiment; that is until the rain falls and rain streaks make it seem like we're peeking through a white curtain. The actual cause of this effect is that raindrops have a high velocity while reflecting light, causing white streaks to appear on any image capturing device. This devalues all other machine learning processes that depend on image processing, such as facial recognition, and thus removing rain streaks becomes a top priority project. There currently exist several de-raining programs, some of which use machine learning, but they generally ruin the picture quality either by blurring the background or by ruining image contrast. "Rain Streak Removal for Single Image via Kernel Guided CNN" suggests using a neural network called KGCNN to clean the images up with the goals being to preserve as much picture quality as possible and remove rain streaks from the image in a computationally lightweight way.

KGCNN would exploit a known property of raindrops, that being that they induce a small but perceptible amount of motion blur. Knowing the general direction of a raindrop, namely that they tend to fall down, can help us build KGCNN that will deconstruct any image feed into a background and texture layer, with the goal being to identify motion blur in the latter and use this knowledge back on the composite image to block out the rain streaks. The background layer contains everything except the rain, and the texture layer contains only the raindrops, making it easy to identify at a glance if KGCNN works as expected.

Computer systems intrusion detection

In sci-fi books such as William Gibson's *Neuromancer*, hackers plug into the internet via a literal plug inserted into the base of the skull and jockey the software around; intrusion detection is done by an AI called Black Ice that guards proprietary cyberspace and tries to fry the hackers' brains. Hackers and intrusion detection exist right now in a much more prosaic manner, but neural networks promise that at least that latter part is about to become much more interesting.

“Statistical Analysis Driven Optimized Deep Learning System for Intrusion Detection” investigates the rising threat of intelligent malware and hacking attacks that might jeopardize banking systems, power grids or hospital record databases. It’s not just that everything is networked, but the sheer size of such networks makes updating a nightmare and increases their attack surface, leaving them exposed to any hacker with an opportunity; it’s as easy as walking to one of these terminals connected to the *intranet*, the internal network, and popping in an infected USB. It’s probably how WannaCry ransomware hit 16 UK hospitals in May 2017, locking all medical files behind a paywall and threatening deletion unless \$300 in ransom was paid in Bitcoin.

This research paper shows a scalable, lightweight way to keep huge networks secure, regardless of whether their components are updated or not, by using neural networks that sift through a massive volume of data to anticipate intruder behavior and deny them access. We know from other scientific areas that neural networks can achieve near-human performance in cases of person recognition and 3D object reconstruction, so it’s of great interest to find a sustainable, cheap alternative to antivirus programs and outdated control access routines that cause endless grief to support staff, such as usernames and passwords.

The neural network assigned to network security does data preprocessing to remove outliers, feature extraction to find commonalities between users, and classification to distinguish between benign and malign users. In general, intrusions come as: probing that scouts the target network for weaknesses and open ports; denial-of-service, which serves to incapacitate target network and estimate its capabilities; user-to-root that is meant to

gain root access; and root-to-local that is meant to perform operations on a local machine after root access has been gained.

Neural network was first trained using 125,973 samples belonging to any of the four intrusion categories and tested with another 22,544 samples, resulting in an accuracy of 77.13% for probing attacks, 97.08% for denial-of-service, 87.10% for user-to-root but only 11.74% for root-to-local. These numbers imply that all security measures should focus on preventing access to root systems, which are those that can issue commands to subordinate machines, including the entire network or individual terminals; once the root has been hijacked, every compromised machine, or even the entire network, is decisively in the hands of the attacker, as seen with WannaCry ransomware.

Logical thinking

Logic sets man apart from amoebas and lets us see both ourselves and amoebas with a high degree of certainty. The ability of logical reasoning derives from a symbolic representation of the world and is the most yearned-for faculty scientists want their neural networks to have. It's not just any kind of logic, though, but a special kind called ontological logic that deals with how we came to be and what ties us together. Ontological logic can be applied to countries, humans, animals, trees, rocks or any other conceivable material or metaphysical entity to grab the fabric of time and unravel it back to its starting point. As neural networks get set to compete against humans in all fields of life and science, they'll slowly but surely learn how to navel-gaze just like the rest of us with the free time to do so.

“Ontology Reasoning with Deep Neural Networks” looks at how a neural network that’s been given facts about people draws conclusions about their relationships; for example, two separate individuals that happen to be parents to the same person must be related as well. The same logic is then applied to cities, provinces, countries themselves, etc., to let the neural network learn new things about the world and update its own information database. This is in part how Facebook’s “People You May Know” feature works – figuring out the origin of a relationship often reveals intimate

details that might have gone unnoticed even by the people involved. We do have such massive information databases, but something like Wikipedia uses throngs of unpaid volunteers bitterly bickering for weeks about interpunction in obscure articles to provide the bulk of text used by the public; this kind of a neural network could be used to either test a wiki or build a brand-new one.

Testing was done with two information databases, Claros and DBpedia. Though not the same size, the neural network learned how to correctly interpret objects, data and relations between them with a 99.8% accuracy. The authors then decided to up the challenge by removing a random fact from each database and replacing them with a diametrically opposite version of themselves, meaning “man” might have been replaced by “woman” etc. This created a conflict we would call a *paradox*, a statement that appears both true and false at the same time, but the neural network managed to resolve on average 92% of all conflicts. The authors did note that facts presented with less than 100% certainty in the information database did throw a wrench in the neural network’s spokes.

Fooling the smart machine

Videos on the Open AI blog examine how image classifying neural networks can be fooled by presenting a printed image of a kitten digitally altered to contain blocky aftereffects. The image is clearly recognizable by the human eye, but a neural network sees a desktop computer under almost all angles and zoom factors, persisting even when the image is rotated or moved aside. The related 2018 research paper titled “Synthesizing Robust Adversarial Examples” investigates the idea of turning 2D images and 3D-printed objects into a source of headaches for the neural network through the use of EOT (Expectation Over Transformation) algorithm that persists even when the image or object are rotated, filmed under different lighting or shown zoomed in or out. In the example shown in the paper, 8/10 images of a 3D-printed turtle were identified by the neural network as a rifle and the rest as “other”.

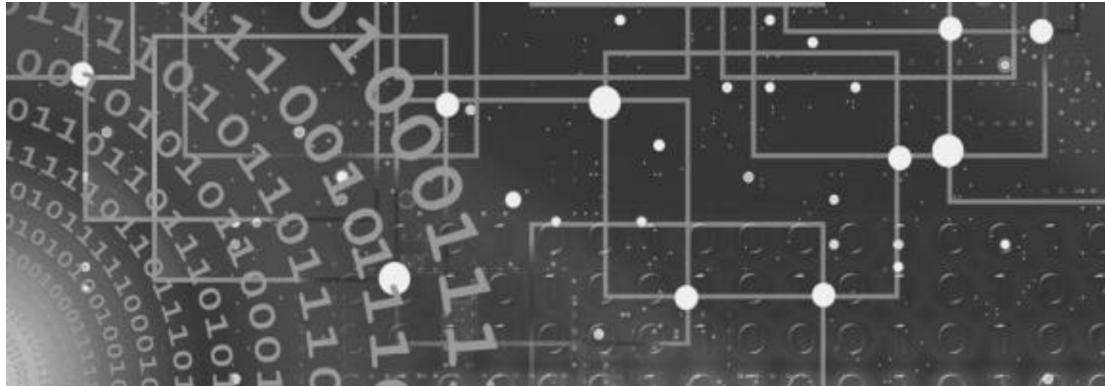
The 2015 research paper titled “DeepFool: a simple and accurate method to fool deep neural networks” describes an algorithm that adds minimal perturbations to any given picture to have the image recognition neural network see it as something else entirely, an example shown being a whale recognized as a turtle. The DeepFool method is then compared to similar perturbation algorithms in terms of cost, speed, and intrusion on the original image, discussing its use in understanding the architecture of any given smart machine and how to optimize the attack.

Chapter 6 Summary

- Search space refers to the number of architecture models we'll let the neural network assess.
- Any number of search strategies can be employed by the neural network.
- Steganography is intentionally hiding information in other data, and steganalysis is a way to unveil such hidden messages.
- Neural network was first trained using 125,973 samples belonging to any of the intrusion categories.

Chapter 7:

Neural Network Algorithms



No doubt, you've heard the term before. It is often associated with all sorts of technical mechanics but in recent years' algorithms are being used in the development of automatic learning, the field that is leading us to advancements in artificial and computational intelligence.

To do this, the algorithms have to be flexible enough to adapt and make adjustments when new data is presented. They are therefore able to give the needed solution without having to create a specific code to solve a problem. Instead of programming a rigid code into the system, the relevant data becomes part of the algorithm which in turn, allows the machine to create its own reasoning based on the data provided.

How does this work?

This might sound a little confusing but we'll try to break this down into certain examples you can relate to. One of the 'learning' functions of machines is the ability to classify information. To do this, the input data can be a mix of all types of information. The algorithm needs to identify the

different elements of the data and then group them into several different categories based on characteristics of similarities, differences, and other factors.

These characteristics can be any number of things ranging from identifying handwriting samples to the types of documents received. If this were code, the machine could only do one single function but because it is an algorithm which can be altered to fit a wide variety of things, the computer can receive this data and classify all sorts of groups that fit within the specific parameters of the circumstances.

This is how machines can change their functions to adapt to the situation at hand. Your email account can analyze all the emails you received, based on a pattern that you have followed, and it divides them into different groups. It can identify which emails are important and you should see right away, those that are spam and junk mail, and even sort out those that may pose a risk to your computer because it carries a virus or malware.

With these types of algorithms, machines can now learn by observing your habits and patterns and adjust their behavior accordingly. So, the very secret to a successful and effective neural pathway depends a great deal on the algorithms your system uses.

At its most basic level, machine learning is the use of different preprogrammed algorithms that collect and analyze data in order to determine possible outcomes within an acceptable range. Each time these algorithms receive new data the system learns and adapts to improve performance.

Commonly Used Algorithms You Should Know

Logistic Regression

Logistic regression is very popular when you need to resolve binary classification problems. This is when the solution can be one of only two options. Sometimes referred to as dichotomy, it works well with problems that require either a true/false or yes/no answer.

To understand logistic regression better, you first have to have a clear understanding of linear regression. As an analyst, you must find the best line to show a specific trend. This requires finding an equation that gives the best direct feature that addresses the regression problem.

The standard practice is to use the least square method, where the idea is to shorten the distance between the line and the training data. Basically, the system is looking for a line that is “nearest” to all the input data.

Logistic regression is a part of a specific type of algorithm called the generalized linear model. Unlike with linear regression, your objective is to find a model that comes closest to the final value of the outcome or the variable. However, remember that you are solving a binary problem so there is no set value to predict. It is just a matter of two possible outcomes. You’re actually looking for the higher probability that one outcome will actually occur.

Problems that linear regression would solve: How many inches of snowfall will we get this year?

Problems that logical regression would solve: Will it snow tomorrow?

Decision Trees

Decision tree algorithms are more often used to classify a model and label it in a tree structure. Many analysts find them to be excellent tools that can give them accurate and reliable output data.

Decision trees are easy to read and understand. In fact, when using them you will be able to see exactly why you need certain classifiers in order to make a decision. If you are new to writing code, this is probably the best algorithm to cut your teeth on.

These algorithms all have exactly the same approach; to break down the data into the smallest possible subsets (those that contain only one group of outcomes). The data is divided up based on whatever predictors are available. Then they group all subsets of the same class together. They will continue to do this until they have the smallest set of data possible.

Once this is accomplished, it is very easy to make a prediction as to the expected behavior. Making this type of prediction is very simple. All the system does is follow the path that matches the given predictors. It will lead to the subset that contains all the yes answers.

Support Vector Machines

Support Vector Machines or SVM, are algorithms that can be used like weapons.

This type of algorithm is extremely complex and utilizes some of the most challenging mathematical equations there are. Because of this unique complexity, SVMs can only slice through very small amounts of datasets. So, if the initial training data is too extensive, it is likely that SVM is not the best option.

Chapter 7 Summary

- Algorithms are used in the development of automatic learning.
- One of the ‘learning’ functions of machines is the ability to classify information.
- Logistic regression is very popular when you need to resolve binary classification problems.
- Decision tree algorithms are more often used to classify a model and label it in a tree structure.
- Support Vector Machines or SVM, are algorithms that can be used like weapons.

Chapter 8: **Machine Learning and Big Data**



Big Data is pretty much what it sounds like — the practice of dealing with large volumes of data. And by large, we are talking about astoundingly huge amounts of data — gigabytes, terabytes, petabytes of data. A petabyte, to put this size into perspective, is 10^{15} bytes. Written out that is 1 PB = 1,000,000,000,000,000 byte. And these sizes are increasing every day.

The term Big Data comes from the 1990s, although computer scientists have been dealing with large volumes of data for decades. What sets Big Data apart from data sets before is the fact the size of data sets began to overwhelm the ability of traditional data analytics software to deal with it. New database storage systems had to be created (Hadoop for example) just to hold the data and new software written to be able to deal with so much information in a meaningful way.

Volume

The term volume refers to the vast amount of data available. When the term Big Data was coined in the early 2000s, the amount of data available for analysis was overwhelming. Since then, the volume of data created has grown exponentially. In fact, the volume of data produced has become so vast that new storage solutions had to be created just to deal with it. This increase in available data shows no sign of slowing and is, in fact, increasing geometrically by doubling every two years.

Velocity

Along with the rise in the amount of data being created is the speed at which it is produced. Things like smartphones, RFID chips, and real-time facial recognition produce not only enormous amounts of data, this data is produced in real time and must be dealt with as it is created. If not processed in real time, it must be stored for later processing. The increasing speed of this data arriving strains the capacity of bandwidth, processing power, and storage space to contain it for later use.

Variety

Data does not get produced in a single format. It is stored numerically in detailed databases, produced in structure-less text and email documents, and stored digitally in streaming audio and video. There is stock market data, financial transactions, and so on, all of it uniquely structured. So not only must large amounts of data be handled very quickly, it is produced in many formats that require distinct methods of handling for each type.

Lately, two more V's have been added:

Value

Data is intrinsically valuable, but only if you are able to extract this value from it. Also, the state of input data, whether it is nicely structured in a numeric database or unstructured text message chains, affects its value. The less structure a data set has, the more work needs to be put into it before it can be processed. In this sense, well-structured data is more valuable than less-structured data.

Veracity

Not all captured data is of equal quality. When dealing with assumptions and predictions parsed out of large data sets, knowing the veracity of the data being used has an important effect on the weight given to the information studying it generates. There are many causes that limit data veracity. Data can be biased by the assumptions made by those who collected it. Software bugs can introduce errors and omission in a data set. Abnormalities can reduce data veracity like when two wind speed sensors next to each other report different wind directions. One of the sensors could be failing, but there is no way to determine this from data itself. Sources can also be of questionable veracity — in a company's social media feed are a series of very negative reviews. Were they human or bot created? Human error, as in a person signing up to a web service enters in their phone number incorrectly. And there are many more ways data veracity can be compromised.

The point of dealing with all this data is to identify useful detail out of all the noise — businesses can find ways to reduce costs, increase speed and efficiency, design new products and brands, and make more intelligent decisions. Governments can find similar benefits in studying the data produced by their citizens and industries.

Here are some examples of current uses of Big Data.

Product Development

Big Data can be used to predict customer demand. Using current and past products and services to classify key attributes, they can then model these attributes' relationships and their success in the market.

Predictive Maintenance

Buried in structured data are indices that can predict mechanical failure of machine parts and systems. Year of manufacture, make and model, and so on, provide a way to predict future breakdowns. Also, there is a wealth of unstructured data in error messages, service logs, operating temperature, and sensor data. This data, when correctly analyzed, can predict problems before they happen so maintenance can be deployed preemptively, reducing both cost and system downtime.

Customer Experience

Many businesses are nothing without their customers. Yet acquiring and keeping customers in a competitive landscape is difficult and expensive. Anything that can give a business an edge will be eagerly utilized. Using Big Data, businesses can get a much clearer view of the customer experience by examining social media, website visit metrics, call logs, and any other recorded customer interaction to modify and improve the customer experience. All in the interests of maximizing the value delivered in order to acquire and maintain customers. Offers to individual customers can become not only more personalized but more relevant and accurate. By using Big Data to identify problematic issues, businesses can handle them quickly and effectively, reducing customer churn and negative press.

Fraud & Compliance

While there may be single rogue bad actors out there in the digital universe attempting to crack system security, the real threats are from organized, well-financed teams of experts, sometimes teams supported by foreign governments. At the same time, security practices and standards never stand still but are constantly changing with new technologies and new approaches to hacking existing ones. Big Data helps identify data patterns suggesting fraud or tampering and aggregation of these large data sets makes regulatory reporting much faster.

Operation Efficiency

Not the sexiest topic, but this is the area in which Big Data is currently providing the most value and return. Analyze and assess production systems, examine customer feedback and product returns, and examine a myriad of other business factors to reduce outages and waste, and even anticipate future demand and trends. Big Data is even useful in assessing current decision-making processes and how well they function in meeting demand.

Innovation

Big Data is all about relations between meaningful labels. For a large business, this can mean examining how people, institutions, other entities, and business processes intersect, and use any interdependencies to drive new ways to take advantage of these insights. New trends can be predicted and existing trends can be better understood. This all leads to understanding what customers actually want and anticipate what they may want in the future. Knowing enough about individual customers may lead to the ability to take advantage of dynamic pricing models. Innovation driven by Big Data is really only limited by the ingenuity and creativity of the people curating it.

Machine Learning is also meant to deal with large amounts of data very quickly. But while Big Data is focused on using existing data to find trends, outliers, and anomalies, Machine Learning uses this same data to “learn” these patterns in order to deal with future data proactively. While Big Data looks to the past and present data, Machine Learning examines the present data to learn how to deal with the data that will be collected in the future. In Big Data, it is people who define what to look for and how to organize and structure this information. In Machine Learning, the algorithm teaches itself what is important through iteration over test data, and when this process is completed, the algorithm can then go ahead to new data it has never experienced before.

Chapter 8 Summary

- Since the early 2000s, the volume of data created has grown exponentially.
- If data is not processed in real time, it must be stored for later processing.
- Big Data can be used to predict customer demand.
- Many businesses are nothing without their customers.
- Big Data is all about relations between meaningful labels.
- Machine Learning is also meant to deal with large amounts of data very quickly.

Chapter 9:

What is Predictive Analytics?



For several companies, huge data - really big volumes of unstructured, semi-structured and raw structured data is an untapped source of information that can aid business decisions and improve operations. As this data continues to change and diversify, more and more companies are taking to predictive analytics to tap the source and make benefits from the large-scale data.

There is a common miscomprehension that machine learning and predictive analytics are one and the same. That is not the case. They do overlap in one area however and that is predictive modeling. Basically, predictive analytics includes a range of statistical techniques including ML, data mining and predictive modeling and uses historical and current statistics for estimating or predicting future outcomes. These outcomes could be the behavior of a customer during the purchase or probable changes in the market. It helps us to guess the possible future occurrences with the analysis of past patterns.

How Does Predictive Analytics Work?

The predictive analytics gets driven by predictive modeling. It is an approach rather than a process. ML and predictive analytics are hand-in-hand because the predictive models typically include ML algorithms. The created models can be trained over a period of time to react to new values or other data thereby delivering the results needed by the organization.

There are two kinds of predictive models. One is the classification model which predicts class membership and the second is the regression model that predicts numbers. The models are made from algorithms which perform data mining and statistical analysis to determine patterns and trends in the data. The predictive analytics software will have built-in algorithms which can be used to create predictive models. Algorithms are called as classifiers and they identify the set of categories to which the data belongs.

Commonly Used Predictive Models

The most widely used predictive models are:

- Regression (Linear and logistic)
- Decision Trees
- Neural Networks

Regression (Linear and logistic)

It is one of the more popular methods available in statistics. Regression analysis provides a relationship between variables and finds key patterns in diverse and big data sets. It also finds out how they relate to each other.

Decision Trees

The decision trees are simple yet powerful forms of multiple variable analysis. Decision trees are produced by algorithms which identify different ways of splitting the data into branch-like segments. They partition the data

into subsets depending on various categories of input variables. It helps you understand a user's path to a decision.

Neural Networks

They are built on the patterns of the neurons in the human brain. Neural networks are often referred to as artificial neural networks and are a variance of deep learning technology. They are commonly used to solve difficult pattern recognition situations and are unbelievably useful for analyzing big data sets. They are very good at handling nonlinear data relationships and also work well when some variables are unknown.

Classifiers

Every classifier approaches the data in a different manner so, for the managers to get the results they require they must select the right classifiers and models.

- Clustering algorithms: They organize data into different groups with similar members.
- Time Series algorithms: They plot the data sequentially and are useful in forecasting constant values over a period of time.
- Outlier Detection algorithms: They focus completely on anomaly detection, identifying events, observations or items that don't conform to a specific expected pattern or standards in a data set.
- Ensemble Models: These models make use of several ML algorithms for obtaining a better predictive performance than compared to the output expected from a single algorithm.
- Naive Bayes: This classifier permits you to predict a category or a class based on a provided set of features by using probability.
- Factor Analysis: It is a method used for describing variations and aims at finding independent latency in variables.
- Support Vector Machines: They are a supervised kind of machine learning technique that uses associated learning algorithms for analyzing data and recognizing patterns.

Machine Learning and Predictive Analytics Applications

The companies that are overflowing with data are struggling to turn all the information into useful insight. For these organizations, ML and predictive analytics can arrange the solution. No matter how big the data is if it cannot be used to enhance the external and internal processes and meet objectives it becomes a useless resource. Predictive analysis is used more commonly in marketing, security, risks, fraud detection and operations. Here are some of the examples of how machine learning and predictive analytics are used in various industries:

Financial Services and Banking: In the financial services and banking industry, ML and predictive analytics are used together to measure market risks, detect and decrease fraud, identify opportunities and there are several other uses.

Security: Cyber security is at the top of the agenda for almost all businesses in the modern world. It is no surprise that ML and predictive analytics play a key role in security aspects. The security organizations use predictive analysis often to improve their performance and services. They can detect anomalies, understand client behavior, detect fraud and as a result, they enhance data security.

Retail: The retail industry is using ML for understanding customer behavior better. Who is buying what and where? They want to know the answer to these queries. These questions can be answered with accurate predictive models and data sets thereby helping retailers to plan beforehand and stock items based on consumer trends and seasonality. Improves the ROI a great deal.

Developing The Right Environment

Although predictive analytics and ML can be a huge boost for most companies, implementing these solutions halfheartedly without consideration for their fitment into everyday operations will only hinder their potency to deliver the insight the company needs. To get the best out of ML and predictive analytics, companies need to make sure that they have

the architecture to support the solutions along with high-quality data which will help them in learning. Data preparation and its quality are the key portions of predictive analytics.

The input data which may span across several platforms and consist of multiple data sources need to be centralized and unified in a coherent way. To achieve this the companies need to develop good reliable data governance programs to govern the overall data management and make sure that only the high-quality data gets captured and used. Another thing is, the current processes might have to be altered to include ML and predictive analytics as this will enable companies to have efficiency at all points of the business. And most importantly the companies must know what issues they want to be resolved as it will aid them in determining the most suitable model for use.

Predictive Models

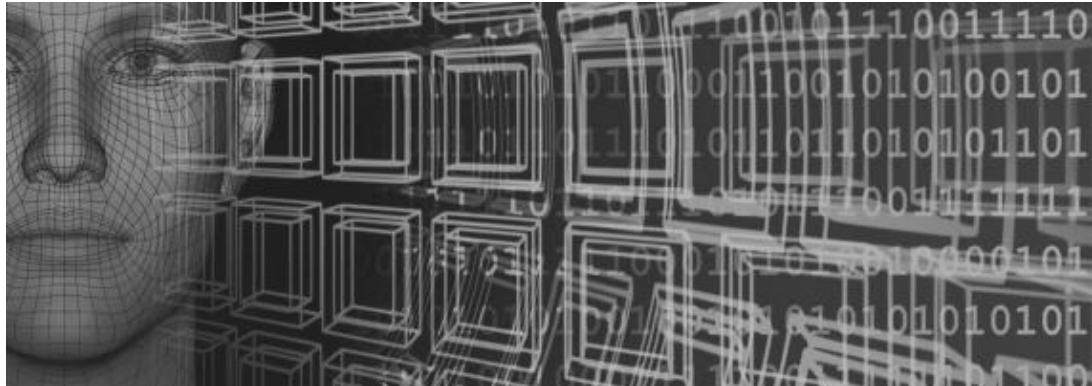
The IT experts and data scientists working in an organization are normally tasked with selecting or developing the right predictive models or possibly build one for themselves to satisfy the organization needs. However, these days ML and predictive analytics is not just the area of expertise for mathematicians, data scientists, and statisticians but there are business consultants and analysts working in the area. More and more people in businesses are using the models to develop insights and improve operations. However, there are issues when they are not aware of what model to use or how to deploy it or in the case when they need some information immediately. There is sophisticated software available to help the employees with the problem.

Chapter 9 Summary

- Predictive analytics includes a range of statistical techniques including ML, data mining and predictive modeling.
- Predictive analytics uses historical and current statistics for estimating or predicting future outcomes.
- Machine learning and predictive analytics are not the same thing, they overlap in one area however and that is predictive modeling.
- Algorithms are called as classifiers and they identify the set of categories to which the data belongs.

Chapter 10:

What is Data Mining?



Data mining means extracting knowledge out of huge quantities of data.

In other words, we can say that it is a process of discovering different kinds of patterns inherited in the data sets which are new, useful and accurate.

Data mining is an iterative process which creates descriptive and predictive models by uncovering previously unknown patterns and trends in a large quantity of data. This exercise is executed to support decision making. It is basically a subset of business analytics and is similar to experimental research. Origins of data mining can be found in statistics and databases. ML, on the other hand, works with algorithms that improve automatically

via experience they gain out of data. In other words, in machine learning, we discover new algorithms from experience. These algorithms of ML can extract information automatically but the source used for machine learning is also data. It involves two kinds of data, one is test data and the second is the training data. Data mining techniques are commonly used in machine learning and along with the learning algorithms it is used to build models of what is happening behind the scenes to predict the outcome of the future.

What is data mining and what is the relationship between ML and data mining? Data Mining means extracting knowledge out of a big amount of data. It was introduced in 1930 and at first it was referred to as knowledge discovery in databases. Data mining is utilized to get rules out of existing data. Its origins lie in conventional databases having unstructured data. It is implemented where you can develop your own models and data mining techniques are used. It is more natural and involves more involvement of human beings. They are used in cluster analysis. Data mining is abstracted from data warehousing. It is more of a research using methods similar to ML but is applied in limited sectors.

Data Mining Techniques

The specialists working in the field of data mining rely on techniques and intersection of statistics, database management, and machine learning. They have dedicated their careers to understanding what conclusions are to be drawn from a huge amount of information. What are the techniques used for turning this into reality? Data mining is effective when it draws on some of these techniques for their analysis:

1. Tracking Pattern

One of the fundamental techniques used in data mining is learning to recognize patterns in the data sets. Normally this is an aberration in the data which is happening at some intervals or a flaw or an ebb in some variables over a period of time. For example, you may observe that sales of certain products spike up immediately before the holidays. Or you may notice that warm weather drives people to your site.

2. Classification

It is a more complex data mining technique that asks you to collect different attributes together in discernable categories which can be later used to arrive at further conclusions or serve in some other function. For example, in case you are evaluating the data on an independent client's financial background and purchase history you may be able to classify the individuals as high, medium or low-risk candidates for credit. You can then use the classifications to learn more about the clients.

3. Association

This is related more to tracking patterns however, it is more specifically involved in the dependently linked variables. In the case of an association, you look for specific attributes or events which are correlated to other attributes or events. For example, you may notice that when your customer purchases some specific item they also buy another related item. This sequence of events is used to populate the "people also bought" section in the online store.

4. Outlier Detection

In some cases just identifying the overarching pattern cannot give you a clear understanding of the data set. You are also required to understand the anomalies also called outliers in the data. For example, your buyers are almost exclusively male, however, during a single week in July there is a sudden rise in female buyers. You may want to investigate the reason for the event and find out what drove the sales, so you can either replicate it or understand the behavior of your audience better.

5. Clustering

This is similar to classification, however, involves grouping of chunks of data together which is similar. For example, you may select to cluster different demographics of the customers in various packets based on how much extra income they earn or how often they are shopping at the online store.

6. Regression

Regression is used basically as a form of modeling and planning. It is used to find out the chances of presence of certain variables because some other variables are there. For example, you may use this to project some price based on factors such as consumer demand, competition, and availability. More specifically the main focus of regression is in helping you uncover exact relationships between two or more variables inside a specific data set.

7. Prediction

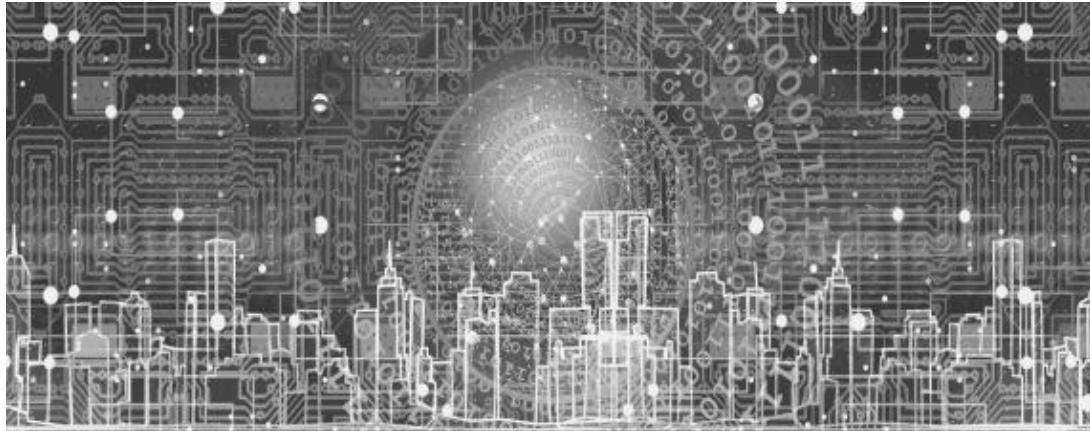
It is easily one of the most valuable data mining techniques used. This is because it is used for predicting the kind of data you will see in the future. In some cases, just by understanding and recognizing the historical trends we can chart an accurate prediction of what will happen in the future. For example, we can see the credit history of clients and their past purchases to predict whether there will be a credit risk in future in case a loan is extended.

Chapter 10 Summary

- Data mining means extracting knowledge out of huge quantities of data.
- Data mining is utilized to get rules out of existing data.
- Data mining is abstracted from data warehousing.
- Data mining is learning to recognize patterns in the data sets.
- Regression is used basically as a form of modeling and planning.

Chapter 11:

Recurrent Neural Networks



In this chapter, we are going to move on to another modern, novel neural network architecture known as the recurrent neural network.

Whereas convolutional neural networks allow us to work with raw images, recurrent neural networks allow us to work with raw sequences.

As with images, you can still “make” a sequence work with a regular feedforward neural network, but this is neither efficient nor effective.

Recall that CNNs allow you to have fewer parameters than an ANN. This is true with RNNs as well.

As promised, we’re still not going to get into the math in this book, so you’ll just have to trust me on this.

Since the theme of this book is about understanding the Keras API vs. the underlying math, we are going to skip over the equations involved in an RNN.

Let’s start by considering what the data should look like.

What is a sequence?

It's a list of ordered items. E.g. 1, 2, 3, 4, 5, ...

One example of a sequence is the USD-EUR exchange rate over time.

One might also think of a sequence like this as a time-series or more generally a "signal".

Note that an image is also a signal, but it is a 2-D signal (the two dimensions are height and width). A time-series is a 1-D signal because there's only 1 dimension (time).

As a side note, you might then wonder: if CNNs process signals and RNNs process sequences, but sequences are really just 1-dimensional signals, can CNNs be used for sequences and RNNs be used for images? The answer is yes! Researchers have tried this and found this approach to be successful.

In fact, CNNs are behind state-of-the-art speech generation systems like Google's WaveNet.

But we are getting a little ahead of ourselves here.

What is the simplest time series one can think of?

How about a sine wave?

Can we build an RNN to predict the next value in this time-series, given a set of previous values?

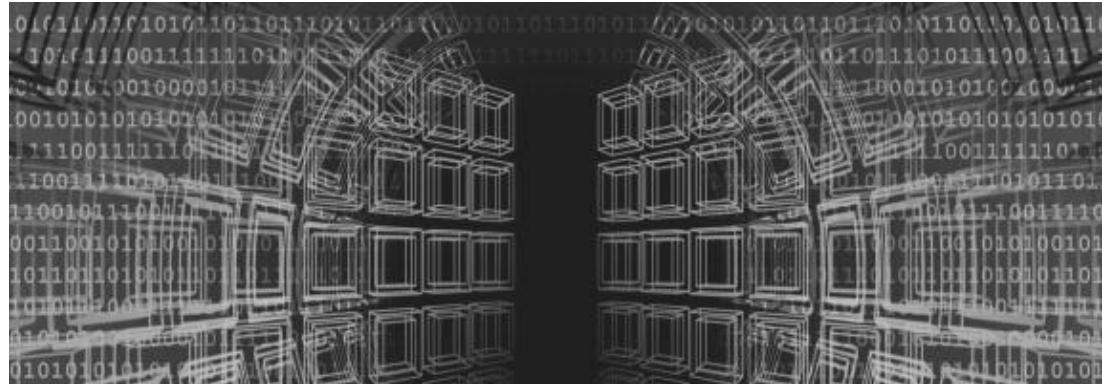
We sure can!

Chapter 11 Summary

- CNNs allow you to have fewer parameters than ANN.
- CNNs are behind state-of-the-art speech generation systems like Google's WaveNet.
- We can build an RNN to predict the next value in this time-series.

Chapter 12:

Predicting Time Series



The relevant file for this example is sine.py.

Warning: if you're not coding along with me by now, this chapter is going to get REALLY confusing.

Let's start with the imports.

```
import numpy as np  
import pandas as pd  
  
import matplotlib.pyplot as plt  
  
from keras.models import Model  
  
from keras.layers import Input, SimpleRNN, Dense
```

The only new layer we need here is the SimpleRNN.

The first thing we need to do is generate the data.

```
# make the original data  
  
series = np.sin(0.1*np.arange(200)) + np.random.randn(200)*0.1  
  
# plot it
```

```
plt.plot(series)
```

```
plt.show()
```

We've also added some noise (because "real" data is noisy), and plotted it so you know what the data looks like (not that you shouldn't know what a noisy sine wave looks like already).

Next, we need to build our dataset.

This requires a little thinking about what a sequential dataset should look like.

Recall that for feedforward ANNs, a data matrix X is a 2-D array of shape $N \times D$.

For a CNN, a data matrix X is a 4-D array of shape $N \times H \times W \times C$.

For an RNN, a data matrix X is a 3-D array of shape $N \times T \times D$.

As usual, N refers to the number of samples.

T refers to the sequence length (you can remember this by noting that the word Time starts with the letter T).

D (as in the case of ANNs) represents the vector dimensionality.

Thus, an RNN is "like" an ANN, where the input is a feature vector of length D , except that for each of the N samples, we consider T of those D -sized feature vectors in a sequence.

In the case of a sine wave, $D=1$ since there is only one dimension.

What is T ?

In this case, we're allowed to choose T .

We've generated a sine wave of length 200.

You can imagine cutting the sine wave into overlapping windows, each of size T .

Let us suppose $T=10$.

That means each sample sequence $X[n]$ will contain 10 sequential values from the sine wave, and $Y[n]$ (the corresponding target) will be the 11th value of the sine wave.

In other words, our RNN model will learn to take in as input a sequence of 10 steps in a time series, and forecast the next step.

Let us express that in code:

```
T = 10
```

```
D = 1
```

```
X = []
```

```
Y = []
```

```
for t in range(len(series) - T - 1):
```

```
    x = series[t:t+T]
```

```
    X.append(x)
```

```
    y = series[t+T]
```

```
    Y.append(y)
```

```
X = np.array(X)
```

```
Y = np.array(Y)
```

```
N = len(X)
```

We start by initializing X and Y to empty lists.

Since each sample will consist of $T+1$ values (T for the input and 1 for the output) we can only loop up to $\text{len(series)} - T - 1$.

The T samples corresponding to the input are $\text{series}[t:t+T]$ and the 1 sample corresponding to the output is $\text{series}[t+T]$.

We append these to X and Y at each iteration of the loop.

After the loop is complete, we cast X and Y to Numpy arrays.

Note that at this point, X is only a 2-D array of shape $N \times T$ (this is because each x we added to X was a 1-D array of length T).

Thus, in order for X to be in the correct format for the RNN (a 3-D array of shape $N \times T \times D$) we need to add the extra 1 dimension to make it $N \times T \times 1$.

```
inputs = np.expand_dims(X, -1)
```

Next, we build the model:

```
i = Input(shape=(T, D))
```

```
x = SimpleRNN(5)(i)
```

```
x = Dense(1)(x)
```

```
model = Model(i, x)
```

Much simpler than a CNN and almost as simple as an ANN. Nice!

As usual, we start with an Input layer. As expected, we specify the input shape as (T, D) (remember the number of samples N is implicit).

We pass it through a SimpleRNN layer.

We create a final Dense layer to map the SimpleRNN's hidden values to a single value (hence the 1 argument in the Dense constructor).

Unlike the previous examples in this book, we are now performing regression instead of classification.

Since we're doing regression, we don't want the final dense layer to have any activation function at all. Recall that the softmax activation bounds the output to be between 0 and 1, so that it represents a valid probability distribution.

Regression outputs can take on any value because you're not trying to predict a class, you're trying to predict a number.

```
model.compile(
```

```
    loss='mse',
```

```
    optimizer='adam',
```

Accordingly, we also don't want to use the categorical_crossentropy loss, because that's for classification. Instead, we want the mean squared error, specified as "mse".

As usual, we use the "adam" optimizer.

Next, we fit the data:

```
r = model.fit(
```

```
inputs[:-N//2], Y[:-N//2],  
batch_size=32,  
epochs=80,  
validation_data=(inputs[-N//2:], Y[-N//2:]),
```

I've selected the first half of the data to be the training samples and the second half to be the validation samples. Why?

Unlike with classification, where we can shuffle the data, this doesn't really make sense when you're dealing with time series.

For classification, it makes sense to shuffle your data because it's not time-dependent, a picture of a cat looks like a picture of a cat no matter when it was taken. A picture of a cat from the future will probably look similar.

For time series, this idea of the "future" really matters. I don't want to mix data. Suppose I have data from days 1, 2, 3, ..., 10.

If I train on days 2, 4, 6, 8, 10, and test on days 1, 3, 5, 7, 9, then I'm not really predicting the future. I'm using the future to predict the past. That's not useful.

What is useful is training on days 1, 2, 3, 4, 5 and using that model to predict days 6...10.

Of course, in our simple sine wave example this doesn't really matter since a sine wave just repeats the same pattern over and over again. For real time-series, it does matter.

Next, as usual, we plot the loss:

```
plt.plot(r.history['loss'], label='loss')  
plt.plot(r.history['val_loss'], label='val_loss')  
plt.legend()  
plt.show()
```

Note that for regression problems, there is no notion of accuracy. If the target is 5, and you predict 5.000001, intuitively, you know that your prediction is close to the target. That is a good prediction. But $5 \neq$

5.000001, so it's not "correct". Thus, any concept of correct rate or error rate here is irrelevant.

In a classification script, things would end here because we've observed that the loss has converged to nearly 0.

In regression, since this is a time-series, we can plot our predictions against the true targets!

```
outputs = model.predict(inputs)  
print(outputs.shape)  
predictions = outputs[:,0]  
plt.plot(Y, label='targets')  
plt.plot(predictions, label='predictions')  
plt.title("many-to-one RNN")  
plt.legend()  
plt.show()
```

Note that since the final layer is a Dense(1), the output returned by `model.predict()` is of shape Nx1.

In order to plot this data, we select the first (and only) column using `outputs[:,0]`.

Chapter 12 Summary

- For a CNN, a data matrix X is a 4-D array of shape $N \times H \times W \times C$.
- For an RNN, a data matrix X is a 3-D array of shape $N \times T \times D$.
- N refers to the number of samples.
- T refers to the sequence length.
- D represents the vector dimensionality.

Chapter 13:

Neural Networks And Artificial Intelligence



Only humans have the ability to think, analyze, and understand their environment, its actions, and the realities that exist. Only humans can explain their world and think in the abstract. Our brains can take in a wealth of information and draw conclusions in the fraction of a second.

Machines, on the other hand, using the same set of data, will face numerous challenges. For example, today, computers can hold a database of thousands of similar images, assess them and draw a conclusion with 95% efficiency. This is quite impressive by any stretch but still, it cannot explain why it chose the images, ascertain their meaning, or distinguish why one image is different from another. In other words, computers can compute but they can't reason. So, even though they are capable of producing amazing results with the tasks they are given, they still are far behind the ability of the human brain in many ways.

To overcome these obstacles, a unique form of machine learning has been developed. Most of us know it as Artificial Intelligence or AI. The term

refers to the simulation of intelligence in machines. Today's machines are programmed to 'think' and mimic the way the human brain operates. These machines are slowly taught to rationalize in given situations, analyze, and choose a course of action that would have the best chance of achieving the optimum goal.

As technology continues to advance, it is only natural that older machines will become outdated. For example, machines that perform basic functions or can recognize and identify text used to be cutting edge but today, they are so obsolete that they are no longer considered to be artificial intelligence. This is because the function now is taken for granted. This ability has become so commonplace that it is often viewed as a normal computer function.

Computers said to have artificial intelligence today are those that show more impressive abilities like being able to play chess, self-driving cars, and smart homes. Why are these functions given the label of artificial intelligence? For those that can play chess, these machines can win the game against a human opponent, self-driving cars have mastered the ability to absorb all the external data surrounding them and compute it fast enough that they can navigate and reach their destination without causing an accident of some kind. Smart homes can interact with members of the household, control the temperature, manage security, and even food supply to ensure the comfort of its inhabitants.

Of course, when anything completely new is introduced to the world, there is often some skepticism involved and the same could be true in regards to artificial intelligence. Even though we interact with AI on a daily basis, most of us don't realize it. Whenever you talk to SIRI through an Apple device, you're using artificial intelligence. Alexa from Amazon also makes use of the same technology. These devices, though impressive, are considered to be what is called 'narrow AI' or a 'weak AI.' They are only programmed to perform a very small number of tasks. These merely represent the first introductory steps of AI on a long-term goal towards a more human interaction with machines. With that said, aside from the fears that many people have about AI, there are numerous benefits that we will gain from these advantages.

Artificial intelligence in medicine

When it comes to healthcare, artificial intelligence is already making impressive progress, especially in the field of health. While we have not reached the point of developing a strong AI, artificial intelligence is already a major part of our lives. With a machine capable of thinking and reasoning, treatment of patients will be much more efficient. Physicians will have easy access to all the data they need and these machines will be instrumental in helping them make a good decision.

This type of technology won't be available until sometime far into the future. Currently, many people are creating algorithms that will help them solve many of the problems in healthcare today. We're not looking at 100 or 200 years down the road but more like five to ten years.

Right now, IBM's Watson is leading the pack with its cognitive computing used for healthcare. They are closely followed by neural networks developed by big names such as Dell, Hewlett-Packard, Apple, Hitachi, Digital Reasoning, Sentient Technologies, and more.

With each new advancement made by these companies, health care improves by a greater margin. Right now, there are countless areas in the field of medicine being addressed. We will see progress in the following areas:

Artificial intelligence in finance

Another area where artificial intelligence is already making an impression is in the finance industry. As of now, nearly every financial company in the industry is ready to embrace AI. It not only will save them time, but it will also cut costs, and add value to their services.

A perfect example of this is the AI computer Wealthfront, which tracks user account activity in order to analyze the account holder's spending behavior. By understanding this behavior, the company can help them make proper financial decisions and even customize the advice they give to their clients.

As of 2016, more than 550 financial companies had already started using artificial intelligence in their business, generating an additional \$5 billion in revenue.

Consider the results generated by JPMorgan Chase's 'Contract Intelligence' platform, which uses AI image recognition software to extract important

data from legal documents. Normally, this task would take approximately 360,000 hours of manpower while the AI did it all in a matter of seconds.

Finance is expected to be heavily influenced by artificial intelligence in every aspect. It is already setting the stage for a complete renovation among the industry leaders, making them more competitive in the process. Any company that has not tapped into artificial intelligence will quickly find themselves falling behind in their ability to maximize their resources, lower their risks, increase their revenue streams, increase their ability to trade, invest, lend, and more.

All of this is now being done through automation. Many of the once repetitive tasks are now being handled through neural networks in artificial intelligence. Their software is capable of matching data records, analyzing and looking for exceptions, and placing the decision-making process in the hand of this new technology. Since AI is designed to be a learning machine that will be constantly improving itself, in time they will be able to take over more of the mundane and repetitive jobs that are now consuming many hours of manpower.

Right now, AI in the finance industry is primarily used in three areas:

- Calculating parameters and numbers beyond human capabilities
- Analyzing and interpreting the written text by using a special program that allows them to grasp the context of the language used in contracts and agreements
- Pattern recognition to detect different types of activity, audit books, and inspect finances in ways that go beyond human accuracy

The benefits of AI in the finance industry will be better trading by making selections through algorithms, better investment insights, improved risk management, improved fraud detection, and better choices when it comes to making credit decisions.

There are countless applications that are already being put to good use in financial services, and there is more of that to come in the future. So, the next time you need to make an investment decision or receive a recommendation from your financial institution, it's a good bet that you're getting your advice from some form of artificial intelligence.

Artificial intelligence in translation

Today, there are more than 6,000 languages spoken around the globe. Anytime someone needs to communicate with someone from another language group problems arise. In most cases, if one party does not understand the other, there will be a need for a translator. This is not only time consuming but expensive.

Now though, according to two separate research papers, artificial intelligence is beginning to become a major part of the translation industry. There are now several different AI programs that use the unsupervised artificial intelligence that can perform language translation efficiently without the use of a language dictionary. The methods work both with parallel text as well as with identical text that is already being used in other languages.

These programs begin with data specifically chosen to supply them with an extensive bilingual dictionary that they can reference without the aid of a human to verify their answers. The machines rely on the relationship that exists between words. For example, tree and leaves or shoes and socks. These relationships exist in some form in all languages. The AI looks at these words and groups them into clusters and connects them from one language to another to help them understand how the syntax of another language is formed.

The dictionary the computer builds from the data is then put together with two additional AI methods called ‘back translation’ and ‘denoising.’ Back translation translates one sentence into the new language and then translates it back again as a way of testing its accuracy. If the back-translated sentence doesn’t match the original, the system will automatically adjust its parameters and make a second attempt. It will continue to do this until it gets as close as possible to the correct answer. Denoising works in a similar way but goes an additional step. It randomly removes a word from the sentence to make sure that the AI is also building on the structure of the sentence rather than just translating them word for word.

Google and Facebook both have incorporated language translation into their platform successfully. We can fully expect more artificial intelligence applications in language translation to be introduced in the future.

There is no doubt that machine translation has improved a great deal in recent years. It is much more accurate, it works at impressive speeds, and it is readily available in almost every language you speak.

A good example of this is Google translate. According to their records, the machine translator has an impressive accuracy that is on par with human translators. Every day, it translates Spanish, Chinese, French, as well as many other languages to English and back again both accurately and quickly. As their program continues to learn, it has reached high accuracy levels so it can be relied on in a wide range of business applications. Today, machine translation is being used in government, software and technology, military and defense, healthcare, finance, legal, and E-commerce.

Whether the need is for text to text, text to speech, speech to text, speech to speech, or even an image to text, there is an AI program that can handle it.

Game playing

While the other applications of artificial intelligence deal with the serious matters of business, money, and health, there is another area where these programs have made a lasting impression that is not always so serious. This is not to say that game playing is not as important as the other industries. In fact, for many, the art of game playing is, in reality, a very serious business.

It would actually be a miscarriage of justice to ignore the gaming industry when talking about modern technology. For many decades now, board games have been at the center of AI research, in more recent years we've witnessed the progression widen out to incorporate video games as well. In fact, video games have become even more sophisticated in many ways, mainly because of the introduction of artificial intelligence.

These programs are designed to control non-player characters, generating data about their opponents and adapting to their behaviors. Game developers have realized that using AI programs to analyze large amounts of data generated from players from all over the globe has helped them develop games that are more intricately designed.

While gamers are a small community in comparison with the other major industries, it is growing in popularity mainly because of what they have learned by applying artificial intelligence analysis in their designs.

It is a thriving field of research, but it is also a phenomenal playing field for testing out new algorithms that can one day be adopted into other industries as well. In fact, it is with the gaming industry that AI often introduces new programs that have gone on to increase computational power and generate countless stories of success in many other industries.

Think about some of the common uses for AI today that was actually started and tested first in the gaming world.

- Image and speech recognition
- Emotion detection
- Self-driving cars
- Web searching
- Creative design

All of these concepts that are now part of the real world were first tested in a game of some kind.

All this helps us see just how important the gaming world is to modern technology. Just like many scientific ideas were first introduced to the world through science fiction books, TV shows, and movies, much of the technology that makes them possible was introduced through games.

But what about game playing itself? The fact is, the gaming world is the safest environment where scientific problem solving can work in harmony with creativity. AI helps humans to better understand how we play games, how we understand the interactions between players, and therefore how to build a better game.

But gaming AI is not limited to the board games or even the online versions of many games but is playing a major role in the gaming industry. You know the kind of games that go in Reno, Las Vegas, and Atlantic City? Once upon a time, these were the only locations where one could put money down on a poker game or gamble a little on the turn of a roulette wheel. But because of artificial intelligence, these games can now be played with just as much vigor as you would if you were there in person. Now, poker games are popping up all over the world. You might be sitting across

the table with one player in Japan, another in Brazil, and another in South Africa.

These types of games have become profitable ventures on all fronts. Many people who would have previously been unknown have become world-renowned players for their winning streaks.

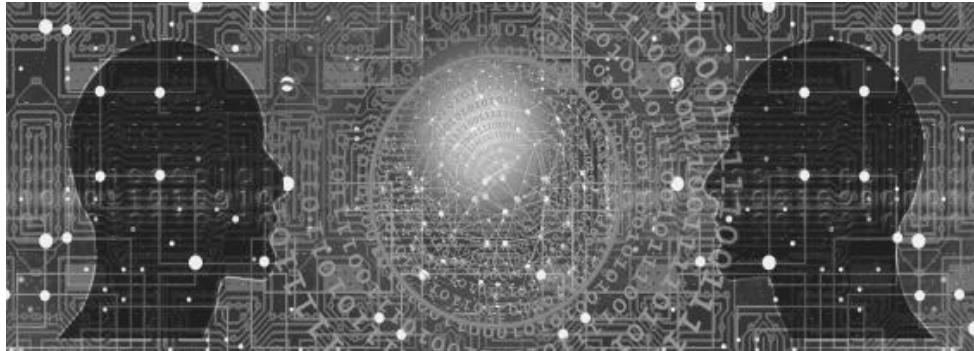
New games have also begun to emerge as a result of artificial intelligence. The game Nevermind, introduced in 2016 is capable of tracking how a player's emotions unfold during the game and adjust its playing strategy accordingly.

Games have been a prominent area for testing AI and will continue to be so. Not only does it provide programmers a safe place to test new algorithms and designs, but it is also the one place where there can be true human-computer interactions so we can see how it plays out in the future. The simple fact that games are so popular opens the door to amazing possibilities. People are more willing to test them out, computers have a wealth of data to analyze, and they can create a host of problems and challenges that can be worked out over time.

Chapter 13 Summary

- Machines using the same set of data, will face numerous challenges.
- Artificial Intelligence refers to the simulation of intelligence in machines.
- Artificial intelligence is already making an impression in the finance industry, health care, translation and game playing.

Chapter 14: **Technical Fields and How They Operate**



Even the most advanced neural networks are pretty primitive when compared to the human brain. They tend to process information one at a time, whereas the biological brain can easily multitask, capable of doing several things at once and that's all without having to burn extra energy to get it done.

Think of what you do every day. You can prepare a meal, have a conversation, and fill out your shopping list all at the same time. If you work in an office, you may be typing a letter, taking phone calls, and doing research altogether. In comparison, the artificial neural networks generally ‘learn’ by comparing one classification of a record with another classification. When there are errors that result from this process, they will readjust their formula and modify the

algorithm and repeat the process, each time reducing the rate of error until they find a solution that best matches the task at hand.

Training the neural network

Let's look at this process in a bit more detail. First by looking at what a neural network is really composed of.

1. A set of input values and their weights.
2. A function that calculates the sum of the weights and then maps the results to an output

These two elements and how they work together can be seen clearly in the diagram below.

As you can see, the neurons on the input level receive the data input. The hidden layer directly above the input layer copies the data and any weights associated with it. Then these sums are totaled up and measured against known biases before the data is passed on to an activation layer where the decision will be made to fire the neuron or not. If the neuron is fired, then an appropriate output is determined, which will send a signal to the external environment dictating what action, if any, should be taken.

To visualize this better, it helps to know that there are no physical neurons on the different layers but rather a compilation of coded values contained in a data record. The next layer or series of layers are the hidden layers, where most of the computations will be performed, and the final layer or the output layer contains a single node for each class. As the data passes through the network, every node in the output layer is assigned a specific value, and the record is awarded to the node that contains the highest value.

In supervised training, during the training phase, each record has a known value so the output nodes can be assigned the correct values. A “1” is assigned to the node with the correct class and a “0” for all the others. This way, the computer can compare the calculated values with the correct values and determine the percentage of error for each node. This is called the Delta Rule.

The delta rule

Sometimes referred to as the ‘least mean square method,’ the delta rule is probably the most commonly used rule in training a neural network. With every input vector, the system compares the output vector to the correct solution. If the difference is 0, it means that the system learned nothing in the process. If the value is anything other than zero, the system automatically goes back to the weights associated with the input data and makes adjustments to bring the difference closer to zero.

For example, if you wanted to calculate the answer for 3×4 , the computer will likely process the information as $4 + 4 + 4$ and give you an output of 12. This is pretty basic thinking and both human brains and neural networks can get the result in a matter of seconds. However, let’s suppose the problem is a bit more complex.

Perhaps we don’t know the right formula to calculate the right answer. All we know is that the information is linear, which means that if one number in the equation is doubled, the other number must also be doubled. In a biological brain, this type of linear thinking is called ‘intuition.’

The relationship between the different factors gives an indication of what needs to be done. The computer will apply an equation which reflects that relationship to solve the problem.

The only other data the system will have could be some examples that multiply numbers together to get the correct value. This data

stored in the memory banks give real evidence that shows exactly how to arrive at the correct answer. The program will then process that calculation and compare it to the data it already has. If the difference is off, it will go back and make adjustments and then recalculate to solve for a better answer. It will continue to do this until it gets an answer that is either exact or close enough that it cannot reduce the error rate any longer.

The network ‘learns’ by checking the error rate and continuously readjusting until it gets to the right solution. This continuous refinement of the answer as a means to get to the nearest point of the correct answer is the simplest form of training the neural network.

There are many algorithms that are already programmed into the system that can be called upon to find the right solution to a given problem. These algorithms are chosen based on the type of problem the computer needs to solve.

Clustering

But how can we get a network to classify data rather than calculate it? This simply requires using a different type of algorithm. Let’s say you have images of dogs and birds all mixed together and you want the computer to separate those images. In a biological brain, you can tell in a fraction of a second which ones are cats and which ones are dogs, but this is something a computer can’t do.

This is almost the same general process of repeating the algorithm and constantly refining the answer until you have successfully completed the task. However, in the case of classification, the network doesn’t need to come up with some type of mathematical theory to solve the problem. Instead, it relies on some examples to guide its choices.

The data received may detail specific characteristics of dogs and birds. These are characteristics that are common knowledge and everyone knows to be true. These details are referred to as the

‘training data.’ Of course, the data needs to be in a language that the computer can understand. Descriptions such as “A dog has four legs, a tail, and pointed ears” will not do. Instead, the input data will be in the form of a table or a graph.

Example	Width	Length	Features	Classification
1	6” - 24’	1’ - 5’	Hair	Dog
2	2” - 2’	5” - 3’	Feathers	Bird

The network will use this set of examples to plot a slope separating any data that matches the criteria. It will begin with a random division so that it has a baseline to work with. It will then check its answers against the known truths it already has to see how close it is to the correct division. Then it will repeat the process over and over again until it reaches the right solution. This process is called ‘clustering’ and is used for classifying different things, and for making predictions based on data received.

Once the correct result is attained, the data is stored in the memory banks and is used as an example the next time another problem like this comes up. In this way, the computer actually ‘learns.’

These are not the only ways a neural network can learn but as you can see from the examples, the process is very basic. It is important, however, to understand that when the system has to figure out a problem with multiple elements in it, each hidden layer in the network can only perform one function. So, an extremely complex problem may require many hidden layers in order to come up with the right solution.

To put it simply, these programs cannot read words so basic mathematical algorithms are used to determine the relationship between the output error of a problem and the correct answer (or a solution that will solve the problem). This process is repeated over and over again, refining the answer until it reaches a point where it cannot be improved any further. In order for this to be accomplished, the neural network must be set up with real-world

examples of data that is known to be true and accurate. If the data the system relies on is not accurate it is not possible for the computer to learn and expand that knowledge.

Chapter 14 Summary

- Artificial neural networks generally ‘learn’ by comparing one classification of a record with another classification.
- Neurons on the input level receive the data input.
- As the data passes through the network, to every node in the output layer is assigned a specific value.
- The network ‘learns’ by checking the error rate and continuously readjusting until it gets to the right solution.
- The data received may detail specific characteristics of dogs and birds.

Chapter 15:

Neural Networks in the Future



It's hard to imagine the possibilities the future holds for neural networks. Because of how this technology is already integrating themselves into every aspect of our lives, the potential for new and innovative ideas is higher than ever. We can envision a Jetson-like world where we will have self-driving cars instead of GPS devices that need to be programmed with our intended destination. Imagine a car that has learned your personal preferences in the music you listen to, the temperature you're most comfortable with, and how to perfectly adjust your seat.

But all of that is possible now. The future holds a lot more possibilities where neural networks can be applied.

There are two different ways that this new learning technology can grow. One area is in the field of virtual intelligence. This type of program could be planned, controlled, predictable, and could eventually become the next evolutionary step in artificial intelligence.

This type of intelligence would be even closer to matching its thinking and learning styles to humans. A machine that can evolve and grow with mankind, adapt to the same environment and learn from its experiences is inevitable.

To advance to this point, however, requires technology that can actually understand and make the necessary adjustments to bridge the gap that now exists between AI and VI. As this new technology slowly engages in our world, more of our activities will be played out in virtual reality. We'll find ourselves spending more time with computers, giving us loads of data to share. We'll communicate through the use of avatars, social platforms, and games.

These virtual worlds will have to be created though, but these are places where it is safe to learn, try, and fail at our attempts to improve. They will take the place of social platforms and make it possible for us to hone our skills in business, finance, and even romance. Whatever you want to test out, there will be a virtual world to work in before you make your idea mainstream.

This will eventually become a fully automated world but not a self-aware world as many people fear. Humans will still set the parameters and put limits on the kind of things they want computers to do. Their intelligent software will be able to simplify and enhance our real life but not take charge and control it. As long as humans put limits on the computer's ability to grow, the future will remain bright for this technological advancement.

Right now, artificial intelligence is still in its infancy, the next decade could be a real eye-opener. Not very far in the future, we will begin to see these machines change the way cars and planes are designed, how they will be operated, and how they will interact with humans. We will watch our days of exploration go further and further into space. In time, we will witness the colonization of new worlds. This time literally.

The future also has many changes in store for a country's military might. Soon, there won't be a need for "boots on the ground" when a country is at war. One soldier will be able to manage an entire fleet of drones that will fight in their place. These are already in use in some partial form now. Called unmanned aerial vehicles or UAVs these drones are capable of being operated from a remote location and responding to a myriad of instructions. In time, these UAVs will become autonomous and work without the aid of human direction.

What does this mean? Imagine a fleet of drones all headed for a single target. If one drone is destroyed by enemy fire, the remaining drones could automatically reassemble and continue on to accomplish their mission. Their ability to learn and grow will allow them to adapt to the function of the destroyed drone and incorporate his assigned task into their programming.

It is expected that with each new system introduced, machine programming will increase in its complexity and capabilities. Today, we think that artificial intelligence is one of the most fascinating forms of technology known to man. What will we think when virtual intelligence becomes available to the mainstream population? These machines will be more capable of interacting with humans and will revolutionize every aspect of our daily life.

Another area where this new technology will improve human life is in the field of disaster response. Areas unsafe for humans to enter can now be accessed by deploying machines to bring aid to people who are cut off from the rest of the world by catastrophic events. Imagine how these intelligent programs can be installed in machines that can search for life underneath the rubble of ruined buildings. How food supplies can be delivered quickly and safely. How rebuilding efforts will be much faster and how the treatment of the injured will be done quickly and efficiently.

We'll see this technology in the movie industry, music, in agriculture, and in an endless parade of other industries as time progresses. Right now, we are pretty sure of what the future holds for a neural network and all of its many applications. What we are not sure of is how quickly humankind will embrace it. No doubt, it will be the younger and more adventurous

generation that will embrace it first. They will be the ones to harness its immense potential and they will be the ones who will have to set its limits.

Science has a lot to offer us in the way of advanced computer technology. The machines which will be produced tomorrow and, in the years, to come will open the door to a whole new world of adventure. But it will happen because people are driven by the powerful force of human desire to always find better ways to do things that are stronger than the many who are powerful and in time, they will make the science fiction of the past become the reality of today.

Implementation and Interpretation

You've analyzed your data, chosen an algorithm, and started training it with the type of data you want to use it to interpret. Even after your machine learning algorithm is all trained up and ready, though, there are a couple more things you need to do before you're really ready to start putting your intelligent software to work.

Analyzing the data will tell you what kind of algorithm to use to interpret it, but before you can actually use the algorithm, you'll likely need to do some prep work on your data. Properly preparing your data helps to ensure that you get the results you're looking for and that the algorithm functions the way you intend.

How much preparation you'll need to do depends on the nature of the data you're working with. In the case of especially large quantities of unlabeled data, you may want to run an unsupervised algorithm on it first to become better acquainted with the underlying structure and help you decide how best to utilize it.

The number and types of features and attributes you want to consider will also have an impact on how much preparation work you need to do on your data. If there are a lot of missing features or outliers, cleaning up the data can help your models run more efficiently. You may also need to transform the data by compiling it or scaling so it's easier for the program to process.

You may also find that you don't want to use every piece of data that you have available to train your algorithm. Curating the dataset that the algorithm learns from can help to direct the types of situations it predicts well. You may choose to leave out entire portions of the data, or simply to have the program ignore certain features.

Before you start creating models, you'll want to test your algorithm thoroughly against known data to test the accuracy of the predictions. This can show you any weak spots that you still need to improve, making sure that, once you're using it for real, the results it gives you are the best results possible.

Iteration

Machine learning algorithms learn by doing. The voice control software Alexa is a prime example of this. The more you interact with it, the more it learns about your speaking style and preferences, and the better it's able to anticipate your needs.

The more data your algorithms process, the better they'll be at performing their intended function. This doesn't mean you should just pour in a heap of data all at once, though. Just like people, the algorithms work best when they can learn in increments. Running short iterative cycles of data through your algorithm allows you to fine-tune it as you go, streamlining the process and ultimately letting you create working models faster.

When you review your results at the end of each iteration, don't just look for overall accuracy. You should also consider the results in the context of the kind of information you're looking for. If they're not, you can re-tool the features that are considered between iterations to maximize the algorithm's usefulness for your life.

Interpretation

Being able to understand and trust your models is the core requirement of good data science. It can be easy to fall into the trap of believing machine learning algorithms are the answers to all of your data interpretation woes.

While they can be immensely helpful, your ability to use the algorithms effectively is equally important to your success.

The data you get from machine learning algorithms are useless if you don't know how to frame it in a way you and others can use and understand. This is the same basic concept but at the end of the process, and helps to make sure everyone can understand the results you've produced.

Always show the context of the problem and the steps you took to find the solution with your results. Even the most valuable data analysis can seem useless if it's viewed out of context or in a vacuum. Putting your results in context lets you see the big picture, which can help to make further connections, and at the least makes sure you're getting the most value out of your data.

Being able to express your results in terms that even lay people can understand is one good indication that you understand the problem, the model, and the data completely. Even if you won't be presenting the data to anyone outside your industry, compiling it in a way that would be comprehensible to outsiders can be a good test of how well you know your own data.

Your data is only as valuable as the results you're able to draw from it. If you're not getting results that you can use, you should ask yourself if you truly understand your data and your model. Double-check your model for accuracy to make sure the answers that it's given you are the ones that you can trust.

Even if you do completely understand and trust your data and methods, the results that are given by machine learning algorithms are often quite complex. They can be tricky to understand even for people who are experts in their industry.

This is largely because, with the exception of more linear models like Linear Regression, the majority of the results produced by machine learning algorithms are nonlinear, non-monotonic, and even non-continuous or non-polynomial. This can make visual representations of the results difficult to create and understand.

If you're going for maximum comprehension, Linear Regression and other similar models are likely your right choice. Since every change in a given

variable response happens at a defined rate and in only one direction, the models are more intuitive. It's easier to draw clear conclusions about the predictions. They may not have as high a level of accuracy as other models and aren't suitable for every situation, but their elegant simplicity makes them much easier to interpret.

With linear models, automatic interpretation of results by an additional software program is more possible than with other models, as well. When explanations for the responses of the program are created automatically, they are known as "reason codes." You'll want to make absolutely sure your model is behaving as expected before going this route, but it can save you a lot of time in the interpretation phase.

Even though most other algorithms are nonlinear, you can impose constraints either during or after the modeling that can make them function as though they're monotonic in regards to one variable. Doing this can help to isolate certain aspects of the data to make it easier to use, though it does obviously also limit the scope of your results.

In some cases, you can't avoid the non-linear and non-monotonic qualities of machine learning algorithms, especially in situations and use cases where the data is especially complex. This means you will have to deal with multiple variables that can change in any direction at a varying rate.

In these situations, a comprehensive explanation of the data that can be interpreted by a layperson may not be a viable option. Instead, you may want to stick to more technical interpretations for the purpose of cross-validation, error checking, and initial analysis. You can then break this down into subsections that can be more easily expressed through a graph or other visual representation.

Using a combination of techniques to interpret the data is often necessary with these more complicated algorithms and results. The exact techniques that will be appropriate vary depending on the type of algorithm and the nature of the data. Machine learning algorithms can deliver results that are intricate and sophisticated. Using multiple methods to examine the results is the best way to cope with this complexity.

Scope

There are two general kinds of scopes you can use to interpret data: global and local. If the model has global interpretability, it can help to understand the relationship between variables across the data in its entirety. Global models can give you more insight into things like population distributions or can help you track complex causal relationships. This can result in a very complex model, however, and models with a global interpretation are more difficult to interpret.

Models with local interpretability instead focus on smaller regions within the data. By looking at particular clusters of the inputted data, you can get a more in-depth look at the interactions and relationships between functions and variables. Since you're working with a more limited subset of the data, local interpretations are more likely to be expressible using monotonic or linear means. This means they're often easier and faster to interpret.

Representational techniques

The whole reason people use things like machine learning algorithms is that real-world data is complex and difficult to visualize. There are more rows and variables than the human mind can make sense of simply by looking at them. Most people can only visualize things effectively in either two or three dimensions, for one thing. There's also an effect known as "Change Blindness," which basically means that it's more difficult to think in an analytical way about information that's split across multiple screens or pages.

There are a few different techniques you can utilize if you want to express complex data visually in a way that can actually be useful. One of the first things you should consider is making greater use of symbols and glyphs in your graphs. You can use visual aspects like the shape, color, opacity, and size of the graphed data to express variations in extra features without adding dimensions.

You can also make 2D projections of multi-dimensional data to help you read it more clearly. This involves flattening the rows of the dataset into a lower-dimensional format without losing the meaning. There are a few popular techniques, each of which has their strengths. The most widely used are Principal Component Analysis, Autoencoder Networks, Multidimensional Scaling, and T-Distributed Stochastic Neighbor

Embedding. 2D projections are good for analyzing structural aspects of the data, like seeing clusters and sparsity or identifying outliers.

A correlation graph is another kind of 2D representation to show relationships between points in a complex dataset. It can be an especially powerful tool if your end goal is a clustering or classification problem. Even datasets that have tens of thousands of variables can be represented in a simple graph using these methods.

You may also find it helpful having a visual representation of the algorithm itself, especially if you find you're not getting the kind of results you're looking for and aren't sure why. Running a residual analysis can show you the difference between the recorded value of dependent variables and what the algorithm predicted. It's basically an easy way to see the accuracy of your model.

If the model is well-fit, the residuals will be randomly-distributed outliers. If you see strong patterns in the residual analysis, that's a sign you have a problem. You can then isolate and analyze that particular data to locate the source of your problem.

You may also want to utilize partial dependence plots. These show how your algorithm functions, especially as related to the variable that is of most interest to you. It can be a way to identify which response functions are monotonal and grants you deeper insights into the way nonlinear data is interacting within the model.

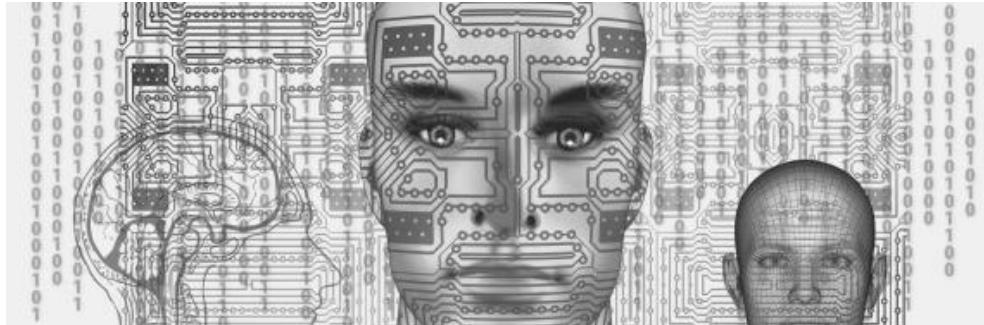
Partial dependence plots are an easy way to make sure you can trust your model's results. It lets you see which relationships are conforming to your expectations and which ones aren't. It is especially useful in models with a lot of variables that have strong relationships with each other.

Chapter 15 Summary

- In time, the UAVs will become autonomous and work without the aid of human direction.
- New technology will improve human life in the field of disaster response.
- Analyzing the data will tell you what kind of algorithm to use to interpret it.
- The more data your algorithms process, the better they'll be at performing their intended function.
- Your data is only as valuable as the results you're able to draw from it.
- There are two general kinds of scopes you can use to interpret data: global and local.

Chapter 16:

The Application of ANN



ANN or Artificial Neural Networks are broad when it comes to their applicability. Today, in this technological era we live in, neural networks are used in a spectrum of industries.

Pattern Recognition

Pattern recognition is probably the category where most of the interesting issues fall. Neural networks have proven to be successful in cracking pattern recognition problems. One of the most popular issues is the automatic handwriting recognition. It is pretty hard to recognize someone's writing using a traditional technique because there are a lot of different sizes, styles, and positions. The backpropagation algorithm has been known to be of great use for pattern recognition. The best part is that even if an application is based on one training algorithm, its architecture can easily be changed to boost performance.

Control

Imagine a new driver backing up a trailer truck. If you have ever tried and failed, or even tried and succeeded in this maneuver, then you already know that it is quite tricky and difficult. On the other hand, an experienced driver who has gone through this process one too many times can now do it with ease.

Now, let's use training a neural network to provide directions to a trailer truck so that it can back up a loading dock, as an example of the control application of ANN. We have the information that describes the position of the truck's cab, the position of the loading docks, the rear's position, as well as the angles that both the trailer and truck make with the loading dock. Now, this neural network is able to learn the best way in which the truck can be steered in order to reach the dock. The neural network can solve the problem in two ways:

1.

The first way *the emulator* learns how to compute a new position for the truck, is by using its current positions and steering angle. At each step, the truck moves a particular distance. With this method, the module can 'feel' how the truck reacts to different steering signals. This emulator contains hidden layers, and it is trained with the help of the backpropagation algorithm.

2.

The second module or *the controller* begins its work only after the emulator is trained. The controller is used to give the right steering signals to get the trailer truck to the dock with its back parallel to it. After each steering signal the controller gives, the emulator determines a new position. This is done for as long as it takes for the truck to get to the dock. Then, errors can be determined and the weights can be easily adjusted.

Speech Production

Neural networks are also used for producing spoken text. And while it may look like an easy task, learning to speak English aloud is not so simple. There are quite a few phonetic pronunciation rules that need to be followed, and given how each letter mainly depends on the context in which it appears, it can get tricky.

Teaching a machine to speak is not that different from teaching a child. At first, all you can hear are a few vowels and consonants and a rather funny, babbling sound. All a baby can say is dada, mama, and some other simple syllables. As the teaching process continues, the baby develops a richer vocabulary and can say up to 50 words, and so on, until the baby is ready to participate in a real conversation.

The same process is used with machines. Let's take NETtalk for example. NETtalk is a neural network created in 1986 that pronounces written English by being shown words as input. The only other requirement that NETtalk needed in order to talk was knowing the correct phonetic transcriptions to use for comparison. The NETtalk was trained with the most common 1000 English words, and it could read new words with few errors. However, NETtalk also learned in stages. First, it had to learn to distinguish consonants and vowels, then it had to learn to recognize the boundary between the words and so on, just like a child.

Speech Recognition

Speech recognition is a part of our daily lives. Think Apple's Siri or Amazon's Alexa. All you have to do is press a button, and voila, you can send a text message without actually typing the text into your phone. However, have you ever given a thought to how this is possible?

In order for the machine to recognize speech, obviously, it has to be trained to do so, which means that it has to have models to use for comparison. Obviously, the easiest way to do so is to feed the machines audio speech recordings, but unfortunately, we are not quite there yet (hopefully this will change in the near future). The

point is that the speech varies in speed. What does that mean? That means that if you say ‘sugar’ and someone else says ‘suuuuuugaaaaaar,’ you will produce different sound files. The second file requires more data, meaning that aligning different-length audio recordings automatically is hard.

In order for the machine to be able to recognize the speech, it goes through many training processes. It must be fed sound waves in the form of pictures and numbers; it must go through sampling, etc. Neural networks have been a real lifesaver for the progress of speech recognition, and today, this is one of their most popular applications.

Business

Neural networks are also applied in a number of different business settings, for instance, mortgage assessment. Moreover, while you may think that mortgage assessment is pretty straightforward and simple, the truth is, it is kind of hard to completely specify the process that experts use to make marginal case related decisions. The whole idea of using a neural network for a mortgage risk assessment is so it can provide much more reliable and consistent evaluation by using past examples.

Trained by professional mortgage evaluators, neural networks can easily screen applications and determine which applicants should be given a loan. The input here is information about the applicant’s employment, dependents, monthly income, etc, and the outcome is a simple ‘reject’ or ‘accept’ response. You can say that this is a simple threshold-type of decision. If the applicants meet the requirements, then they will get a positive response. If not, they will be rejected. It is as simple as that.

Medicine

This application of neural networks is extremely helpful and important. The best example of this application comes from the mid-1980s when it was created. This application was called “Instant

Physician,” and the idea behind it was to be able to train a network to store medical records so that it can offer the right treatment. By being taught different conditions, symptoms, and diagnosis, the network was able to recognize diseases and give a diagnosis.

We know that this opportunity is nothing unfamiliar to us now, but it is thanks to the effect of neural networks that we can simplify our lives in so many different ways. This simplification is something that people in the early 1980s could only dream of.

Chapter 16 Summary

- Artificial Neural Networks are broad when it comes to their applicability.
- Neural networks are also used for producing spoken text.
- Neural networks are also applied in a number of different business settings, for instance, mortgage assessment.
- The application of neural networks is extremely helpful and important in recognizing diseases and giving a diagnosis.

Chapter 17:

Learning in Artificial Neural Networks



The most important part, and the part that makes machines intelligent is the ability of the neural networks to learn. By being fed information from their environment, the artificial neural networks can learn from it in order to improve their performance. Through the iteration of the learning process, the networks learn more about their environment, and the processes of adjustments that are applied to its bias levels and synaptic levels are what makes these networks more knowledgeable.

The learning process implies these events:

- 1.

2. The ANN is stimulated by the environment.
3. This stimulation results in changes in the network's free parameters.
3. Because of the change, the ANN becomes able to respond to the environment in a new way.

There are many different ways in which a machine can learn. Below you will find each of them explained and simplified for you to understand.

Error-Correction Learning

This learning technique, as the name itself suggests, uses errors to direct the process of training. This means that when comparing the output of the system to the output that is desired if errors occur, they are used to learn. This technique is similar to a 'learn from your mistakes' type of learning. By using some algorithms (usually, the backpropagation algorithm), the weights are directly adjusted by the error values.

This learning-from-errors type of learning process is great at preventing errors from happening in each of the training iterations.

Memory-Based Learning

Memory-based learning is a process of storing and retrieving information. All or most of the past experiences are stored in a large memory that consists of classified input – output examples.

The memory-based learning uses algorithms that involve:

- A) A criterion that is used to define the local neighborhood of the test vector
- B) A learning rule that is applied to the local neighborhood's training examples.

Hebbian Learning

Hebbian learning, or the Hebb's rule, is the oldest and most popular learning rule. You can learn more about the Hebb Rule in <http://penta.ufrgs.br/edu/telelab/3/hebbian.htm>

Competitive Learning

In this learning process, the output neurons of the network are in a constant competition over which one will end up being fired. While the Hebb rule allows a couple of neurons to be fired at the same time, here, only one neuron can be active at a given time. This learning rule has three basic elements:

1.

A group of neurons that are the same, except for the distributed weights that can respond differently to the given input patterns.

2.

A limit that is imposed on the strength or weight of each neuron.

3.

A competing mechanism that allows the neuron to compete for the right to be fired and respond to the input.

The simplest form of this learning process is the one where the network has a single layer of output neurons that are connected to the input nodes. The network often includes feedback connections that perform lateral inhibition, meaning that each of the neurons tend to inhibit the one that it is connected to.

Boltzmann Learning

This learning rule, named after Ludwig Boltzmann, is a stochastic algorithm that has its roots in statistical mechanics. Those networks

that are based upon this learning rule are called *Boltzmann machines*.

The neurons in Boltzmann machines have a recurrent structure which means that they are either ‘on’ and can be denoted by +1, or are in an ‘off’ state that is denoted by -1.

The Paradigms

Although there are many different learning rules, there are only three major learning paradigms.

Supervised Learning

Supervised learning is learning with a teacher. In conceptual terms, you may think about how the teacher is the one that has the knowledge of the environment, and the environment is unknown to the network. The teacher, through a process of learning, may use different input methods to teach the ANN the desired output, and to eventually receive it. It is just like teaching a child. You use your built-in knowledge, and through different learning methods, you expect that the child will respond with the desired output and show you that they have actually learned something.

The error-correction learning process is a great example of supervised learning or learning with a teacher. By many input-output examples, the teacher is able to calculate the error, make adjustments, and even change the teaching process in order to get the desired outcome.

Unsupervised Learning

Contrary to supervised learning, unsupervised learning means learning on your own, or without a teacher. Here, the network is on

its own to find the desired output. There is no one to oversee the learning process, meaning that there are only inputs, and it is the network's job to find the pattern within these inputs and generate the right outcome.

This type of learning is used in data mining and also used by recommendation algorithms because of their ability to foresee the preferences of the user based on those preferences of similar users that have been grouped together.

A great way to perform unsupervised learning is by using the competitive learning rule. In fact, the best way to do it is to use an ANN that has two layers. One of the layers is the input layer that is in charge of receiving available data, while the other layer is made of neurons that compete for the chance to 'fire' first and respond to the features of the input data.

Reinforcement Learning

Reinforcement learning, although it is mostly considered to be an unsupervised type of learning, is actually somewhere in between learning with and without a teacher. Here, some feedback is given, but output is still not provided. Reinforcement learning means learning with a reward. Based on how well the system responded, a reward is given. The main goal here is to increase the reward through the process of trial-and-error. Reinforcement learning is a great way of learning because it is how nature works. Why do you think they give puppies a treat every time they find the right spot to eliminate waste or respond to a 'sit' command? The reason for this is because it is much easier to remember those actions for which you were rewarded.

Chapter 17 Summary

- Error-Correction Learning uses errors to direct the process of training.
- Memory-based learning is a process of storing and retrieving information.
- Boltzmann Learning is a stochastic algorithm that has its roots in statistical mechanics.
- Supervised learning uses different input methods to teach the ANN the desired output.
- Unsupervised learning has only inputs, and it is the network's job to find the pattern within these inputs and generate the right outcome.

Chapter 18:

Recursive Neural Network in Theano



```
import sys
import numpy as np
import matplotlib.pyplot as plt
import theano
import theano.tensor as T
from sklearn.utils import shuffle
from util import init_weight, get_ptb_data, display_tree
from datetime import datetime
class RecursiveNN:
    def __init__(self, V, D, K):
        self.V = V
        self.D = D
        self.K = K
    def fit(self, trees, learning_rate=3*10e-4, mu=0.99, reg=10e-5,
            epochs=15, activation=T.nnet.relu, train_inner_nodes=False):
        D = self.D
```

```

V = self.V
K = self.K
self.f = activation
N = len(trees)
We = init_weight(V, D)
Wh = np.random.randn(2, D, D) / np.sqrt(2 + D + D)
bh = np.zeros(D)
Wo = init_weight(D, K)
bo = np.zeros(K)
self.We = theano.shared(We)
self.Wh = theano.shared(Wh)
self.bh = theano.shared(bh)
self.Wo = theano.shared(Wo)
self.bo = theano.shared(bo)
self.params = [self.We, self.Wh, self.bh, self.Wo, self.bo]
words = T.ivector('words')
parents = T.ivector('parents')
relations = T.ivector('relations')
labels = T.ivector('labels')
def recurrence(n, hiddens, words, parents, relations):
    w = words[n]
    # any non-word will have index -1
    hiddens = T.switch(
        T.ge(w, 0),
        T.set_subtensor(hiddens[n], self.We[w]),
        T.set_subtensor(hiddens[n], self.f(hiddens[n] + self.bh)))
)
```

```

r = relations[n] # 0 = is_left, 1 = is_right
p = parents[n] # parent idx
hiddens = T.switch (
    T.ge(p, 0),
    T.set_subtensor(hiddens[p], hiddens[p] +  

hiddens[n].dot(self.Wh[r])),  

    hiddens
)
return hiddens

hiddens = T.zeros((words.shape[0], D))
h, _ = theano.scan(
    fn=recurrence,
    outputs_info=[hiddens],
    n_steps=words.shape[0],
    sequences=T.arange(words.shape[0]),
    non_sequences=[words, parents, relations],
)
# shape of h that is returned by scan is TxTxD
# because hiddens is TxD, and it does the recurrence T times
# technically this stores T times too much data
py_x = T.nnet.softmax(h[-1].dot(self.Wo) + self.bo)
prediction = T.argmax(py_x, axis=1)
rcost = reg*T.mean([(p*p).sum() for p in self.params])
if train_inner_nodes:
    # won't work for binary classification
    cost = -T.mean(T.log(py_x[T.arange(labels.shape[0]), labels])) + rcost
else:

```

```

cost = -T.mean(T.log(py_x[-1, labels[-1]])) + rcost
grads = T.grad(cost, self.params)
dparams = [theano.shared(p.get_value()*0) for p in self.params ]
updates = [
    (p, p + mu*dp - learning_rate*g) for p, dp, g in zip(self.params,
dparams, grads)
]
+ [
    (dp, mu*dp - learning_rate*g) for dp, g in zip(dparams, grads)
]
self.cost_predict_op = theano.function(
    inputs=[words, parents, relations, labels],
    outputs=[cost, prediction],
    allow_input_downcast=True,
)
self.train_op = theano.function(
    inputs=[words, parents, relations, labels],
    outputs=[h, cost, prediction],
    updates=updates
)
costs = []
sequence_indexes = range(N)
if train_inner_nodes:
    n_total = sum(len(words) for words, _, _, _ in trees)
else:
    n_total = N
for i in xrange(epochs):
    t0 = datetime.now()

```

```

sequence_indexes = shuffle(sequence_indexes)
n_correct = 0
cost = 0
it = 0
for j in sequence_indexes :
    words, par, rel, lab = trees[j]
    _, c, p = self.train_op(words, par, rel, lab)
    if np.isnan(c):
        print "Cost is nan! Let's stop here. Why don't you try decreasing the
learning rate?"
        exit()
    cost += c
    if train_inner_nodes:
        n_correct += np.sum(p == lab)
    else:
        n_correct += (p[-1] == lab[-1])
    it += 1
    if it % 1 == 0:
        sys.stdout.write("j/N: %d/%d correct rate so far: %f, cost so far:
%f\r" % (it, N, float(n_correct)/n_total, cost))
        sys.stdout.flush()
    print "i:", i, "cost:", cost, "correct rate:", (float(n_correct)/n_total),
"time for epoch:", (datetime.now() - t0)
    costs.append(cost)
plt.plot(costs)
plt.show()
def score(self, trees, idx2word=None):
    n_total = len(trees)

```

```

n_correct = 0
for words, par, rel, lab in trees:
    _, p = self.cost_predict_op(words, par, rel, lab)
    n_correct += (p[-1] == lab[-1])
print "n_correct:", n_correct, "n_total:", n_total,
return float(n_correct) / n_total

def add_idx_to_tree(tree, current_idx):
    # post-order labeling of tree nodes
    if tree is None:
        return current_idx
    current_idx = add_idx_to_tree(tree.left, current_idx)
    current_idx = add_idx_to_tree(tree.right, current_idx)
    tree.idx = current_idx
    current_idx += 1
    return current_idx

def tree2list(tree, parent_idx, is_binary=False, is_left=False,
is_right=False):
    if tree is None:
        return [], [], [], []
    w = tree.word if tree.word is not None else -1
    if is_left:
        r = 0
    elif is_right:
        r = 1
    else:
        r = -1
    words_left, parents_left, relations_left, labels_left = tree2list(tree.left,
tree.idx, is_binary, is_left=True)
    words_left.append(w)
    parents_left.append(parent_idx)
    relations_left.append(r)
    labels_left.append(tree.label)
    return words_left, parents_left, relations_left, labels_left

```

```

words_right, parents_right, relations_right, labels_right =
tree2list(tree.right, tree.idx, is_binary, is_right=True)

words = words_left + words_right + [w]
parents = parents_left + parents_right + [parent_idx]
relations = relations_left + relations_right + [r]
if is_binary:

    if tree.label > 2:
        label = 1
    elif tree.label < 2:
        label = 0
    else:
        label = -1 # we will eventually filter these out
else:
    label = tree.label

labels = labels_left + labels_right + [label]

return words, parents, relations, labels

def print_sentence(words, idx2word):

    for w in words:
        if w >= 0:
            print idx2word[w],

def main(is_binary=True):
    train, test, word2idx = get_ptb_data()

    for t in train:
        add_idx_to_tree(t, 0)

    train = [tree2list(t, -1, is_binary) for t in train]

    if is_binary:
        train = [t for t in train if t[3][-1] >= 0] # for filtering binary labels

```

```
for t in test:  
    add_idx_to_tree(t, 0)  
test = [tree2list(t, -1, is_binary) for t in test]  
if is_binary:  
    test = [t for t in test if t[3][-1] >= 0] # for filtering binary labels  
train = shuffle(train)  
train = train[:2000]  
n_pos = sum(t[3][-1] for t in train)  
test = shuffle(test )  
test = test[:100]  
V = len(word2idx)  
print "vocab size:", V  
D = 10  
K = 2 if is_binary else 5  
model = RecursiveNN(V, D, K)  
model.fit(train, learning_rate=10e-3, reg=10e-3, mu=0, epochs=30,  
activation=T.tanh, train_inner_nodes=False)  
print "train accuracy:", model.score(train)  
print "test accuracy:", model.score(test)  
if __name__ == '__main__':  
    main()
```

Conclusion

You've learned that neural networks are the key component that allows the machine to learn. Without them, the network is simply reduced down to a very basic and simple vending machine that takes in information and then gives you what you want.

But neural networks are far from using such a simple strategy. You also learned the main concepts behind neural networks, the basic architecture, some of the rules and guidelines that have been set to help the machine to learn. You learned about the technical fields and how they operate, and the many different ways that neural networks can be applied in our day-to-day lives.

No doubt, this book has probably raised more questions than answers but it is our hope that we have at least piqued your interest so that you are eager to learn more. There is much to learn on this subject and sad to say, we have barely just scratched the surface.

Regardless of what you expect to achieve with this knowledge, you have taken the first step in your quest to better understand neural networks and their role in our lives today, tomorrow, and well on into the future.

Book 3: Machine Learning with Python

*An Advanced Guide to Go Deep into
Artificial Intelligence. Tools, Tips and Tricks
for Going into Data Science and Data
Analysis using Python and TensorFlow*

To Enza and Angelomaria

© Copyright 2019 - All rights reserved.

The content contained within this book may not be reproduced, duplicated or transmitted without direct written permission from the author or the publisher.

Under no circumstances will any blame or legal responsibility be held against the publisher, or author, for any damages, reparation, or monetary loss due to the information contained within this book. Either directly or indirectly.

Legal Notice:

This book is copyright protected. This book is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part, or the content within this book, without the consent of the author or publisher.

Disclaimer Notice:

Please note the information contained within this document is for educational and entertainment purposes only. All effort has been executed to present accurate, up to date, and reliable, complete information. No warranties of any kind are declared or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice. The content within this book has been derived from various sources. Please consult a licensed professional before attempting any techniques outlined in this book.

By reading this document, the reader agrees that under no circumstances is the author responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to, — errors, omissions, or inaccuracies.

Introduction

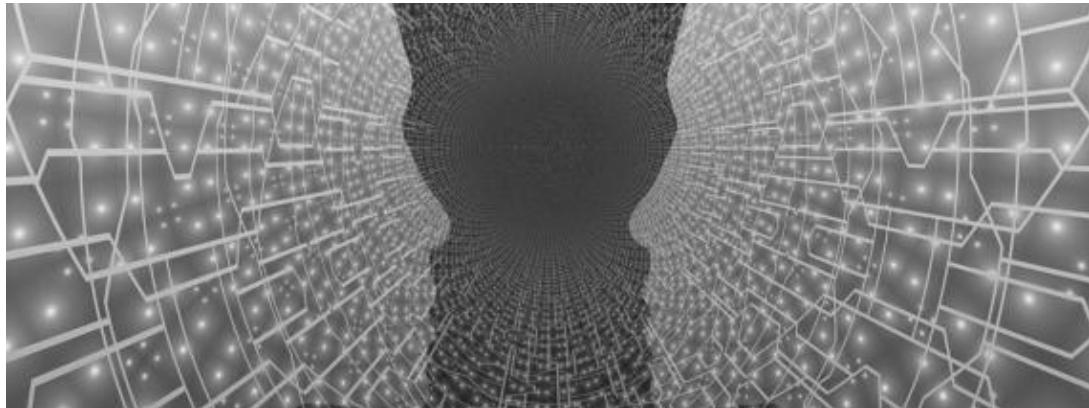
Machine learning permits models for training on datasets before being used. Some machine learning models are always online and continuously adapt as new information is ingested. On the other hand, other models, called offline machine learning models, are derived from machine learning algorithms but, once deployed, do not change. This reiterative method of

on-line models results in an improvement in the kinds of associations that are made between data components. Due to their quality and size, these associations and patterns could have easily been disregarded by human observation. After a model has been prepared, these models can be used in real time to learn from data.

Also, complex algorithms may be automatically adjusted based on rapid changes in variables, such as customer sentiment metrics, sensor data, time, and weather data. For instance, inferences can be made from a machine learning model — if the weather changes quickly, a model that predicts model weather can predict a tornado, and a warning siren can be triggered. The developments in accuracy are a result of the training process and automation that is part of machine learning. Online machine learning algorithms continuously refine the models by training the system to adapt to changing patterns and associations in the data and continue processing new data in near real time.

Chapter 1:

Machine Learning - Automation Within Learning



Machine learning and the requirement for it

Machine learning is a subfield of Artificial Intelligence, in which a PC framework is nourished with algorithms that are intended to analyze and decipher various sorts of data all alone. These learning algorithms acquire the dissecting capacity when they are prepared for a similar utilizing test data.

It proves to be useful when the measure of data to be analyzed is considerable and out of personal points of confinement. It very well may be utilized to land at significant ends and make meaningful choices.

Some significant fields where it is being executed:

Malignant growth treatment

Chemotherapy, which is utilized in executing harmful cells represents the threat of slaughtering even the sound cells in the human body. A successful option in contrast to chemotherapy is radiotherapy, which makes the utilization of machine learning algorithms to make the correct refinement between cells.

Automated medical procedure

Utilizing this innovation, hazard-free activities can be performed in parts of the human body where the spaces are thin, and the danger of a specialist destroying the medical procedure is high. An automated therapeutic process is prepared to utilize machine learning algorithms.

Money

It is utilized to distinguish fake bank exchanges inside seconds for which a human would take hours to figure it out.

The utility of Machine learning is perpetual and can be utilized in various fields.

What does one realize in Machine Learning?

Regulated algorithms

Regulated learning is the learning wherein input and yield are known, and you compose an algorithm to get familiar with the mapping procedure or connection between them.

Most algorithms depend on supervised learning.

Unsupervised algorithms

In unsupervised learning, the yield is obscure, and the algorithms must be written such that makes them independent in deciding the structure and dissemination of data.

Requirements

Software engineering undergraduates and different undergraduates with a designing foundation think that it is simpler to learn Machine learning. Be that as it may, anyone with useful or possibly essential information in the accompanying domains can master the subject at beginner level.

Basics of programming

Essentials of programming incorporate a good grasp of basic programming, data structures, and algorithms.

Likelihood and measurements

Major likelihood points like sayings and guidelines, Bayes hypothesis, relapse, and so on must be known.

Learning on measurable themes like mean, middle, mode, difference, and appropriations like normal, Poisson, binomial, and so forth is required.

Direct Algebra

Direct variable-based math is the portrayal of straight articulations as matrices and vector spaces. For this, one must be all around educated about subjects like matrices, complex numbers, and polynomial conditions.

NOTE: These requirements are for beginners.

Occupation prospects in Machine learning

Attributable to its vast applications and use in the present day and ad-libbed innovation, demand for its experts is expanding step by step, and it could

never under any circumstances go out of the pattern.

An expert can secure positions in the accompanying fields:

- Machine learning engineer
- Data engineer
- Data investigator
- Data researcher

Chapter 1 Summary

- A subfield of Artificial Intelligence, Machine learning aims to equip PC frameworks with the ability to analyze and decipher different types, quantities, and frequencies of data without the input of a human.
- Machine learning is broadly applied in automated medical procedures, malignant growth treatment, and money.
- The application of machine learning occurs in the form of regulated algorithms or unsupervised ones.

Chapter 2:

Building Blocks Of Machine Learning



In this chapter, we explore the building blocks of machine learning that will help you have a big picture view of ML. By the end of the section, you should be in a position to understand the finite elements of machine learning that can help you map ideas into hardware and software primitives. Let's dive in together.

A typical machine learning problem

Suppose you would like to program a computer that learns how to filter the spam e-mails. A simple solution to this problem can be made to resemble the learning process that usually takes place in rats and how they avoid

poisonous foods. You simply need to program the computer to memorize all its previous emails that were labeled as spam emails by human users.

When any new e-mail arrives, the computer will search for it in the set of all the spam emails. When it finds one of them, it will automatically be labeled as spam and trashed. Otherwise, it will be stored in the user's inbox folder. While the preceding example of learning by memorization is sometimes worthwhile, it lacks an integral component of any learning process—the ability to label the hidden e-mail messages.

Any successful learner should start learning from specific examples to the broader generalization in what is called inductive reasoning. To achieve the generalization in the spam-filtering process, the spam filter program can scan the hitherto seen e-mails, and mine a set of words whose appearance in the email is pinpointing to spam.

Next, when any new e-mail arrives, the program can now check whether one of the suspicious words is appearing in it and foretell its label accordingly. Such a system can now correctly predict any unseen email and label it as spam or store it in the inbox. However, inductive reasoning isn't the only approach that machine-learning systems should take.

What will distinguish learning mechanisms in spam filtering programs that just result in new emails being labeled as spam or stored in the box from a useful learning process?

Answering this question is vital to the development of machine learning systems. While human beings can depend on common sense to filter out the random spam messages, computers don't use common sense. Therefore, we must define refined principles that protect the program from reaching illogical and meaningless conclusions. The development of these principles is what underpins machine learning.

In other words, the combination of prior knowledge, biasing the learning task, is necessary for the success of learning algorithms. The development of tools that can express the domain expertise, translate it into a learning bias, and quantify its effect on the success of learning are integral components of any successful machine learning system.

Components of a machine learning system

Any machine learning model must incorporate the following components:

- Learner's input
- Learner's output
- Simple data generalization model
- Measures of success with the model

Let us now dive in and explore these critical components of the learning process.

1: Learner's input

At the outset, the learner has to access the following:

- Domain set. The domain is an arbitrary set say X that has the objects that we want to label. In statistics, the domain points can be represented using a vector of features such as the email's sender and content.
- The label set. In the spam-filtering learning problem, we can restrict the label set to a two-element set where Y can represent being spam and X stands for not spam.
- Training data. The training data S is the set $((x_1; y_1); \dots; (x_m; y_m))$ that has a finite sequence of pairs in the $X \times Y$ matrix. This forms the input that the learner will access to find out whether the email is spam or not.

2: Learner's output

The learner's output can be viewed as a function that produces the result to be used by the ML system. It usually takes the form:

$h : X \rightarrow Y$

The above function which is often called predictor, hypothesis, or even classifier is used to predict the label of the new domain points. In the spam-filtering problem example, it is a general rule that the learner will employ to foretell whether future emails he/she examines will be spam or not.

3: Data generalization model

The data generalization model is used to produce the training data. For such data to be produced, all instances such as the emails will be generated using a probability distribution that represents the environment. Assuming the probability distribution over the random variable X is D . Then, for learning to take place, we have to assume that all the labels will assume any arbitrary probability distribution over D .

4: Measures of success

The measures of success are used as performance indicators for the ML system. In other words, is the ML model successful in predicting the outcomes of the learning process? When determining the performance indicators of ML models, we must always assess the learning model to determine whether it's meeting ML objectives or not. Consider the following examples that illustrate machine learning in action:

Problem: A robot driving learning experience

The following are the components of a robot driving learning problem:

- Learner's Output: Driving on any public 4-lane highways using the vision sensors
- Learner's Input: A series of images and other steering commands that are recorded while observing the human driver.
- Data Generalization Model: An average distance that is traveled before any error (as judged by the human overseer) should inform the process.
- A measure of Success: The percentage of successful road tests taken.

Problem: A handwriting recognition learning task

The following are the components of a handwriting recognition learning problem:

- Learner's Output: recognizing and classifying any handwritten words or phrases within images
- Learner's Input: a database of handwritten phrases and words.
- Data Generalization Model: The system should capture and learn all the words that are correctly classified.
- A measure of Success: The percentage of words or phrases that are successfully classified.

Chapter 2 Summary

- The design of machine learning is influenced by a specific task, problem, type of targeted data, or other expected externalities.
- The machine learning process is dependent on prior knowledge to bias the learning task.
- Typical machine learning models comprise learner's input and output, a simple data generalization model, and a model to measure success.

Chapter 3:

Understanding The Syntax Of Python



Having examined how Python runs and executes its program, it is critical to examine the syntax of Python in order to give sufficient information to learners. For the most part, syntax implies the rules of the acceptable ways of arranging words for a particular purpose contained in the series of words. In any case, Python syntax describes how the arrangement of words that characterizes how human users and the framework ought to be composed and interpreted as a Python program. Since you have composed and run

your program in Python, being comfortable with its syntax is an absolute necessity.

The principal thing to know first is the keywords; these are words that ought not to fill in as factors, code identifier, steady and function name. The Python keywords are constantly held solely. It is imperative to ace the language structure of Python so as to spare oneself of the up and coming slip-ups, mistakes and even disappointment. You should take the accompanying keywords in all respects cautiously if you would prefer not to keep running into errors when you are executing your program:

- and state
- break class
- continue def
- del elif
- else with the exception of
- exec at last
- for from
- global if
- import in
- is lambda
- not or
- pass print
- raise return
- try while
- with yield

The Quotations used in Python Punctuation

In the punctuation of Python, there are bunches of directions done by using the quote. As a matter of fact, Python permits the utilization of quotes to show string literals. Regardless of whether you are using single, twofold, or triple statements, you should begin and end the string with a similar sort to guarantee the program is executed as determined. Note that you'd use the triple statements just when your string keeps running over a few lines.

Declarations Writing in Python

The arrangement of directions given to the Python translator to execute and run is called an explanation Statement. For instance, when you dole out an incentive to a variable, state my variable = "feline", you've recently owned a task expression. Be that as it may, the linguistic structure of a task doesn't live in to what extent or with the citation as task explanation may likewise be as short as c = 3. Different sorts of statements in Python incorporate; if explanations, for statements, while declarations, and some more.

The Language Structure of Multi-line Declarations

As examined, declarations are known as guidelines. These directions may range more than a few lines; henceforth multiline declaration. While composing code, you'll have to break a long declaration over various lines. To do this, you may wrap the statement inside enclosures, supports, and sections. In view of styles of composing, this procedure is the favored style for taking care of multi-line statements. Then again, there is an approach to wrap different lines by using an oblique punctuation line () toward the finish of each line to show line continuation. As you compose your own code, ideal comprehension of the multi-line grammar turns out to be simple.

The Sentence Structure of Indentation

For the most part, space is used to indicate squares of code. This is like what is found in other programming dialects like C, C++, and so on as support. Python is arranged in spaces.

Using Python, squares of codes are arranged by space not by style or inclination but rather as an unbending language prerequisite. Numerous developers have finished up this guideline combined with others makes Python codes increasingly comprehensible and reasonable. Based on distinguishing proof and structure, a square of code can be effectively recognized when you take a gander at a Python program as they begin a similar separation to one side. If it must be all the more profoundly settled, you can basically indent another square further to one side. For instance, here is a section of a program characterizing the expense of shoes:

```
def cost of shoes ($5):
    cost = 5 * $
    if 3 >= 15:
        cost - = 4
    elif days >= 2:
        cost - = 10
    return cost
```

It is important to take note of that you should dependably ensure that the indent space is steady inside a square. When you use IDLE and different IDEs to include your codes, Python naturally gives space on the consequent line when you enter an explanation that requires space. Space, by the principles of Python, is proportionate to 4 spaces to one side; keep at it.

After the formation of space, you might need to realize how to end it. Simply, the end of each space is the end of the square. The third and last syntactic part that Python expels which could appear to be the one for most learners of other programming C-like languages is that there is no need of composing anything irrelevant in your code to grammatically access the start and end of a bracketed square of code. You don't need to incorporate

start/end, at that point/end if, or surround the bracket square, as you do in C-like language like:

```
if (x > y) {  
    x = 1;  
    y = 2;  
}
```

In Python, rather, you reliably indent every one of the codes in a given single settled square giving it a similar separation to one side. Note that Python uses the codes' physical space to learn where the square begins and stops:

If a > b:

```
a = 1  
b = 2
```

Space implies, now, the clear whitespace around the two settled declarations. Python couldn't care less to what degree you indent (you may use either spaces or tabs), or how you indent; you have the decision of using any number of spaces or tabs. To state, the space of one settled square can be very surprising from that of another. The general language structure decides is just that the majority of the space explanations must be indented with a similar separation to one side. This is in such a case that else, you will get a grammar mistake, and your code won't keep running until you fix its space to be predictable. Check everything before experiencing the execution.

The Language Structure of Comments

When composing a program, now and then, you'll want to put a few notes just inside your code to give a depiction of what that statement does; such a thing is known as a remark. A remark is most valuable when you need to audit or return to your program for deficiency or change; it is simpler to follow. Also, for different developers who wish to go over your source code, the remark will make it simpler. How you compose and structure remark inside your program is by beginning the line with a hash (#) image.

To the Python interpreter, the hash image advises the Python interpreter to disregard the remark when running your code. If your remark is on a multi-line, you can use a hash image toward the start of each line. On the other hand, you can likewise wrap multi-line remarks with triple statements.

The Language Structure of Python Identifiers

Inside the Python program, Python Identifiers are names given to the class, module, function, variable, or different articles. This is any substance you'll be using in Python to compose codes ought to be suitably named or given right recognizable proof as they will shape some portion of your program. Here are Python naming shows that you ought to know about:

An identifier can be a mix of capitalized letters, lowercase letters, underscores, and digits (0-9). Consequently, coming up next are substantial identifiers: theMovement, my_movement, move_1, and print_what_is_world.

Coming up next are exceptional tips on identifiers in Python:

- Within the identifiers, unique characters, for example, %, @, and \$ are not allowed.
- Note that 3life isn't legitimate rather life3 is on the grounds that an identifier ought not start with a number.
- Generally, Python is a case-sensitive language. This is a commonplace of identifiers and this is the reason the word Happy and glad are two particular identifiers in Python.
- Python keywords can't be used as identifiers.
- When making a class, the Class identifiers must start with a capitalized letter, yet the remainder of the identifiers lowercase.
- Separation of different words in your identifier is by an underscore.
- You ought to dependably pick identifiers that will sound good to you even after a long hole. This is the reason it is anything but difficult to set your variable to c = 2.

- It is fitting for the future reason that thinks that it's more you use a more extended yet progressively important variable name, for example, consider = 2 this becomes easier at any point of confusion.

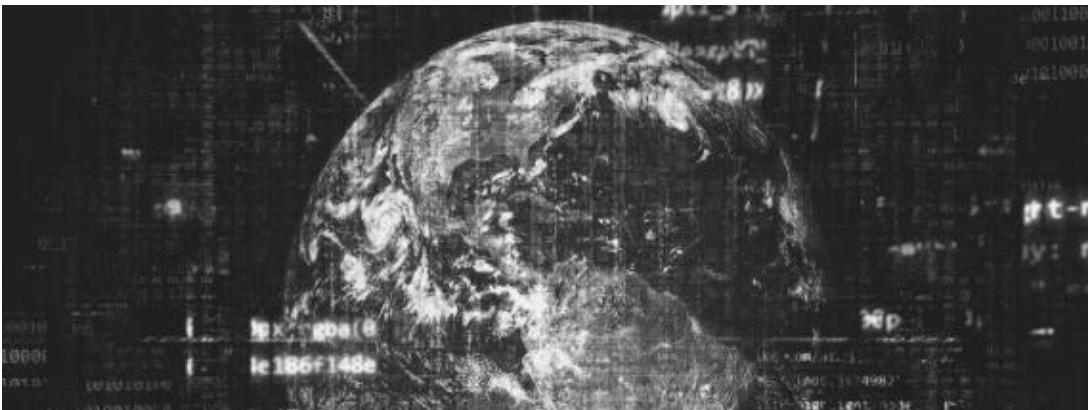
Taking everything into account, the syntax of Python has been managed by the 'modules' where you'd need them. This is on the grounds that you will get everything about it independently rather than assembling it. Remember to run the projects as you learn them.

Chapter 3 Summary

- Python adopts a multi-line declarations language structure because its (language) directions range more than a few lines.
- In Python's sentence structure of indentation, space is utilized to square codes. As a result, Python is organized in spaces.
- Python's syntax is managed by the 'modules.'
- The arrangement of directions given to the Python translator to execute and run is called explanation statement.

Chapter 4:

Variables and Data Types Of Python



The real deal about Python programming is surfacing now. In this chapter, we will examine variables, their definition, syntax, etc. and the types of data found in Python.

Within the Python is a container; more like a memory card that can store values; called the variables. It can be accessed or changed depending on the kind of information stored therein. Variables are like ways of pinpointing a specific location used by a program while coding. In a way, variables are used to track the particular place programs coded are stored for further usage like exporting to other platforms, merging, among others. For Python, variables can instruct (give statements) the computer to save or retrieve data to and from this same memory location.

In dealing with variables and data types, Python works significantly different from other related languages like Java, C, etc. This is chiefly

because while they declare and bind a variable to a specific data type, Python binds with different data –it can only store a unique data type. To put in simple terms, if the variable to be stored is of integer type, you will only be able to save integers in that variable when running your program. In short, in dealing with variables, there is no rigid way of handling them in Python; rather, you wish to use a variable, think of a name, declare it through value assignment which can be changed as well during the execution of the program if need be.

How do you declare a variable through a value assignment? To answer this, examine the illustration given below;

If my_variable = 20

Note that you are equating the variable my_variable as 20 in actuality rather than the variable is just set to 20

If you want to increase the value of the variable, enter this statement on the command line:

```
>>>my_variable = my_variable + 5
```

In order to know the way Python will respond to this command, invoke the print command with: >>>print(my_variable)

This will be the result on the next line:

25

If you wish to use my_variable to store a literal string “violet”, simply set the variable to

“violet” in this way:

```
>>>my_variable = "violet"
```

In order to know the current store in my_variable, use the print command:

```
>>>print(my_variable)
```

On the next line, you’ll see:

Violet

These are just simple ways of handling variables in Python, make sure you follow practically.

At this point, the journey to variables will be on hold as we turn to the data types found in Python programming. Python has different data types in order to enhance the importance of it to application developers and programmers as enough effective data are supplied. The data types are numbers, date, Strings, time, Booleans, and lists.

The Numbers Data Types

Python can differentiate numeric types without you declaring them; this is one of the greatest advantages of using Python. With its in-built functions, Python knows for itself one data from the other when you are running your statement. These in-built functions for numeric values are four, namely: floating, complex, integer, and long integer numbers. It is pertinent to note that the type of Python you are using will determine the number of active in-built numeric functions. For Python three, however, only three (floating, integer, and complex) types are supported.

The Floating numbers

These are real numbers. They are known as floats too. Also known as floats, floating-point numbers signify real numbers. Significantly, when you want to write floats, put a decimal in them to differentiate them from fractional numbers. In another way, you can write floats using scientific notations; upper and lower cases; where the letter ‘e’ will be equivalent to the power of 10th. The illustration is:

```
>>>7.4e2
```

```
7400.0
```

```
>>>7.4e2
```

You can try it with other number sequences too to enhance flexibility with floats.

The Integer (int) numbers

Integers, unlike floats, are not written with decimal points. They are whole positive or negative numbers. Integers in Python are numerous but few will be given below as a guide to how others will be treated;

Base 2 integers

In order to write binary integers, use either '0b' or '0B' as prefix (it is zero and not 'o'). As an illustration, see:

```
>>> a = 0b3200
```

```
>>> print(a)
```

```
3200
```

Regular integers

They are just normal numbers whether positive or negative polar. Examples are 243, -110, 3, etc.

Base 16 integers

They are also known as hexadecimal. In order to write the base 16 integers, use the prefix '0X' or '0x' (zero eks). To illustrate see below:

```
>>>hex_lit = 0xA0B
```

```
>>>print(hex_lit)
```

```
238
```

Base 8

This like others, too, is represented by using '0o' or '0O' as a prefix too. Example:

```
>>>c = 0O29  
>>>print(c)  
29
```

Complex numbers

They are like the real numbers in that they are real numbers, though they are always in pairs. Additionally, complex numbers are also sets of imaginary numbers. In their combination, they take the form of a float and real number in this way: ‘c + bJ’ where ‘c’ is a float, ‘b’ a float too and ‘J’ square root of the imaginary number. See the examples below:

```
>>>x = 5 + 6j  
>>>y = 8 - 2j  
>>>z = x + y  
>>>print(c)  
(6 + 3j)
```

Note that complex numbers are restricted in usage in Python.

The Strings Data Types

Strings are special data as they are Unicode characters with letters, symbols and numbers combinations. In their definition, while coding, enclose strings in ‘matching single or double quotations’. Check the examples below:

```
>>>stringA = “The best student is Python.”  
>>>stringB = “The best student is Python.”
```

Peradventure you are dealing with a literal string enclosed in single quotes which contains a single quote as well, place a backslash (\) before the single quote within the string to escape the character. For example:

```
>>> stringC = 'It is good to be smart.'
```

To print stringC:

```
>>> print(stringC)
```

It is good to be smart.

If the quote had been enclosed in the normal double quote, the coding would have been straightforward in this manner:

```
>>>stringC = "It is good to be smart"
```

In the same vein, whenever your quote is enclosed in another double quote, place a backslash before the embedded quote. Check this example:

```
>>>doc= "The programmer reiterated: \"You will always choose Python programming language as the best among others.\""
```

```
>>> print(doc)
```

The programmer reiterated: “You will always choose Python programming language as the best among others”

There are ways in which strings treat indexed or subscripted numbers. Bear in mind that Python indexing starts from 0 (zero) instead of 1. Therefore, the first character of any string must start from zero.

To illustrate how string indexing works in Python, define the string "Hello Python" on the command line:

```
>>>s = "Hello everyone, it is Python"
```

This is how Python would index the string:

-12 -11 -10 -9 -8 -6 -5 -4 -3 -2 -1

Hello everyone, it is Python

(write from 0- infinity)

In order to check for the first character of the string, type and enter the variable name “s” and the index 0 within square brackets like this:

```
>>>s[0]
```

You'll get this output:

‘H’

Note that it is easier to check for this because you know that its index number is zero. Interestingly, you do not have this privilege when you want to access the last character on the same string. How do you do it then?

In order to check for the last character, use this expression:

```
>>>s[len(s)-1]
```

You'll get the output:

‘n’

The expression introduces you to the LEN function. There is actually an easier way to

access the last item on the string:

```
>>>s[-1]
```

‘n’

To access the penultimate character:

```
>>>s[-2]
```

‘o’

Apart from using the index, you can use other functions and mathematical operators on a string.

Date and Time Data Types

The fact that a good number of applications works effectively with date and time information is true of Python programming. To use the function of time and date, use this command; `datetime.now()`. This will generate the current date and time as it communicates the in-built Python code to execute the function.

You can access time and date right from Python by encoding the following on the command prompt:

```
>>> from datetime import datetime  
>>> datetime.now()  
datetime.datetime(2019, 4, 20, 4, 17, 19, 60352)
```

You will notice that the format of the numbers is not really readable and you'd love to work with something readable, therefore, use the command ‘`strftime`’ to make it possible. Check the format given below:

```
>>>from time import strftime  
>>> strftime("%Y-%m-%d %H:%M:%S")  
'2019-04-20 05:35:03'
```

Note that the ‘`strftime`’ commands acronym are: Y, (year), M (month), D (day), M (month), and S (seconds).

Chapter 4 Summary

- Variables are values stored in Python's container (memory card).
- Python operates distinctly to related languages like C, Java, etc. in the way it deals with variables.
- The data types stored or found in Python programming affect the way variables are handled and behave. Although there is no rigid approach in handling data in Python.
- Python has the ability to learn new numbers without declaring them. Some of the data types typically used in Python include floating numbers, base 2 integers, integer (int) numbers, regular integers, base 8, and complex numbers.

Chapter 5:

Manipulation of Data in Python



In this chapter we will look at how we can use the NumPy and Pandas libraries to manipulate data in a data set.

NumPy

```
#Load the library and check its version, just to make sure we aren't using an older version
```

```
import numpy as np
```

```
np.__version__
```

```
'1.12.1'
```

```
#Create a list comprising numbers from 0 to 9
```

```
L = list(range(10))
```

```
#Converting integers to string - this style of handling lists is known as list comprehension.
```

```
#List comprehension offers in a versatile way to handle list manipulations tasks easily.
```

We'll learn about them in future tutorials. Here's an example.

```
[str(c) for c in L]
```

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```

```
[type(item) for item in L]
```

```
[int, int, int, int, int, int, int, int, int]
```

Creating Arrays

An array is a homogeneous data type, in the sense that it can only hold variables of the same data type. This holds true for arrays in NumPy as well.

```
#Creating arrays
```

```
np.zeros(10, dtype='int')
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
#Creating a 3 row x 5 column matrix
```

```
np.ones((3,5), dtype=float)
```

```
array([[ 1.,  1.,  1.,  1.,  1.],
```

```
      1.,  1.,  1.,  1.],
```

```
      1.,  1.,  1.,  1.,  1.]])
```

```
#Creating a matrix with a predefined value np.full((3,5),1.23)
```

```
array([[ 1.23, 1.23, 1.23, 1.23, 1.23],
```

```
      1.23, 1.23, 1.23, 1.23, 1.23],
```

```
      1.23, 1.23, 1.23, 1.23, 1.23]])
```

```
#Create an array with a set sequence np.arange(0, 20, 2)
```

```
array([0, 2, 4, 6, 8, 10, 12, 14, 16, 18])
```

```
#Create an array of even space between the given range of values  
np.linspace(0, 1, 5)  
array([ 0., 0.25, 0.5 , 0.75, 1.])  
  
#Create a 3x3 array with mean 0 and standard deviation 1 in a given  
dimension np.random.normal(0, 1, (3,3))  
array([[ 0.72432142, -0.90024075, 0.27363808],  
       0.88426129,  1.45096856, -1.03547109], [-0.42930994, -1.02284441,  
       -1.59753603]]) #Create an identity matrix  
np.eye(3) array([[ 1., 0., 0.],  
       0., 1., 0.],  
       0., 0., 1.]))  
  
#Set a random seed np.random.seed(0)  
x1 = np.random.randint(10, size=6) #One dimension  
x2 = np.random.randint(10, size=(3,4)) #Two dimension  
x3 = np.random.randint(10, size=(3,4,5)) #Three dimension  
print("x3 ndim:", x3.ndim)  
print("x3 shape:", x3.shape)  
print("x3 size: ", x3.size)  
('x3 ndim:', 3)  
('x3 shape:', (3, 4, 5))  
('x3 size: ', 60)
```

Array Indexing

If you are familiar with programming languages, you will be aware that the indexing in an array always begins at zero.

```
x1 = np.array([4, 3, 4, 4, 8, 4])  
x1
```

```
array([4, 3, 4, 4, 8, 4])
#Assess value to index zero
x1[0]
4
#Assess fifth value
x1[4]
8
#Get the last value
x1[-1]
4
#Get the second last value
x1[-2]
8
#In a multidimensional array, we need to specify row and column index x2
array([[3, 7, 5, 5],
       [0, 1, 5, 9],
       [3, 0, 5, 0]])
#1st row and 2nd column value
x2[2,3]
0
#3rd row and last value from the 3rd column
x2[2,-1]
0
#Replace value at 0,0 index
x2[0,0] = 12
x2
array([[12, 7, 5, 5],
```

```
0, 1, 5, 9],  
3, 0, 5, 0]])
```

Array Slicing

You can slice an array to access a specific element or a range of elements within an array.

```
x = np.arange(10)  
  
x  
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])  
  
#From start to 4th position  
x[:5]  
array([0, 1, 2, 3, 4])  
  
#From 4th position to end  
x[4:]  
array([4, 5, 6, 7, 8, 9])  
  
#From 4th to 6th position  
x[4:7]  
array([4, 5, 6])  
  
#Return elements at even place  
x[ :: 2]  
array([0, 2, 4, 6, 8])  
  
#Return elements from first position step by two  
x[1::2]  
array([1, 3, 5, 7, 9])  
  
#Reverse the array  
x[::-1]  
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

Array Concatenation

It helps to combine multiple arrays to perform complex operations. You will not have to type the elements in the different arrays, but can instead concatenate those arrays to handle these complex operations with ease.

#You can concatenate two or more arrays at once.

```
x = np.array([1, 2, 3])  
y = np.array([3, 2, 1])  
z = [21,21,21]  
np.concatenate([x, y,z])  
array([ 1, 2, 3, 3, 2, 1, 21, 21, 21])
```

#You can also use this function to create 2-dimensional arrays.

```
grid = np.array([[1,2,3],[4,5,6]])  
np.concatenate([grid,grid])  
array([[1, 2, 3],  
[4, 5, 6],  
[1, 2, 3],  
[4, 5, 6]])
```

#Using its axis parameter, you can define row-wise or column-wise matrix

```
np.concatenate([grid,grid],axis=1)  
array([[1, 2, 3, 1, 2, 3],  
[4, 5, 6, 4, 5, 6]])
```

In the above code, we have used the concatenation function on those arrays that have variables of the same data type and the same dimensions. So, what if you need to combine a one-dimensional array with a two-dimensional array? You can use the np.vstack or the np.hstack functions in such instances.

```
x = np.array([3,4,5])  
grid = np.array([[1,2,3],[17,18,19]])
```

```

np.vstack([x,grid])
array([[ 3,  4,  5],
       [ 1,  2,  3], [17, 18, 19]])

#Similarly, you can add an array using np.hstack z = np.array([[9],[9]])
np.hstack([grid,z]) array([[ 1,  2,  3,  9], [17, 18, 19, 9]])

It is always a good idea to use a predefined position or condition to split an
array.

x = np.arange(10)

x
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]) x1,x2,x3 = np.split(x,[3,6]) print x1,x2,x3
[0 1 2] [3 4 5] [6 7 8 9]

grid = np.arange(16).reshape((4,4)) grid
upper,lower = np.vsplit(grid,[2]) print (upper, lower)
(array([[0, 1, 2, 3],
       [4, 5, 6, 7]]), array([[ 8,  9, 10, 11], [12, 13, 14, 15]]))

```

The NumPy directory gives you access to numerous mathematical functions apart from the functions that have been used in the examples above. Some of these functions include sum, divide, abs, multiple, mod, power, log, sin, tan, cos, mean, min, max, var, etc. You can use these functions to perform numerous arithmetic calculations. To learn more about these functions, you should refer to the NumPy documentation to learn more about the functions. Let us now take a look at how you can manipulate data using the Pandas library. Ensure that you look at every line in the code carefully before you manipulate the data.

Let's start with Pandas

#Load library - pd is just an alias. I used pd because it's short and literally abbreviates pandas.

#You can use any name as an alias.

```
import pandas as pd

#Create a DataFrame - dictionary is used here where keys get converted to
column names and values to row values.

data = pd.DataFrame({'Country':
['Russia','Colombia','Chile','Ecuador','Nigeria'],
Rank':[121,40,100,130,11]})

data
CountryRank
0Russia121
1Colombia40
2Chile100
3Ecuador130
4Nigeria11

#We can do a quick analysis of any data set using:
data.describe()
Rank
count5.000000
mean80.400000
std52.300096
min11.000000
25%40.000000
50%100.000000
75%121.000000
max130.000000
```

You can obtain a summary of the statistics of only the integer and double variables using the `describe()` method. If you want to obtain all the information available about the data set, you should use the function named `info()`.

```
#Among other things, it shows the data set has 5 rows and 2 columns with  
their respective names. data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 5 entries, 0 to 4
```

```
Data columns (total 2 columns):
```

```
Country 5 non-null object
```

```
Rank 5 non-null int64
```

```
dtypes: int64(1), object(1)
```

```
memory usage: 152.0+ bytes
```

```
#Let's create another data frame
```

```
data = pd.DataFrame({'group':['a', 'a', 'a', 'b','b', 'b', 'c', 'c','c'],'ounces':[4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

```
data
```

```
groupounces
```

```
0a4.0
```

```
1a3.0
```

```
2a12.0
```

```
3b6.0
```

```
4b7.5
```

```
5b8.0
```

```
6c3.0
```

```
7c5.0
```

```
8c6.0
```

```
#Let's sort the data frame by ounces - inplace = True will make changes to  
the data data.sort_values(by=['ounces'],ascending=True,inplace=False)  
groupounces
```

```
1a3.0
```

```
6c3.0
```

```
0a4.0  
7c5.0  
3b6.0  
8c6.0  
4b7.5  
5b8.0  
2a12.0
```

The data can now be sorted using numerous columns.

```
data.sort_values(by=['group','ounces'],ascending=[True,False],inplace=False)
```

```
groupounces  
2a12.0  
0a4.0  
1a3.0  
5b8.0  
4b7.5  
3b6.0  
8c6.0  
7c5.0  
6c3.0
```

Most data sets have duplicate rows and columns, and these duplicate rows are called noise. It is for this reason that you should always remove all the inconsistencies in the data set before you feed the model with the data set. Let us look at an alternative way to remove the duplicates in the data set.

```
#Create another data with duplicate rows
```

```
data = pd.DataFrame({'k1':['one']*3 + ['two']*4, 'k2':[3,2,1,3,3,4,4]})  
data  
k1k2
```

```
0one3
1one2
2one1
3two3
4two3
5two4
6two4
#Sort values
data.sort_values(by='k2')
k1k2
2one1
1one2
0one3
3two3
4two3
5two4
6two4
#Remove duplicates - ta da!
data.drop_duplicates()
k1k2
0one3
1one2
2one1
3two3
5two4
```

In the example above, the values present in the rows and columns have been matched to remove the duplicates in the data set. The duplicates can

also be removed using specific columns as parameters. Let us look at how the duplicates in Column K can be removed.

```
data.drop_duplicates(subset='k1')
```

k1k2

0one3

3two3

Let us now understand how the data can be categorized based on some predefined rules or criteria. This will often happen when you work on processing data, especially that data which needs to be categorized. For example, if you have a column with the names of countries in it, and you want to split the countries into separate columns based on their continents. To do this you will need to create a new variable. The code below will help you achieve this with ease.

```
data = pd.DataFrame({'food': ['bacon', 'pulled pork', 'bacon',  
'Pastrami','corned beef', 'Bacon', 'pastrami', 'honey ham','nova lox'],
```

```
'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

data

foodounces

0bacon4.0

1pulled pork3.0

2bacon12.0

3Pastrami6.0

4corned beef7.5

5Bacon8.0

6pastrami3.0

7honey ham5.0

8nova lox6.0

Let us now look at how we can create a new variable that will help the model predict which animal will be a source of food for another animal. For the model to do this, we should first map the food to the animals by using

the dictionary. The map function can then be used to pull the values from the dictionary. The code below will help us achieve this.

```
meat_to_animal = {  
    'bacon': 'pig',  
    'pulled pork': 'pig',  
    'pastrami': 'cow',  
    'corned beef': 'cow',  
    'honey ham': 'pig',  
    'nova lox': 'salmon'  
}  
  
def meat_2_animal(series):  
    if series['food'] == 'bacon':  
        return 'pig'  
    elif series['food'] == 'pulled pork':  
        return 'pig'  
    elif series['food'] == 'pastrami':  
        return 'cow'  
    elif series['food'] == 'corned beef':  
        return 'cow'  
    elif series['food'] == 'honey ham':  
        return 'pig'  
    else:  
        return 'salmon'  
  
#Create a new variable  
data['animal'] = data['food'].map(str.lower).map(meat_to_animal)  
data  
foodouncesanimal
```

```
0bacon4.0pig
1pulled pork3.0pig
2bacon12.0pig
3Pastrami6.0cow
4corned beef7.5cow
5Bacon8.0pig
6pastrami3.0cow
7honey ham5.0pig
8nova lox6.0salmon
```

#Another way of doing it is: convert the food values to the lower case and apply the function

```
lower = lambda x: x.lower()
data['food'] = data['food'].apply(lower)
data['animal2'] = data.apply(meat_2_animal, axis='columns')
data
foodouncesanimalanimal2
0bacon4.0pigpig
1pulled pork3.0pigpig
2bacon12.0pigpig
3pastrami6.0cowcow
4corned beef7.5cowcow
5bacon8.0pigpig
6pastrami3.0cowcow
7honey ham5.0pigpig
8nova lox6.0salmonsalmon
```

Alternatively, you can use the assign function if you wish to create another variable. You will come across numerous functions in this chapter that you

should keep in mind. This section will help you understand how you can use the Pandas library to solve different machine learning problems.

```
data.assign(new_variable = data['ounces']*10)
```

```
foodouncesanimalanimal2new_variable
```

```
0bacon4.0pigpig40.0
```

```
1pulled pork3.0pigpig30.0
```

```
2bacon12.0pigpig120.0
```

```
3pastrami6.0cowcow60.0
```

```
4corned beef7.5cowcow75.0
```

```
5bacon8.0pigpig80.0
```

```
6pastrami3.0cowcow30.0
```

```
7honey ham5.0pigpig50.0
```

```
8nova lox6.0salmonsalmon60.0
```

Let us now remove the animal2 column from the dataframe.

```
data.drop('animal2',axis='columns',inplace=True)
```

```
data
```

```
foodouncesanimal
```

```
0bacon4.0pig
```

```
1pulled pork3.0pig
```

```
2bacon12.0pig
```

```
3Pastrami6.0cow
```

```
4corned beef7.5cow
```

```
5Bacon8.0pig
```

```
6pastrami3.0cow
```

```
7honey ham5.0pig
```

```
8nova lox6.0salmon
```

When you load a data set from the Internet, it is possible that there will be some missing variables in the set. You can either substitute the missing

variable with a dummy variable or a default value depending on the type of problem you are solving. It can also be that there are many outliers in the data set that you want to get rid of. Let us look at how this can be done using the Pandas directory.

```
#Series function from pandas are used to create arrays
```

```
data = pd.Series([1., -999., 2., -999., -1000., 3.])
```

```
data
```

```
0 1.0
```

```
-999.0
```

```
2 2.0
```

```
3 -999.0
```

```
4 -1000.0
```

```
5 3.0
```

```
dtype: float64
```

```
#Replace -999 with NaN values data.replace(-999, np.nan,inplace=True)
```

```
data
```

```
0 1.0
```

```
NaN
```

```
2.0
```

```
3 NaN
```

```
4 -1000.0
```

```
5 3.0
```

```
dtype: float64
```

```
#We can also replace multiple values at once. data = pd.Series([1., -999., 2., -999., -1000., 3.]) data.replace([-999,-1000],np.nan,inplace=True) data
```

```
0 1.0
```

```
1 NaN
```

```
2.0
```

```
NaN  
4 NaN  
5 3.0  
dtype: float64
```

Let us now look at how we can rename the rows and columns.

```
data = pd.DataFrame(np.arange(12).reshape((3, 4)), index=['Ohio',  
'Colorado', 'New York'], columns=['one', 'two', 'three', 'four'])
```

```
data  
onetwothreefour  
Ohio0123  
Colorado4567  
New York891011
```

Using rename function

```
data.rename(index={'Ohio': 'SanF', 'one': 'one_p', 'two': 'two_p'}, columns=  
           inplace=True)  
data  
one_ptwo_pthreefour  
SanF0123
```

```
Colorado4567  
New York891011
```

#You can also use string functions

```
data.rename(index = str.upper, columns=str.title, inplace=True)  
data  
One_pTwo_pThreeFour  
SANF0123  
COLORADO4567  
NEW YORK891011
```

We will need to split or categorize the continuous variables.

```
ages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32]
```

Let us now divide the ages into smaller segments or bins like 18-25, 26-35, 36-60, and 60 and above.

#Understand the output - '(' means the value is included in the bin, '[' means the value is excluded

```
bins = [18, 25, 35, 60, 100]
```

```
cats = pd.cut(ages, bins)
```

```
cats
```

```
[(18, 25], (18, 25], (18, 25], (25, 35], (18, 25], ..., (25, 35], (60, 100], (35, 60], (35, 60], (25, 35]) Length: 12
```

Categories (4, object): [(18, 25] < (25, 35] < (35, 60] < (60, 100]] #To include the right bin value, we can do: pd.cut(ages,bins,right=False)

```
[[18, 25), [18, 25), [25, 35), [25, 35), [18, 25), ..., [25, 35), [60, 100), [35, 60), [35, 60), [25, 35)] Length: 12
```

Categories (4, object): [[18, 25) < [25, 35) < [35, 60) < [60, 100]] #Pandas library intrinsically assigns an encoding to categorical variables. cats.labels array([0, 0, 0, 1, 0, 0, 2, 1, 3, 2, 2, 1], dtype=int8)

#Let's check how many observations fall under each bin

```
pd.value_counts(cats)
```

```
(18, 25] 5
```

```
(35, 60] 3
```

```
(25, 35] 3
```

```
(60, 100] 1
```

```
dtype: int64
```

We can also pass a name to every label.

```
bin_names = ['Youth', 'YoungAdult', 'MiddleAge', 'Senior'] new_cats = pd.cut(ages, bins,labels=bin_names)
```

```
pd.value_counts(new_cats)
```

```
Youth5
```

```
MiddleAge3
```

```
YoungAdult3
```

```
Senior1
```

```
dtype: int64
```

```
#We can also calculate their cumulative sum
```

```
pd.value_counts(new_cats).cumsum()
```

```
Youth5
```

```
MiddleAge3
```

```
YoungAdult3
```

```
Senior1
```

```
dtype: int64
```

You can also create pivots and group different variables in the data set using the Pandas directries. A pivot table is one of the easiest ways to perform an analysis of the data set, and it is for this reason that it is essential that you understand how this can be done.

```
df = pd.DataFrame({'key1' : ['a', 'a', 'b', 'b', 'a'],
'key2' : ['one', 'two', 'one', 'two', 'one'],
'data1' : np.random.randn(5),
'data2' : np.random.randn(5)})
```

```
df
```

```
data1data2key1key2
```

```
00.9735990.001761a
```

```
10.207283-0.990160a
```

```
21.0996421.872394b
```

```
30.939897-0.241074b
```

```
40.6063890.053345a
```

```
#Calculate the mean of data1 column by key1
```

```
grouped = df['data1'].groupby(df['key1'])
```

```
grouped.mean()
```

```
key1
0.595757 b 1.019769
Name: data1, dtype: float64
#We will now slice the data frame
dates = pd.date_range('20130101',periods=6)
df = pd.DataFrame(np.random.randn(6,4),index=dates,columns=list('ABCD')) df
ABCD
2013-01-01 1.030816 -1.276989 0.837720 -1.490111    2013-01-02 1.070215-
0.209129 0.604572 -1.743058
2013-01-03 1.524227 1.863575 1.291378 1.300696
2013-01-04 0.918203 -0.158800 -0.964063 -1.990779
2013-01-05 0.089731 0.114854 -0.585815 0.298772
2013-01-06 0.222260 0.435183 -0.045748 0.049898 #Get first n rows from
the data frame df[:3]
ABCD
2013-01-01 1.030816 -1.276989 0.837720 -1.490111
2013-01-02 1.070215 -0.209129 0.604572 -1.743058
2013-01-03 1.524227 1.863575 1.291378 1.300696
#Slice based on date range
df['20130101':'20130104']
ABCD
2013-01-01 1.030816 -1.276989 0.837720 -1.490111
2013-01-02 1.070215 -0.209129 0.604572 -1.743058
2013-01-03 1.524227 1.863575 1.291378 1.300696
2013-01-04 0.918203 -0.158800 -0.964063 -1.990779 #Slicing based on
column names df.loc[:,['A','B']]
AB
```

```
2013-01-011.030816-1.276989
2013-01-02-1.070215-0.209129
2013-01-031.5242271.863575
2013-01-040.918203-0.158800
2013-01-050.0897310.114854
2013-01-060.2222600.435183
#Slicing based on both row index labels and column names
df.loc['20130102':'20130103',['A','B']] AB
2013-01-02-1.070215-0.209129
2013-01-031.5242271.863575
#Slicing based on index of columns
df.iloc[3] #Returns 4th row (index is 3rd)
0.918203 B -0.158800 C -0.964063 D -1.990779
Name: 2013-01-04 00:00:00, dtype: float64 #Returns a specific range of
rows df.iloc[2:4, 0:2]
AB
2013-01-031.5242271.863575
2013-01-040.918203-0.158800
#Returns specific rows and columns using lists containing columns or row
indexes
df.iloc[[1,5],[0,2]]
AC
2013-01-02-1.0702150.604572
2013-01-060.222260-0.045748
You can also perform Boolean indexing using the values in the columns.
This will help you filter the data set based on some conditions. These
conditions must be pre-defined.
df[df.A > 1]
```

ABCD

2013-01-011.030816-1.2769890.837720-1.490111

2013-01-031.5242271.8635751.2913781.300696

#We can copy the data set

```
df2 = df.copy()
```

```
df2['E']=['one', 'one','two','three','four','three']
```

```
df2
```

ABCDE

2013-01-011.030816-1.2769890.837720-1.490111one

2013-01-02-1.070215-0.2091290.604572-1.743058one

2013-01-031.5242271.8635751.2913781.300696two

2013-01-040.918203-0.158800-0.964063-1.990779three

2013-01-050.0897310.114854-0.5858150.298772four

2013-01-060.2222600.435183-0.0457480.049898three #Select rows based on column values df2[df2['E'].isin(['two','four'])] ABCDE

2013-01-031.5242271.8635751.2913781.300696two

2013-01-050.0897310.114854-0.5858150.298772four #Select all rows except those with two and four df2[~df2['E'].isin(['two','four'])] ABCDE

2013-01-011.030816-1.2769890.837720-1.490111one

2013-01-02-1.070215-0.2091290.604572-1.743058one

2013-01-040.918203-0.158800-0.964063-1.990779three

2013-01-060.2222600.435183-0.0457480.049898three

You can also select columns using a query method. In this method, you will need to enter a criterion.

#List all columns where A is greater than C

```
df.query('A > C')
```

ABCD

2013-01-011.030816-1.2769890.837720-1.490111

```
2013-01-031.5242271.8635751.2913781.300696
```

```
2013-01-040.918203-0.158800-0.964063-1.990779
```

```
2013-01-050.0897310.114854-0.5858150.298772
```

```
2013-01-060.2222600.435183-0.0457480.049898
```

```
#Using OR condition
```

```
df.query('A < B | C > A')
```

```
ABCD
```

```
2013-01-02-1.070215-0.2091290.604572-1.743058
```

```
2013-01-031.5242271.8635751.2913781.300696
```

```
2013-01-050.0897310.114854-0.5858150.298772
```

```
2013-01-060.2222600.435183-0.0457480.049898
```

A pivot table allows you to customize the format of the information to help you understand the data set better. Most people use Excel to build a pivot table since that helps them understand and analyze the data fairly easily.

```
#Create a data frame
```

```
data = pd.DataFrame({'group': ['a', 'a', 'a', 'b','b', 'b', 'c', 'c','c'],  
'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

```
data
```

```
groupounces
```

```
0a4.0
```

```
1a3.0
```

```
2a12.0
```

```
3b6.0
```

```
4b7.5
```

```
5b8.0
```

```
6c3.0
```

```
7c5.0
```

```
8c6.0
```

```
#Calculate means of each group
data.pivot_table(values='ounces',index='group',aggfunc=np.mean)
group
6.333333 b 7.166667 c 4.666667
Name: ounces, dtype: float64 #Calculate count by each group
data.pivot_table(values='ounces',index='group',aggfunc='count') group
3
3 c 3
Name: ounces, dtype: int64
```

Chapter 5 Summary

- Pandas and NumPy libraries can be used to manipulate data in data sets in Python.
- The NumPy directory offers access to a broad range of mathematical functions crucial in manipulating data in Python.
- The primary functions typically utilized in manipulating data in Python include mod, multiple, log, tan, cos, sin, divide, sum, power, min, mean, var, max, and others.
- The widely used methods in manipulating data in Python offer a summary of statistics of the doable variables and the integers.

Chapter 6:

Understanding K-Means Clustering



The next thing that we need to explore here is the algorithm for k-means clustering. This is a great algorithm to use with machine learning, and it can help out a lot with the kind of programming that you do. The basic idea that comes with this is that you can take the data from your system—the type that hasn't been labeled yet—and then put it together into clusters.

Clustering is unsupervised machine learning. It is going to be applied when your data doesn't come with labels—and the goal of using this algorithm rather than one of the others available is to make it easier to find the various groups or related clusters that are already present in the data.

The main goal of a programmer working with these clusters is that the objects that end up together in one cluster should be closely related to one another, and they are not going to have a lot of similarities to those that are in other clusters. The similarity here is going to be some metric that will show how strong this relationship is between the data objects.

The field of data mining is going to use this clustering quite a bit. This is especially true if you are doing explorations. But this is not the only field that is going to benefit from this algorithm. You could also use it in fields like data compression, information retrieval, image analysis, pattern recognition, and bioinformatics.

The algorithm can work to form some clusters of data based on how similar the different values of data are. You will then go through and specify what you would like the value of K to be, which will be the number of clusters that you would like the algorithm to make from your data. This algorithm can start by selecting a centroid value that goes with each of the clusters, and then it will go through a total of three steps, including:

1.

You will want to start with the Euclidean distance between each data instance and the centroids for all of the clusters.

2.

Assign the instances of data to the cluster of centroid with the nearest distance possible.

3.

Calculate the new centroid values, depending on the mean values of the coordinates of the data instances from the corresponding cluster.

Working with the K-Means Clustering Algorithm

To work with this algorithm, the input to use for the k-means is going to be found on the matrix X. You can then go through and add in some organization of your choice to ensure that the rows you

create are a different sample, while each of the columns are going to include a different factor or feature that you would like to consider. To ensure this is going to happen, you have to focus on two main steps.

For the first part, you need to take some time to choose the centers that will make up the individual clusters. If you are looking through the data and you are not sure about where you should put these centers, you may have to start with some random points for this to see what happens. If you do this and things don't seem to be matching up that well, you can go back through and change it up and see if a new center point is going to work better.

The second step that we need to focus on is going to be the main loop. After you have the chosen centers in place, you can then decide which cluster all of the points of data need to go to. You can look at all of the samples that you have to help you figure out which of the clusters the point of data is going to belong to.

From this point, you will need to go through and do some recalculations on the centers of your clusters. You will need to do this based on the points that you assigned for each part. To make this happen, you will need to grab all of your samples and then figure out what the meaning is for these samples. And once you can come up with this answer, then you have the k-means.

You will need to go through these steps until you can make convergence happen with this algorithm. There are not going to be more changes to the cluster centers once this happens. For the most part, depending on how much data you are starting with, this is going to be done in five steps or less. But if you have a lot of data points that have a significant variance, then it is possible that you will need to do it in more steps.

To understand what is going on, let's take a look at how the k-means is going to work.

- - - - ^ -

| 1 | | 2 |

| |

| |

| |

- * - - -

| 3 | | 4 |

This is going to be our initialization point. Here, we will have four vectors that are going to be labeled as 1, 2, 3, and 4. The two cluster centers, which are k=2, have been randomly assigned in this one to points 2 and 3. We used the (*) and the (^) signs to help denote these. It is now time to start with the main loop.

The first step we need to complete during this point is to decide which cluster of our points is going to head over to. Looking above, we will be able to see that both point one and point three are going to be the centers of the clusters on the left because they are both going to be closer to that one than the center cluster. And then, the center points on the right are going to be both points 2 and 4.

The second thing we need to work with here is to do our recalculation. This needs to be done with the centers of the clusters and will be based on the points that are inside each cluster. The (*) cluster is going to be the one that moves between points 1 and points 2 because this ends up being the main points. The same thing will happen with the (^) cluster, but it will move between points two and four. It is pretty easy to come up with the mean for these two points because of the lower amount of data points, but with more complex data, you would be able to use an algorithm to make it happen. For this example, you would need to use the following code:

----- ^ -

| 1 | | 2 |

| |

* ^

| |

- * - - -

| 3 | | 4 |

You will not see any changes happen in the subsequent iterations, so for this example, we are going to be all done.

The Difference Between Fuzzy and Soft K-Means

As you are working on these, you may notice that when you are working with the k-means, they are going to be sensitive as you move them, especially to what is known as the initialization. But how do we fix this so that the areas aren't so sensitive, and you will be able to get some accurate answers that you are able to work on?

There are several strategies that you can use with this one, but one of the best is to go through and restart the k-means at least two times. When you do this, you can compare the results and see which one provides you with the best results when it is all done with. But it allows us to know the cost function and how it is going to be affected, and we can then compare the results and figure out which result is the best one to use.

As you do this, you may find that one of the methods that are going to make it easier to get through any challenges, and will ensure that you find the best results to add in information that is known as a membership that is fuzzy to each class, and see what is going on in that matter.

For example, your fuzzy point may end up being able to fit in 60 percent of your first cluster, along with matching up to 40 percent of the information in the second cluster. You would then use the

process of the soft k-means to make adjustments to the algorithm you used before. The first part of it is going to be the same, but you need to make sure that the first k-cluster centers go back to the random points that are a part of your data. But the changes that you will see can come inside some of the loops that you write.

To make this happen, we need to go through and write out some of the responsibilities of the clusters. The best formula to use to make this happen and work out includes:

$$r(k,n) = \exp[-b * d(m(k), x(n))] / \sum[j=1..K] \{ \exp[-b * d(m(j), x(n))] \}$$

From here, you can see that the $r(k,n)$ is going to work out as a fraction, some number that is in between 0 and 1, where you can interpret the hard k-means or the regular k-means to be the case where $r(k,n)$ is always exactly equal to 0 or 1. The $d(*,*)$ can be any valid distance metric, but the Euclidean or squared Euclidean distances are the ones that are used the most.

Then for the second step, we are going to work on a formula that is similar to the hard k-means, but we are recalculating the means based on the responsibilities that we want to send with it. The algorithm that we are going to use for this includes:

$$m(k) = \sum[n=1..N] \{ r(k,n) * x(n) \} / \sum[n=1..N] \{ r(k,n) \}$$

So, when you consider the algorithm above, you will see that it is similar to the weighted mean. This is going to show you that if the $r(k,n)$ is higher, then the mean is going to be more important to the cluster of k . When you see this, it is going to show that this is going to have the biggest influence on the calculation of the mean. But if the mean is higher when you look at the algorithm, you will know that the opposite is true.

What Is the K-Means Objective Function?

Just like what we have discussed about supervised learning, you must make sure that you spend some of your time looking at and talking about the objective function that you wish to get the most

out of your unsupervised learning. To get started with this idea, we will need to use what is known as the Euclidean distance to make it easier to measure the distance of each center. The function that we will need to use to figure this out includes:

$$J = \sum[n=1..N] \{ \sum[k=1..K] \{ r(k,n) \| m(k) - x(n) \|^2 \} \}$$

What all of this means is that you are working with the squared distance that was weighted through the responsibilities that it is given. Hence, if the $x(n)$ part ends up being far away from the mean of your k cluster, hopefully, the responsibility that goes with this one is set to be quite low. What happens then is known as a coordinate descent. What this is going to mean is that we are looking to move in the direction of a smaller J concerning just one of the variables at a time. We know that this is something that happened here because we went through and updated one single variable at a time.

Now, when you take a look back at all of this, you will see that there is a bit of a mathematical guarantee that all of the iterations can result in the objective function that you work with going down. If you continue to do this for long enough, and you are properly doing the algorithm, you will find that the points are going to converge together.

Keep in mind that just because you can see a convergence show up doesn't mean that they are going to necessarily come up with the global minimum that you are looking for. The numbers will go off the patterns that show up there, and they will spend time focusing on math, and less about what you may think is the most important. Going through and reading through the results and making sure that they work how you want, and retrying the whole process a few times to see if it comes up with different results can help to fix any issues this can bring up.

How to Add in the K-Means?

In this chapter, we have closely looked at k-means, and how this algorithm can work, along with the different ways that this algorithm is used to create the right solution to your program. Now

that this is done, it is time to implement some of the ideas that we have talked about, with the help of machine learning skills and Python to ensure that this can work. And the best way to make all of this come together is to implement the soft k-means into your code.

But, how are we supposed to make sure that this happens? For this to work, you have to make sure that you have your standard imports in place, and the functions of the utilities ready. This will be pretty much the same thing as the Euclidean distance as well as the function of cost put together. The formula that can help you figure out this information is going to be below:

```
import numpy as np  
import matplotlib.pyplot as plt  
  
def d(u, v):  
    diff = u - v  
    return diff.dot(diff)  
  
def cost(X, R, M):  
    cost = 0  
    for k in xrange(len(M)):  
        for n in xrange(len(X)):  
            cost += R[n,k]*d(M[k], X[n])  
    return cost
```

Once you have taken the time to add this part into the compiler and you use it to define your function so that it can run the k-means algorithm before it plots out the results. This will mean that you will end up with a scatterplot of the information. And the colors that are there will represent the amount of membership of the information found in each cluster. To make this happen, you need to work with the code below:

```
def plot_k_means(X, K, max_iter=20, beta=1.0):
```

```

N, D = X.shape
M = np.zeros((K, D))
R = np.ones((N, K)) / K
# initialize M to random
for k in xrange(K):
    M[k] = X[np.random.choice(N)]
grid_width = 5
grid_height = max_iter / grid_width
random_colors = np.random.random((K, 3))
plt.figure()
costs = np.zeros(max_iter)
for i in xrange(max_iter):
    # moved the plot inside the for loop
    colors = R.dot(random_colors)
    plt.subplot(grid_width, grid_height, i+1)
    plt.scatter(X[:,0], X[:,1], c=colors)
    # step 1: determine assignments / responsibilities
    # is this inefficient?
    for k in xrange(K):
        for n in xrange(N):
            R[n,k] = np.exp(-beta*d(M[k], X[n])) / np.sum(np.exp(-beta*d(M[j], X[n])) for j in xrange(K) )
    # step 2: recalculate means
    for k in xrange(K):

```

```

M[k] = R[:,k].dot(X) / R[:,k].sum()

costs[i] = cost(X, R, M)

if i > 0:

    if np.abs(costs[i] - costs[i-1]) < 10e-5:

        break

plt.show()

```

Notice in this one that both the R and the M are going to have their matrices. The R is going to be a new matrix because it can hold onto two different indices—n and k. But the M will also become a matrix because it is going to include the D-dimensional vectors of K. The beta variable is going to be there because it is responsible for controlling how spread-out or fuzzy the memberships of the cluster are, and it is known as the hyperparameter. From this, we can create a new main function that will create the random clusters, and then calls up the functions that we defined above. The code that you will need to use to make all of this work well together includes:

```

def main():

    # assume 3 means

    D = 2 # so we can visualize it more easily

    s = 4 # separation so we can control how far apart the means are

    mu1 = np.array([0, 0])

    mu2 = np.array([s, s])

    mu3 = np.array([0, s])

    N = 900 # number of samples

    X = np.zeros((N, D))

    X[:300, :] = np.random.randn(300, D) + mu1

    X[300:600, :] = np.random.randn(300, D) + mu2

```

```
X[600:, :] = np.random.randn(300, D) + mu3  
# what does it look like without clustering?  
plt.scatter(X[:,0], X[:,1])  
plt.show()  
K = 3 # luckily, we already know this  
plot_k_means(X, K)  
# K = 5 # what happens if we choose a "bad" K?  
# plot_k_means(X, K, max_iter=30)  
# K = 5 # what happens if we change beta?  
# plot_k_means(X, K, max_iter=30, beta=0.3)  
if __name__ == '__main__':  
    main()
```

With the formulas above, you will be able to work with the k-means algorithm, and you should have a better understanding of how this works. Open up your compiler and work on some of these codes and getting a bit of practice with writing your codes in Python. You will find that this can sharpen your skills and makes it so much easier for you to get the expected results from your machine learning.

Chapter 6 Summary

- K-means clustering algorithm helps in grouping unlabeled data into distinct and defined clusters containing closely related data.
- This sets the stage for further processing and analysis for effective machine learning.
- Unsupervised clustering offers a crucial step in machine learning since it eliminates the role of the human in the process, elevating the ‘individuality’ of the machine(s) in question.
- Clustering provides an essential basis for the data mining field, especially the unsupervised aspect of this process.
- The distance to the center in k-means clustering determines the potency of machine learning, In this case, establishing this distance is achieved as follows:

$$J = \sum[n=1..N] \{ \sum[k=1..K] \{ r(k,n) \| m(k) - x(n) \|^2 \} \}$$

Chapter 7:

Lists



Lists are another data structure in Python that also use a sequence such as strings. For each element that is on your list, there is a number assigned to it so that it is placed on the index with which you are working. The index always starts with zero and continues until you tell it to stop.

There are about six sequences that Python has built into it. However, the ones that you notice most often are lists and tuples.

While working with a sequence, you can slice, index, add, check for membership, or even multiply items that are in the list. Python allows you to use functions that are built into the program, so you can determine how long a sequence is or locate the largest and smallest items in that sequence.

Creating a List

The creation of lists is simple because a list is one of the most versatile data types that you will work with in Python. All you have to do to create a list is place the values you want in a list, and then place a comma between the variables. Always remember to place your values between a set of square brackets. Remember, you do not have to keep the same data type in your list; you are given the option to mix and match if you want to.

Example:

```
list a = ['toys', 'cars', 'ducks', 3992, 4929];
```

```
list b = [ 1, 2, 3, 4, 5];
```

```
list c = [a, b, c, d, e]
```

Accessing Elements on a List

A word of advice while working with lists in Python: you should not label them as lists. However, for the sake of convenience, the examples in this book are labelled as lists. While creating the lists, you may want to label them as what you are working with. For example, label them as a city name or the name of a person.

In order to access the elements on your list, you print the index to which you need access. Keep in mind that you are accessing where the index starts, and the one right after where you need it to end.

Example :

```
Print (list 1 [3] [3], list 1 [ 3] [4])
```

Adding Elements to the List

There will be times that you will need to add an element to your list, which means that your list will need to be updated. You will use the append method to successfully add the element to the list with which you are working.

Example :

```
#!/usr/bin/Python  
list 1 = ['car', 'truck', '76', '56'];  
print "the value located at the second index"  
print list [1]  
list [1] dump truck  
print "add a new element at the second index"  
print list [1]
```

Result:

The previous value was truck.
The new value is dump truck.

Changing the Elements on the List

There are times that an updated element will not give you the results that you need. Therefore, you have to replace the element on your list. To do this, you need to reference the place on the list where the object is located, and then what you would like the element to be changed to.

Example :

```
list 2 = [d, a, n]  
list 2 = [3]  
n  
list 2 = [3] = m  
list 2  
[d, a, m]
```

Concatenating and Repeating Lists

If you want to add lists together with Python, you are able to do this using the same method that you used with the strings. Ensure that both of the lists that you are working with are separated by a plus sign. By doing this, Python will add the lists together.

Example :

list 3 = list 2 + list 1

Result: List 3 now contains all of the data that was separated into list 2 and list 1.

Python offers you two ways to repeat items that need to be in your code. You can use the long way, or you can use the short way (which is highly recommended because it is less code that you have to write out). No matter which way you decide to use, you will be following the same steps and accomplishing the same goal.

Method 1: the number of times in xrange(d)

Method 2: [a]*d

In this example, you assume that you want to use the variable of “a”, and it needs to be repeated in two lists with a different amount in order for the variable to appear in each list.

Example :

[b] * 2

B, b

[b] * 6

B, b, b, b, b, b

Removing or Deleting Items from a List

There will be times when you have to delete objects from your list, and, just like when you need to repeat an object on your list, there are two methods that you can follow. You can use the “del” statement, so you know that you are removing the exact elements from your list. The other option is to use

the “remove” method, but this method should only be used if you do not know exactly what you need to delete off the list.

Example :

```
List 1= ['truck', 'car', 'toys', 23, 54];
```

```
Print list 1
```

```
del list 1 [2]
```

print “delete everything that occurs after the second index.

```
print list 1
```

```
Result: list 1 = ['truck', 'car']
```

Sorting Lists

If you use the sort() method, you are able to sort the objects on your list. This is similar to the “func” function that you saw earlier.

Syntax:

```
list. sort ([func])
```

This function does not have any parameters that you have to follow, so all you need to do is enter the syntax into the program, and the program will do the rest for you.

Example :

```
#!/usr/bin/Python
```

```
Alist = [32, 342, in, dent, if];
```

```
Alist.sort()
```

```
print list
```

```
Result: Alist [32, 342, dent, if, in];
```

As you can see, the program sorted out the list, so everything is in alphabetical and numeric order.

The Count() Method

As you can tell by the name, this method is going to be simple. The count method gives you a result of how many objects are in your list.

Syntax:

```
list.count (obj)
```

There is only one parameter that you have to follow while using this method. You have to remember that obj will only inform you of how many times the object appears in the list with which you are working.

Example :

```
#!/usr/bin/Python
```

```
Alist = [23, 32, def, 23, light, 23];
```

```
Print "how many times 23 appears in Alist.count(23)
```

```
Result: 23 appears 3 times.
```

List Comprehension

When you use list comprehension, you create a list in Python, and the only difference is that you are also defining your list. Lists typically have the same qualities that sets have, but this is not always the case.

While you use list comprehension, you substitute having to use the lambda function, and the map, filter, and reduce functions as well. Therefore, list comprehension does everything for you, and you will not have to go through multiple steps using multiple functions.

The use of list comprehension makes you use square brackets in your expression. It is normally followed by a “for” clause, which is then followed by another “for” clause or an “if” clause.

Example :

```
[(a,b) for l in [3,4,5], for a in [6,7,8] if l ==a]
```

Map and Filter Function

Each person has their own opinion on using lists in Python. There are some people that think that these functions waste time. However, you cannot listen to someone else's opinion; what works for them may not work for you, and vice versa.

Python provides you with the opportunity to use the map and filter functions separately, but you also have the option to use them together.

Example :

You are searching for the square root of any number that is under 2.

```
Num = [9, 2, 3, 4, 5]
```

```
squares = [ ]
```

```
for num in num:
```

```
    if num < 2
```

```
        squares.append (num*num)
```

```
Result: [3]
```

Now that you have gotten the answer that you want, your code will begin to get longer, and that is not necessarily something that you want to happen. You want your code to be as short as possible so you can find an error more quickly. This is where your map and filter functions will be especially helpful.

Example :

```
Num = [ 1, 3, 5, 7, 9]
```

```
Squares = map(lambda a: a*a, filter (lambda a: a < 6, num))
```

```
# result [ 1, 9, 25]
```

Do you notice that the code is smaller with this second example? This is a good thing. However, it is not easy to read, and if you end up getting an error, you will still have to go through the code and find where the error is. If you, the creator of the code, cannot read the code that you created, what is the point in using it?

Now, let's explore what the list comprehension function would make the list look like.

Example :

```
Num = [ 1, 3, 5, 7, 9]
```

```
Squares = [ num * num for num in num if num < 6 ]
```

```
# result [ 1, 9, 25]
```

You get the same result, and your code is much easier to read! The map and filter functions provide you with a shorter code block, but they are not necessarily easier to read. This is one example where using the longer code block is better.

It does not matter which method you prefer to use. The point is for you to find the function that works best for you. This ensures that you are able to read your code easier in the event that you are given an error by the program.

Generated Expressions

List comprehension is a great tool to use while working with Python. However, there will be times when you should try to avoid using it. Despite how hard people try, nothing is foolproof. The biggest disadvantage to using the list comprehension function is that the entire list that you are working with is going to be placed on the program's memory all at the same time. This isn't a problem if you are working with a list that is small and is only a few objects long. However, what happens when you are working with a longer list? In the end, you will just be wasting your time, and you will end up having to redo your code anyways. So, save yourself the trouble and just skip using list comprehension in the first place.

The best option that you have is to use generated expressions. Generated expressions were introduced to Python users in Python 2.4, and have helped change how people write their code so that they no longer have to use list comprehension. Whenever you are working with generated expressions, you don't enter your entire list into the memory bank at the same time.

Instead, you create an object that will cause a single element to be loaded into the list at a time.

If you need to use your entire list, you won't want to work with a generated expression. However, if you just want to pass the expression off for something like a "for" loop, then your generator function is the perfect tool for you.

The syntax for a generated expression is the same syntax that was used when using list comprehension. The only difference is that your parentheses need to be outside of your brackets.

Example:

```
Num = (2, 4, 6, 8)
```

```
Squares_under_30 = (num * num for num in num if num * num < 30)
```

```
# any square that is found to be under 30 will generate an object which will force each value to be called on.
```

```
For square in squares_under_30:
```

```
Print square,
```

```
# result: '2, 16'
```

The code is not necessarily as short as you want it to be, but it will be much more efficient than if you were to use the list comprehension function. You will not be loading your entire list into the memory bank at the same time.

If you need to use this function for a list that has multiple elements in it, then you have the option to use the list comprehension technique. However, you need to keep in mind that your entire list will be loaded. If that is not something that you want to happen, you need to use the generated expression, even if it takes a little extra time to go through the function.

Whether you use the list comprehension function or the generated expression function is up to you and how long you want your code to be. Just remember that, when you use a generated expression, you need to use a single set of parentheses. So, if you are calling on a function that has a generated expression, you only need to use a single set of parentheses.

Checking Element Conditions

There are conditions that will need to be met by the items that are located on your list, and there will be times that you will need to make sure that these elements are meeting each of these conditions.

If you are using Python 2.5, then the code that you are working with will be similar to this:

Example :

```
Num = [2, 4, 6, 8, 10]
if [num for num in num if num < 2]:
    print 'two elements that are found to be over two.'
#the result will be any element that is over 2
```

If there are elements on your list that are able to satisfy the condition, then, by default, Python will make an empty list and evaluate that list to be false. However, a non-empty list will be set up and evaluated as true in the event that the condition is met. You won't have to go over every item that is located on your list. Instead, you can quit assessing the elements the minute that you find one to be true.

When Python 2.5 was introduced, there was a built-in function known as "any". This function completes the same job as the one that you just saw, except your code is shorter and even easier to understand. The "any" function was programmed to bail and provide you with a true answer once it located the first element that satisfies the condition that you set. This function can also be used with generated expressions, so you don't have to go through and evaluate each element on the list, and you'll be provided with a true or false answer.

Example :

```
num = [2, 4, 6, 8, 10]
if any (num < 2 for num in num):
    Output: success!
```

If you want to, you have the option of double checking each element that meets your condition. However, if you are not using Python 2.5, your code

is going to look different.

Example :

```
num [2, 4, 6, 8, 10]
```

```
if len (num) == len([ num for num in num if num < 2]):
```

Output: success!

In the example, the list comprehension technique was used to filter and determine if there were any elements that met the condition as there were before. It also assessed to see if the elements met the condition that you set forth. Unfortunately, this is not the most efficient way to accomplish this task because you need to check each element on your list to check if it satisfies the condition that you defined. However, if you are not using Python 2.5, this is the only option that you have of checking the elements against your condition.

When you look at Python 2.5, there is another function that you are able to use known as the “all” function. Just like any other function, the “all” function is set to terminate after it finds one element that does not meet your condition, which would give you a false evaluation.

Example :

```
num = [ 2, 4, 6, 8, 10]
```

```
if all (num < 2 for num in num):
```

Output: success!

Chapter 7 Summary

- Lists are Python structures that employ sequences such as strings. Working with sequences allows one to index, slice, check for membership, add, or multiply items on the lists. These processes are facilitated by in-built Python functions.
- Accessing lists in Python requires one to print the index to which access is needed.
- Lists are created with a specific set of data designed to be inculcated into the memory of a machine. It bolsters specific or targeted machine learning.
- Generated expressions play a crucial role in guiding or influencing code writing.
- A single set of parentheses is required to call on a function with a generated expression.

Chapter 8:

Neural Networks With Scikit-Learn



Neural networks are a machine learning framework that tries to

mimic the way the natural biological neural networks operate.

Humans have the capacity of identifying patterns with a very high

degree of accuracy. Anytime you see a cow, you can immediately

recognize that it is a cow. This also applies to when you see a goat.

The reason is that you have learned over a period of time how a cow

or a goat looks like and what differentiates between the two.

Artificial neural networks refer to computation systems that try to imitate the capabilities of human learning via a complex architecture that resembles the nervous system of a human being.

The structure of Neuron

A neuron is made up of the cell body, having a number of extensions from it. Majority of these are in the form of branches commonly known as “*dendrites*” .

A long process or a branching exists, and this is referred to as the “*axon*”. The transmission of signals begins at a region in this axon, and this region is known as the “*hillock*”.

The neuron has a boundary which is known as the “*cell membrane*”. A potential difference exists between the inside and the outside of the cell membrane. This is known as the “*membrane potential*” .

If the input becomes large enough, some action potential will be generated. This action potential then travels will then travel down the axon and away from the cell body.

A neuron is connected to another neuron by synapses. The information leaves the neuron via an axon and is then passed to the synapses and to the neuron which is expected to receive it. Note that a neuron will only fire once the threshold exceeds a certain amount. The signals are very important as they are received by the other neurons. The neurons use the signals or the spikes for communication. The spikes are also responsible for encoding the information which is being sent.

Synapses can either be inhibitory or excitatory. When spikes arrive at the excitatory synapse, the receiving neuron will be caused to fire. If the signals are received at an inhibitory synapse, then the receiving neuron is inhibited from firing.

The synapses and the cell body usually calculate the difference between the incoming inhibitory and excitatory inputs. If this difference is found to be too large, the neuron will be made to fire.

Chapter 8 Summary

- Neural networks functionally operate at the neuron level.
- Neurons are composed of cell bodies with many extensions emanating from it called dendrites.
- Operationally, Neurons operate like cells within a human body system – complete with boundaries (equivalents of the cell membrane)
- Neurons are connected to each other through synapses to establish an elaborate network.
- The collection of neurons make up a neural network complete with a system of signaling or flow of information that informs inference.
- Neurons use the signaling system or the spikes for communication, leading to the encoding of information for specific purposes or functions that the machine might be required to discharge.

Chapter 9:

Machine Learning And Big Data



In machine learning, we use algorithms that don't rely on programming rules in order to learn from data. However, this data is sometimes massive and heavily influences an algorithm's prediction accuracy and the time it takes to perform the training process. In machine learning, this is referred to as big data, and it can measure up to petabytes in size, resulting in a treasure trove of information. If you stopped to think what a petabyte is, that's how large big data can be.

The idea of handling big data, however, is nothing new. The issue surrounding the work with massive quantities of data has existed for

decades, even in the days of the old computers with very limited processing power and storage capacity. The classical types of analysis techniques could not handle big data, simply because the technology was too limited at the time.

The Challenge

Today, though, big data has become a field of its own, but it is still a part of data science and machine learning. A more recent definition has been coined, and it describes today's big data as any chunk of information that contains a varied amount of data that keeps coming at an exponentially increasing volume and speed. In this case, when we refer to the data volume, we are discussing a quantity of information that is too large to be able to analyze. As mentioned, this issue has existed since before the days of the Internet, and just because our computers are far more developed than back then doesn't mean that big data is no longer overwhelming. Keep in mind that nearly every sector of society relies on computers and the constant stream of data that needs to flow continuously. The volume of data we have to handle is increasing exponentially, and according to professional machine learners and data scientists, every couple of years, the volume doubles, and the speed at which it comes isn't being reduced either.

Consumer electronics have led to a massive increase of data flow, and with ever-evolving processors and storage systems, together with the improvement of the Internet's infrastructure, the speed at which new big data is created is almost unimaginable. New technology had to be devised in order to handle massive data, and new hardware had to be developed at an increased rate in order to be able to process all this information in real time. Imagine that back in 1986, the world's communication infrastructure could handle approximately 280 petabytes. In 2017, the amount of information that was being uploaded on a daily basis reached beyond 300 petabytes. Take a moment to let the idea sink in.

With that being said, not all information is created equal, because there are a lot of different formats. There's a great variety of data everywhere - from emails to stock market information - and it all needs to be collected and

stored. The problem is that not all storage systems are created equally, and some formats need specifically designed applications to handle them. On top of the storage space problem, all of the data needs to be categorized so that it can actually be used. This is another aspect that makes big data difficult to handle and also makes it challenging for machine learning algorithms to work with it.

Remember that all of this collected data needs to be given a value in order to be worth anything. This is where the value of big data is directly affected by what type we are talking about. If we need to analyze data that encompasses all of the tweets ever typed, for example, it might first need to go through a pre-processing phase that would take a great deal of time and resources to manage. From this issue, another challenge arises which is related to cybersecurity. Continuing with Twitter data as an example, the information that's gathered and stored is related to the platform's users one way or another. Security becomes a major source of concern when it comes to big data, especially after scandals such as the one involving Cambridge Analytica. For instance, all of these challenges and security issues that revolve around big data have led to the European Union adopting the General Data Protection Regulation, which clearly defines the consequences of irresponsible data gathering. A small data leak can lead to serious problems; however, a big data leak is disastrous.

Now, if we also take a look at all the collected big data, we will notice that the quality varies as much as the value. The accuracy of the information coming from enormous datasets is the main factor that contributes to how useful that information is, and there are many factors that contribute to its inaccuracy. The first thing you should ask yourself is whether those who gathered the data have attempted to fill in some of the gaps with theoretical assumptions that aren't based on factual data. Secondly, you might also wonder about the storage system that held on to that data. Was it ever tempered with? Did it suffer any damage? Think about several air pressure measurements that were done by different sensors in the same area but that all recorded different values at the same time. Which one is the right one? Is there even such a thing as a right one? Did the sensor malfunction? So many things can go wrong, especially when it comes to large amounts of data. Even the source of it can be tainted. What if our dataset contains information that was extracted from someone's social media page? There's

no way of knowing whether it came from a series of bots or actual people. Plus, we have to take into account the possibility of mistakes made by those who handled the data between any of the steps we mentioned.

As you can see, working with big data is challenging due to so many factors. Whether they are due to technological limitations, human error, or ineffective algorithms, this is a subfield of machine learning that still requires a lot of work. This is why, in this chapter, we are discussing the basics of big data and what we can do with it. Data continues to pile up, and eventually, you will have to work with petabytes worth of it, whether tomorrow or in 20 years, as new techniques are developed.

Real-World Applications

Big data is as part of machine learning as machine learning is of big data. As long as data and information are being produced, machine learning is a field that is here to stay for the foreseeable future. For instance, let's say the market crashes so badly that we relive the Great Depression. Who is going to process and analyze all the data that was gathered during the crash? That's right - you will fulfill that task as a machine learner.

However, the classical techniques for pre-processing, processing, analyzing, and cleaning the data will become relics at some point. As datasets become bigger and bigger at a faster pace than ever before, we will soon no longer be dealing with small datasets. Big data will become the future, because it's far more powerful and it offers the potential for a more accurate prediction. For now, working with it is not an easy task, but that will eventually change, and this is why you should prepare yourself for the future. With that being said, let's take a look at how useful big data can be and understand its applications:

1. Predicting market demand: Market data is massive, and it just keeps getting bigger and bigger. Corporations are fighting each other and competing constantly to acquire data on their customers and users in order to understand the consumer's demands and purchasing habits, thus further increasing their profits. This is an example of big data, as we can gather all

the information of past and present services, products, and purchases in order to predict the patterns.

2. Maintenance practices: Hidden deep under mountains of data lies the factors that can predict when a tool or component will fail. Big data is and will be vital to all industries, whether we are talking about screw manufacturers or aeronautics. Sifting through big data yields information like the material used, certain characteristics, the model, the manufacturing year, and much more. This data can then be processed with the help of machine learning techniques, and the result will predict when a part will fail at functioning under optimal parameters. Once we have an accurate prediction, maintenance crews can be dispatched to take care of the problem before it even happens, thus lowering any financial or health risks.

3. Efficiency: Big data has already proven itself to be extremely valuable when it comes to yielding results regarding value and return. Even with today's costly methods of analyzing it, companies gain profitable information about their own production facilities, product returns, resource waste, and even future trends. There are many factors that go into a business, especially when we are discussing a manufacturing facility. Calculating costs of production and predicting profits is no easy task, but a growing number of machine learners are more valuable in this field than economists.

As already mentioned, big data is emerging as the next best thing, and machine learning will be at the forefront of it all. When it comes to data preparation, predictive analysis, and overcoming future data challenges, machine learning is the safest bet. Keep in mind that only in the last two years, humanity has produced approximately 90% of all the data ever recorded in history, and this trend is only accelerating. As a way to compare today's classical machine learning and big data, think of the earlier marketers who were interested in gathering data on every web page you visit and compare them to today's marketers who want to know about every single click you make.

Cloud Computing

So far, we have discussed the impact of big data and the many challenges we are faced with, but luckily, cloud computing technology is here to save both big data and machine learning. Cloud computing is about easily available and affordable data storage that anyone can take advantage of, whether a private citizen or a mega-corporation. While the concept behind it is very easy to understand, implementing this tech is costly; however, it's already present on the global market for mass consumers, and it's growing in popularity exponentially.

In order to understand cloud computing, you should imagine a vast building that houses thousands, or even tens of thousands, of computers that are operating at the same time in order to provide us with this data storage option. This is a data center (one of many in fact), and this is what it takes for cloud computing to be available to all of us. As you can imagine, cloud storage is a massive undertaking that requires an unbelievable amount of equipment and electricity. The fact that there are several companies that took on this titanic undertaking tells us that cloud computing is the next best thing, especially for machine learning and data analysis.

Another major reason why cloud storage is so useful is due to its ability to provide the user with a self-adjustable on-demand storage system. Imagine you own a business, and you're running a website, together with some web-based application attached to it. This means you are using a server where you store all the online information, and normally it fulfills your requirements. However, what will happen when you experience a significant increase in website users, like Amazon does when the holiday seasons are approaching? The website will be too stressed due to too many users and a lack of server space, and it will eventually crash, or at the very least lead to a terrible user experience. Cloud computing is the cure to this ailment; you no longer have to risk running into such situations due to its ability to automatically scale the storage amount you need. This is where the idea of load balancing comes in. There's no need to pay for a large amount of storage that will sit unused throughout the year except during particular events. With load balancing, the cloud storage system will distribute storage space as needed, and the process is done nearly instantly. When the system detects that you are closing in on the storage threshold, it will automatically increase the amount of storage you need. Once the need for storage decreases, the system scales it down. Because of this factor,

storage is becoming more affordable. One of the biggest issues with big data is that it requires equally big storage.

You should already know that some algorithms are more complex and they require large datasets, as well as access to enough processing power to perform every process in an adequate amount of time. Naturally, all of these factors that need to be taken into account by the machine learner lead to a process that is extremely costly. When trying to work out real-world problems under normal circumstances, an individual or a small company cannot afford all of these resources with the necessary tech and specialists. Machine learning is normally utilized by large companies and corporations, governments, and research labs. This is where cloud computing technology begins to shine and change everything. Machine learning becomes a more affordable endeavor, and everyone has the opportunity to win in the process.

As you already know, machine learning techniques are used a lot in pushing the limits in technology research and by major companies that are looking for that extra margin to push them ahead of their competition. However, machine learning methods are also integrated into the Cloud itself, because certain services require them to fulfill various functions. There are cloud services specifically arranged for data scientists, analysts, and machine learners that offer them an affordable way to work on massive datasets without purchasing a large number of hard drives on which to store them.

All of this means that now is the perfect time for you to become involved in machine learning, as it has never been easier or more affordable than it is currently. The machine learning enthusiast can achieve a lot and develop skills at a professional level after gaining access to cutting edge technology that doesn't require a massive loan from the bank. You even have access to other cloud-based services that include machine learning development kits and various software programming interfaces. These solutions allow for hassle-free implementation of machine learning techniques directly in the application you're working on. Another major selling point of these services is that you don't even need to use Python if you are used to working with another programming language (however, it is recommended to opt for Python in machine learning). Cloud computing technology includes support for a multitude of programming languages, meaning that

machine learners can focus more on data processing and less on adjusting themselves to new languages.

Chapter 9 Summary

- Ubiquitous consumer applications and the proliferation of their use across the society have led to a massive surge in the quantity and variety of data.
- Machine learning is typically independent of programming. However, with the increase in Big Data, algorithms have become crucial in machine learning to enhance the ability of subject organizations to create value.
- The importance of the machine learning-Big Data intersection can be seen in predicting market demand, efficiency, and maintenance practice in business.
- Cloud computing offers an accessible solution to big data at the individual and organizational levels.

Chapter 10:

Machine Learning and Support Vector Machines



We have now taken the time to look through two different algorithms that work with machine learning. When we look towards using the SVM algorithm, you will find that programmers are often going to use this when they are going to have regression and classification problems that show up with the data that they use with their work. When we take a look at this one, there is a lot of work that needs to be done with classification, which is going to make the work that is done trickier to handle. But with SVM and

the algorithms that come there, you will find that you are able to handle these challenges in a more efficient manner overall.

To start with when you are ready to work with this particular algorithm, it is important for the programmer to take each of the sets of data and then you need to plot them out so that they are just one point in an n-dimensional space. N is going to be for all of the features and all of the work to translate this over to the value that shows up on the coordinates. The work that is then done to reach this point is to take the time to determine the hyperplane. We want to know this because it shows us the different classes that are showing up.

At this point, you may notice that with the algorithm for SVM, there can be more than one support vector that will show up. However, not all of these are going to be the ones that you want to use and some are just coordinates of the individual observations that you see. Then you are able to use the SVM to be the frontier that comes in and helps to separate the different support vectors that you see into classes. Once they are divided up this way, it is easier for you to focus on the ones that actually matter.

This may again seem a bit confusing when we just list out the uses and the instructions that go with the SVM, without taking a look at how they work or seeing some examples. There are a few steps that you are able to work with and take in order to help use the SVM to help you sort through the data that you have.

To get started with this one, we need to look through the hyperplane. There is the possibility that there are two or more hyperplanes available for you to choose from, and this adds in a challenge because you want to pick out the one that works best for you and will give you the most accurate results overall. The good news here when you feel a bit worried about which hyperplane you should work with is that there are actually some suggestions that you can follow in order to look through your choices of hyperplane and help you to pick out the one that is right. The steps that you need for this one includes:

- We are going to start out with three hyperplanes that we will call 1, 2, and 3. Then we are going to spend time figuring out which hyperplane is right so that we can classify the star and the circle.

- The good news is that there is a pretty simple rule that you can follow so that it becomes easier to identify which hyperplane is the right one. The hyperplane that you want to go with will be the one that segregates your classes the best.

- That one was easy to work with, but in the next one, our hyperplanes of 1, 2, and 3 are all going through the classes and they segregate them in a manner that is similar. For example, all of the lines or these hyperplanes are going to run parallel with each other. From here you may find that it is hard to pick which hyperplane is the right one.

- For the issue that is above, we will need to use what is known as the margin. This is basically the distance that occurs between the hyperplane and the nearest data point from either of the two classes. Then you will be able to get some numbers that can help you out. These numbers may be closer together, but they will point out which hyperplane is going to be the best.

The example that we have listed out in steps above is only one example of when you will be able to use SVM as a tool with machine learning. But it is a good one that gives us an idea of what we are able to do when we bring this algorithm into the mix. When you have a set of data that you can use, and you see that there is some kind of margin in place that helps you to separate out these points, then it is possible that you will want to bring out the SVM algorithm to help you get the information that you need.

Another reason that you will want to work with this particular machine learning algorithm is that with this model, you will be able to increase it and make it work on any kind of project that has a dimensional space that is considered higher than normal.

While there are going to be some times when the SVM method is not going to be the best one for you, it is still a good machine learning algorithm that will help you to sort through your data set and get some answers. It is going to be a technique that works when you have a subset of training points that are present with your decision function, which is going to be the support vector, and when you see that the memory of any system or program you have is high enough to make this happen as well.

There are a lot of benefits to bringing this algorithm out and having it go to work for you. But there are some times that it is going to cause more harm than good. For example, when there is a really large set of data that you need to sort through, this algorithm may not be able to get you the most accurate predictions like other options. The training time that comes with these can be higher too, and this is a disappointment for some people who are not used to working in machine learning. And finally, when there are some of your target classes that end up overlapping, then this algorithm is going to not show you the results, or behave in the manner, that you want.

Chapter 10 Summary

- The SVM algorithm offers a platform when dealing with classification and regression challenges in the course of handling data.
- The SVM algorithm can support more than one vector.
- The SVM algorithm employed in machine learning is flexible regarding the size of the project (quantity and variety of data).
- The machine learning algorithm enables the sorting of data specifically to provide answers for targeted questions.

Chapter 11:

Using The Neural Networks



Now we need to take a look at how to actually do a few things with this library and what better place to work with this library than with the neural networks.

When we look at neural networks, we are going to look at a method that works as an unsupervised machine learning algorithm. They are going to be used when it is time to catch on with a pattern that is there with the data or the information that you are looking through. Now, this is something that is going to be done at different levels, and in a manner that is faster and more effective than what can happen with the human eye.

In these neural networks, you will find that each layer that you have will spend some time looking at the object or the image and see if there is some kind of pattern that is found there. If this process means that it is able to find a new pattern with that layer, then it is time to go through a new layer and will start it all over. This is a process that will continue on, with one layer that finishes and moves

on to another, until all of the layers are done. From here, the algorithm is going to be able to give a good prediction to the program about what is inside the image.

From this kind of process, there are a few things that are going to show up, based on the way that the program is able to work. If the algorithm went through this kind of process, and it is successful with sorting through the layers, it will then make a prediction. When this algorithm is able to make an accurate prediction, then you are going to find that the neurons of the system remember this and become stronger.

The reason that this works, and why the neurons are going to be able to get stronger is that it works with artificial intelligence. This helps the system to make some really strong associations between the patterns that show up and the object. The neat thing here is that the more times that your system is able to do this process and provides you with the right answers, the more efficient it is going to be at doing this.

Now, this may seem a little bit far fetched and like it isn't something that could actually happen. But a closer examination of these neural networks will help us to see how they work together and why they are so important. For our example, let's say that your goal is to create a program that is able to take a picture that you input into it, and then, by looking at that picture and going through the layers, the program is able to recognize that the image in that picture is that of a car.

If the program has been set up in the proper manner, it is going to make the right prediction that there is a car in the picture. The program is able to come up with this prediction based on some of the features that it already knows belongs to the car, including the color, the number on the license plate, the placement of the doors, the headlights, and more.

When you are working with some of the conventional coding methods that are available, this process can be really difficult to do. You will find that the neural network system can make this a really easy system to work with.

To get this algorithm to work, you need to take out an image of a car and then provide it to the system. When the neural network is instigated, you will find that it is able to take a look at the picture. It is going to start its work with the first layer, which in this case is going to be the edges of the car. Then it would continue on with more and more layers to ensure that it is able to figure out which unique characteristics are going to be in the picture until it figures out that the image is of a car.

Now, if the program is successful and is good at making the right prediction then it is going to store this information and will get better at recognizing all of the smaller details of the car to use the next time. It will get good at recognizing the windows, the color, the wheel patterns, and so much more and it can use this information the next time that you present it with information and a picture of the car.

Depending on the image or the process that you are going to work with in neural networks, you will find that there could be a lot of different layers that come with it. But this is actually a good thing, though originally it is going to slow the whole process. It allows the neural networks to get better at finding the features that it is looking for. For example, when you are trying to do a kind of software that works with facial recognition, the neural network needs to be able to go through a lot of different layers in order to figure out who belongs in one area, and who does not.

Now, you can imagine that there are going to be a lot of ways that you are able to use neural networks to help with machine learning, and this brings a lot of benefits with it as well. One of the advantages that you will be able to see when you work with the neural networks is that you can use this method without the requirement of having to control the statistics of the algorithm from the beginning.

Even when you are not able to bring in all of the statistics from the beginning, or you are not sure exactly how you will be able to use them, you will find that the neural networks can be used to ensure that any complex relationship that is there will actually show up. This is important because it is going to happen between the

variables, whether they are dependent or independent, and even if you find that the variables are nonlinear.

This method is not going to work all of the time, and there are situations where it is going to take too much time and will be too complicated in order to get the results that you want. For example, the biggest negatives that a lot of programmers are going to see with this is that the cost of computing is high. For some projects, and some of the businesses, this is just going to take up too much time, too much computational power, and too much money to make it worth the effort.

Scripting and Modularization

In scripting mode, all you need to do is write your program on a text editor. If you are done with it and you want to try it out, you must save it as a .py file. If you have Python installed properly, you can just open the .py file you saved and it will execute your program or script.

On the other hand, if you have a large project going on, you can modularize. Modularizing means that you will code your program using multiple files. For example, you can write all of your program's functions in a .py file named modFunctions.py and then you can put the main program flow in a .py file named main.py.

Of course, they will not automatically work together. To make them work together, you will need to load the module files you have in your program. And you can do that by using the import and from keywords.

Import Keyword

The import keyword allows you to load a module in your program and it then creates a reference for the module that you loaded. If you

need to access or use an attribute or method, you will need to use the complete path from the module.

Below is an example module:

```
# SimpleMath module

def add(x, y): #add two values and return the sum
    result = x + y
    return result

def subtract(x, y): #subtract y from x and return the difference
    result = x - y
    return result
```

Now, this is a short project done in the interpreter that will import the method add() from the example module.

```
>>> import SimpleMath
>>> simpleMath.add(1, 2)
3
>>> simpleMath.subtract(1, 1)
0
>>> _
```

From Keyword

An alternative to call in modules to the project is to use the from keyword. With it, you can use an alternate format in loading a module. Aside from that, you can use it to selectively retrieve attributes or methods from the module.

For example:

```
>>> from SimpleMath import add
```

```
>>> add(2, 3)
```

```
5
```

```
>>> _
```

If you want to load everything from the module, you can indicate it by using the asterisk (*) expression. For example:

```
>>> from SimpleMath import *
```

```
>>> add(6, 23)
```

```
5
```

```
>>> subtract(10, 2)
```

```
8
```

```
>>> _
```

As Keyword

If you do not want to create conflict in your project and the module you will load, you can assign an alias, using the as keyword, to the method or attribute that you will import. Below is an example:

```
>>> from SimpleMath import add as addition
```

```
>>> addition(12, 67)
```

```
79
```

```
>>> _
```

In machine learning and other large projects, loading modules will be common practice. Also, with machine learning, you will be forced to work on multiple modules and libraries from other users. After all, modularization is an efficient way to create programs.

As a pro tip, never forget about your old projects. Never delete them, and instead, let them sit in your computer for a while. Some of the functions that you might have created before can become

handy in your current project. You can modularize them to save some time.

On the other hand, keep accumulating those pieces of modules. There will be a point in time that you have created so much that you will be able to have a useful library.

How to Use Python With Machine Learning

At this point, you are now familiar with programming and Python. Now, you will learn how to use Python in machine learning. There are three things you need:

- Python
- Data Set
- Learning Model

Warnings

As a word of caution, it is advised that you work on Python programs if you are a complete beginner. Play around with the programming language and familiarize yourself with the quirks of Python. The more time you spend on programming, the “easier” dealing with machine learning will be. Also, when you create a project, be sure to finish it. There is nothing worse in hindering your learning progress than not finishing your projects.

Another thing that you should note is that creating machine learning is not a linear project. However, below is a rough guide on the things you need to learn and do:

1. *Identifying the Problem You Want to Solve*
2. *Preparing the Data You Need*
- 3.

Establishing Your Algorithm

4. Refining Your Results
5. Presenting Your Results

With machine learning, you should have a clear goal. And more often than not, the initial goal of machine learning is to solve a problem. For example, the problem can be to identify which phrase in the whole Harry Potter series was the most memorable.

To solve that problem, it is, of course, mandatory to get data. For the example problem, your data can be the entire text of all the books in the Harry Potter series and/or search data in search engines or Harry Potter-related websites.

Once you have your problem and data, the next step is to establish how you will solve the problem using the data you have. For this part, you will need to create the algorithm. How will your program know which phrase in the Harry Potter book is memorable? How will it judge? How will it decide? How will it parse the data?

The next step after establishing your algorithm is to test it. Let your program run and wait for the result. Once the results are out, analyze it. Was the program able to solve the problem? Were the results satisfying? If the program returned unsatisfying results, then it is time to refine it.

After refining and achieving the results you want, and then it is time to work on how the program will present its data. Will it just be generating tables? Will it create prose to describe and present the results?

Data Visualization and Pre-Processing

Nowadays, data visualization has become all too common. After all, data is now everywhere you look — social media, books, television, and even executive dashboards — and people yearn for it like water in a desert.

What is data visualization anyway? Data visualization is a method of presenting data, usually in large amounts, in an understandable and easily digestible way. It might involve static and dynamic elements. It might take advantage of design trends.

Why is this important in machine learning? Well, first and foremost, machine learning's usual job is to take in and generate data. It is typically machine learning's purpose. Computers process and manipulate data. And of course, the one that will use all those data are people. And if a computer or user did not bother to work on data visualization, the end result of machine learning will be just a jumbled mess of strings and numbers.

Aside from the advantage of providing people with understandable data, the machine can also benefit from data visualization. Adding data visualization into the program can make it understand, which data are important to the end-user.

For example, if you create data visualization for the machine-learning program to find the most popular phrase in the Harry Potter series, the program will instantly know how to present and pick the data, which is important to it and the user.

Pre-Processing Methods of Data Mining

Data pre-processing has been an important aspect in the IT industry. With the existence of big data and more modern and effective data mining, pre-processing data has never been crucial. Without it, processing huge amounts of data will be impossible or, at the very least, heavily time- and resource-consuming.

In machine learning, processing huge volumes of data will be a typical scenario. And if you want efficiency, applying pre-processors for your data should be a default component.

There are multiple data pre-processing modules that can be found on the Internet. You can use them in your program freely. However, if you are going to deal with a unique data set or the result that you

want is complex, you will need to create the program for your pre-processor by yourself.

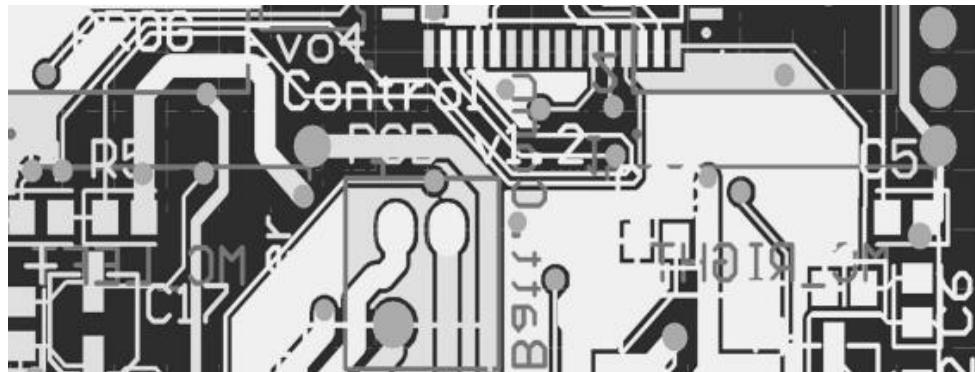
On the other hand, pre-processing is important for you to harvest and manipulate data precisely. It is inevitable that some data sets may contain incompatible pieces of data that may easily ruin your program's algorithms. Because of that, it is best that you get your data filtered to avoid getting errors and inaccuracies in your data processing. This is especially crucial if your data set contains inconsistencies and is not 100% reliable.

Chapter 11 Summary

- Functionally, neural networks operate as unsupervised machine learning algorithms.
- Learning is instigated through the provision of targeted data.
- The set-up of the algorithm is fundamental to its ability to offer any predictive insight or projections.
- Additionally, the functioning of machine learning and neural networks is dependent on the learning process, which is bolstered through the efficiency built from continued provision of right answers.
- Python offers a crucial platform for machine learning – it offers the functions needed, data sets, and the learning model required.

Chapter 12:

Artificial Neuron Networks



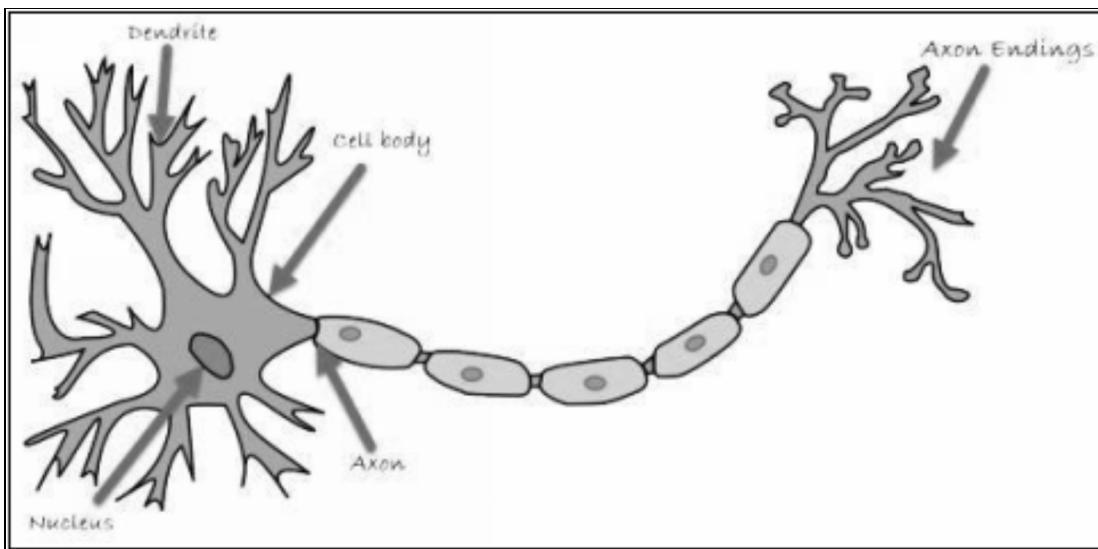
Neural networks are one of the very recent fields. Currently, large corporations and also growth companies are all rushing into this state-of-the-art field. Deep learning is a term that is associated with neural networks. Both deep learning and neural networks have great ability to learn from data and the environment. Therefore, they are the first and preferred choice for machine learning. Common application in which neural networks are employed are:

- Cars auto-driver.
- Image processing and recognition.

- Consult and recommender systems.

Artificial neural networks are powerful schemes that show great adaptive ability to a wide range of data types. Being artificial brings to the surface the origin of the name. Artificial neural networks mimic the human nervous system. A neuron consists of the following main parts:

- *Dendrites*: These are the recipients that receive the input electrical impulses from other connected neurons. The decision is then taken by the cell body, which concludes to the action needed from those inputs.
- *The axon and the axon endings*: These are the neuron transmitters. They transmit the output decisions as electrical impulses to other neurons.



Human Neuron

The neuron can be divided into three parts/stages: inputs, processing, and output. This is the key to state the analogy between the biological and artificial neural networks.

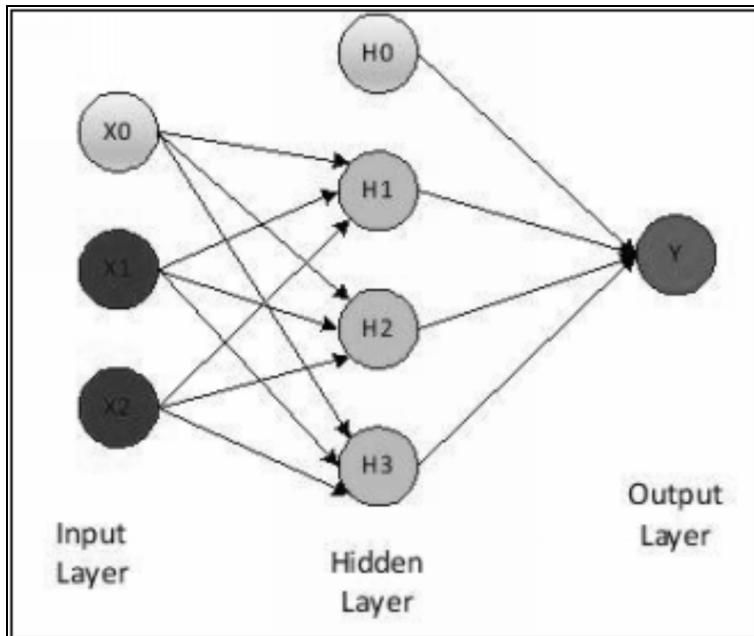
Artificial Neural Networks

An Artificial Neural Network (ANN), is a computational model that benefits the idea of the biological neural networks, as depicted previously. The artificial neural network is a machine learning method that mimics human intelligence in dealing with data. Neural networks are a remarkable class of supervised machine learning algorithms. They are employed in both regression and classification. Other known types, used according to the data to be analyzed, are convolutional and recurrent neural networks. The first showed very good performance in image processing. Recurrent networks, on the other hand, achieved a very good level of performance in sequential data related problems such as dynamic system modeling and natural language analysis.

There are mainly three different layers in neural networks:

1. The input Layer.
2. The hidden Layers: In this layer, the inputs received from the previous layer are processed. The hidden layer can contain more than one layer.
3. The output Layer: After having the data processed, it becomes available in this layer.

It can be noticed that the variables are represented by nodes. The network layers are distinguished via colors: red, green, blue, for input, hidden and output layers, respectively. Let us define X_0 and H_0 as dummy variables. They always take the value of one. The dummy variables are in the image below.



Neural Network layers

1- Input Layer

The Input layer rule is to interact with the external environment. It deals with the inputs coming from the data sources. The input is then transferred to the hidden layer. Each input represents an independent variable which affects the output of the network. The inputs are some features such as: Height, Age, weight, image data or hours of sleep.

2- Hidden Layer

The hidden layer is the intermediate layer, lying between the input and the output layers. It consists of the group of neurons that have activation functions applied on it. The job of the hidden layer is to deal with and process the inputs received from the previous layer. Therefore, its responsibility is to extract features of interest from the input data. The hidden layer can include more than one hidden layer. So that, one of the problems to be considered is the choice of the number of the hidden layers depending on the learning category. If the input data can be linearly separated, then we can skip the hidden

layer. This is because the activation function can be designed to input a layer that is able to solve the problem. In the problems that have complex decisions to make, three to five hidden layers can be used. In the case of multiple hidden layers, the symbol H will have a superscript that indicates the hidden layer number. A neural network with more than one hidden layer is also referred to as deep learning or a deep neural network.

3- Output layer

The output layer is the stage where we got the values that we want to predict. A good benefit of the neural networks is that the output pattern can be traced back to the input layer. This means that there should be a logical relation between the number of neurons in the output layer and the type of work performed by the neural network. The number of neurons in the output layer is related to the intended use of the neural network.

Latent variables

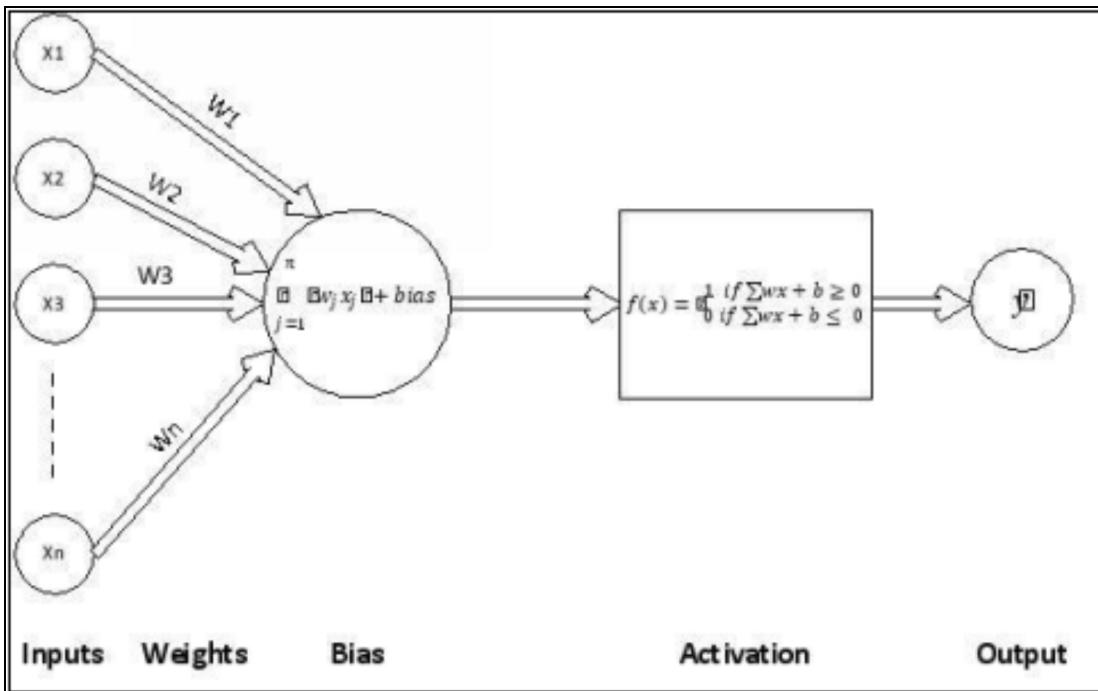
In the neural networks, the latent variable term is common. However, it is not very new to the reader of this book. Latent variables are functions of the inputs. This relation is expressed by edges, in neural networks. Each edge flows from left to right, which means that every variable that receives an edge is a function of the variable from which the edge comes and also the weight associated with this edge.

The Flow in the Neural Network

Every neuron, in the ANN, represents an activation node. The activation node is connected to the input nodes in order to apply learning algorithms to calculate the weighted sum. The weighted sum is then passed to an activation function which leads to predict the results. Here where we have the concept of “a perceptron”

arises. Perceptron means things that take many inputs that lead to one output.

Every input X has a weight of W . The importance of weights is that they are indications of the significance of inputs. In other words, as a weight goes higher, the input has a greater contribution to the output results, and vice-versa. The bias can be compared to the constant B in the famous line equation: $Y=AX+B$, as it enables us to adapt the line either up or down to have better prediction results. As known from simple elementary algebra rules, if we do not have the constant B , the line will always pass through the origin point and in our case in machine learning, we will have poor results.



Detailed Neural Network

This sum goes then through an activation function. The activation function forms a gate that opens and closes. The activation function can give ones or zeros, judge based on a threshold or make probabilities. Regarding the probability part, it may utilize the sigmoid or the rectified linear function.

The activation function output is the waited, predicted, output from the neural network. It is indicated as \hat{y} in the figure above. \hat{y} may be regression, binary or classification values, depending on the nature of the dataset and the required output.

Activation function

In this section, we will make a quick review for common activation functions used for the ANN.

- *Threshold or step function:* If the sum of the weights and inputs is below a specific threshold it is zero and if it is above, it is one. Recall the gate analogy we talked about.
- *Sigmoid function:* It is the most widely used activation function. It looks like the threshold function but it is smoother. Sigmoid activation function is suitable for making probabilities. This means that the output range of the function will be always between zero and one. For example, if we are investigating a picture, the output will be the probability whether it is for a cat or a dog?!
- *Hyperbolic Tangent:* This function is a stretched version of the sigmoid function, as it ranges from -1 to 1. This function is chosen when we want a deeper gradient and steeper derivative. Therefore, the choice between the sigmoid or the hyperbolic tangent functions depends on the gradient strength requirements.
- *Rectifier Linear Unit (RELU):* This function is normally applied in the reinforcement learning application. From its name, it is a linear function that has been rectified. This means that it has the value of zero in the negative domain and it increments linearly, i.e. if for a positive input x , it gives an output x and it is zero otherwise. One major advantage of RELU is its simplicity compared to the hyperbolic tangent and the sigmoid functions. This is a main point to consider in designing the deep neural networks.

According to the characteristics of the function we are trying to approximate, we can choose the activation function that achieves faster approximation, which means faster learning process. The sigmoid function, for example, shows a better performance in classification problems than the RELU. This means a faster training process and also a faster convergence. One may use his own custom function. However, if the nature of the function we are trying to learn is not very clear, then it is preferable to start with RELU as a general approximator and then work backwards.

Working Example

In this section we are giving an example that translates some of the concepts that explained in the previous sections. This means that we have the weights optimized as seen in the figure. The input layer contains a group of features: age, distance to the hospital, gender and income. The output on the other hand, is the variable that is dependent on the input features. It is the probability that a person will be hospitalized. If we take the age, for example, we can state that the older the person, the more likely he/she will be hospitalized. In terms of gender, statistically men are more likely to be hospitalized than women. The data collected can be utilized to make several predictions depending on the input features and this is the realization of the neural networks job. The hidden layer in this example was considered as a black box which contains the neurons and weights that result in the hospitality probability. The machine can be trained to predict the relation between the distance to the hospital and the probability of being hospitalized. Therefore, the neural network has to try first to find patterns in the data and figure out the relations between these different patterns. This is a kind of understanding data before making other processing or decisions. Once trained, we are able to input different features extracted from the targeted dataset. For example, one record can be something like that: Age is 65 years old, gender is female, moderate distance to the hospital and high income. The neural network is a magical method

that expects and predicts the desired results after processing the features in a smart way that mimics the human neuron.

The neural network estimates the different combinations of the input features to identify patterns in the dataset based on the neural network architecture. It may be seen as a black box, but a deeper look allows us to see patterns amongst the weights, inputs and hidden layers.

Neural Network, Hospitality Probability Example

The learning process (How the weights work)

We normally need machines to learn since we have large amounts of data to be processed and analyzed. Therefore, if we have a large dataset, we feed its inputs into the neural network to make predictions. The first step is to determine the neuron's weights. They can be determined based on initial assumption or they can be predefined based on the application features and the required output.

The normal flow of the neural network is towards the output \hat{y} . That is, the input features are fed to the hidden layer where the calculations are performed. Then, the network flow continues until we have the desired output. This kind of flow is known as: *Forward Propagation*. At this point, the smart error handling appears. The resulting \hat{y} is compared to the actual value of y . The target is to make the difference between them as small as possible, i.e. minimizing the error. This can be clearer if we imagine the analogy of a child learning math. If he answered a specific question with 8 while the accurate answer is 5, then we have an error of 3. In this case, the calculations have to be re-performed until we converge as close as possible to 8.

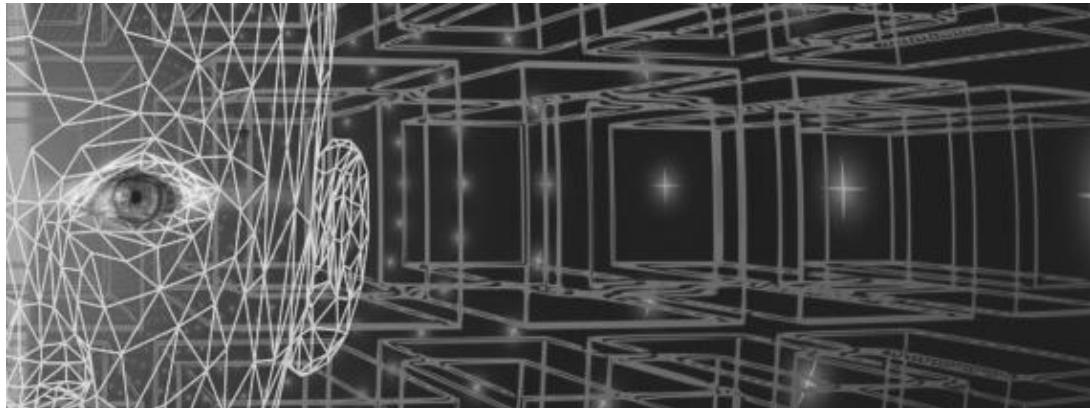
The neural network utilizes the weights in its task to minimize the resulting error. It reduces the weights of the neurons that make a significant contribution to the error. This process is known as: *Back Propagation*. This is because in this tuning process we travel back from the output to the neurons and input, in order to locate where exactly the error happens. Here is where the importance of the activation functions comes to the surface. The activation functions are differentiable and this helps in performing the backpropagation process through the neural network; by computing the gradients. Hence the weights are adjusted. The simultaneous adjusting process of the weights continues until we have a neural network output that is close to the actual output. The weights are slightly adjusted in each iteration to have a smaller error at each run. The process is repeated for all the inputs and outputs possibilities of the training dataset until the error is significantly small and acceptable by the application.

Chapter 12 Summary

- Artificial Neuron Networks are typically applied in auto-driving cars, image recognition and processing, recommender and consult systems.
- ANNs are made up of three layers: the input layer, the hidden layer, and the output layer.
- ANNs are made up of several neurons. Each of the neurons is an activation node linked to the input nodes to successfully apply the learning algorithms.
- The operational aspects of ANNs mimic the core functionality of the human neural network.
- This approach establishes a basis for deep learning, which facilitates effective machine learning approaches and subsequent application of ANNs.

Chapter 13:

Data Scrubbing



Similar to Swiss or Japanese watch design, a good machine learning model should run smoothly and contain no extra parts. This means avoiding syntax or other errors that prevent the code from executing and removing redundant variables that might clog up the model's decision path.

This push towards simplicity extends to beginners developing their first model. When working with a new algorithm, for example, try to create a minimal viable model and add complexity to the code later. If you find yourself at an impasse, look at the troublesome element and ask, “Do I need it?” If the model can’t handle missing values or multiple variable types, the quickest cure is to remove the troublesome elements. This should help the afflicted model spring to life and breathe normally. Once the model is working, you can go back and add complexity to your code.

What is Data Scrubbing?

Data scrubbing is an umbrella term for manipulating data in preparation for analysis. Some algorithms, for example, can't read certain data types or return an error message in response to missing values or non-numeric input. Variables, too, may need to be scaled to size or converted to a more compatible data type. Linear regression, for example, analyzes continuous variables, whereas gradient boosting asks that both discrete (categorical) and continuous variables are expressed numerically as an integer or floating-point number.

Duplicate information, redundant variables, and errors in the data can also conspire to derail the model's capacity to dispense valuable insight.

Another potential consideration when working with data, and specifically private data, is removing personal identifiers that could contravene relevant data privacy regulations or damage the trust of customers, users, and other stakeholders. This is less of a problem for publicly-available datasets but something to be mindful of when working with private data.

Removing Variables

Preparing the data for further processing generally starts with removing variables that aren't compatible with the chosen algorithm or variables that are deemed less relevant to your target output. Determining which variables to remove from the dataset is normally determined by exploratory data analysis and domain knowledge.

In regards to exploratory data analysis, checking the data type of your variables (i.e. string, Boolean, integer, etc.) and correlation between variables is a useful measure to eliminate variables. Domain knowledge, meanwhile, is useful for spotting duplicate variables such as country and county code, and eliminating less relevant variables like latitude and longitude, for example.

In Python, variables can be removed from the dataframe using the **del** command alongside the variable name of the dataframe and the title of the column you wish to remove. The column title should be nested inside quotation marks and parentheses as shown here.

```
del df('latitude')
```

```
del df('longitude')
```

Note that this code example, in addition to other changes made inside your notebook, won't affect or alter the source data. You can even restore variables removed from the development environment by deleting the relevant line(s) of code. It's common to reverse the removal of features when retesting the model with different variable combinations.

One-hot Encoding

One of the common roadblocks in data science is a mismatch between the data type of your variables and the conditions of the model's algorithm. While the contents of the variable might be relevant, the algorithm might not be able to read the data in its default form. Text-based categorical values, for example, can't be parsed and mathematically modeled using general clustering and regression algorithms.

One quick remedy involves re-expressing categorical variables as a numeric categorizer. This can be performed using a common technique called one-hot encoding that converts categorical variables into binary form, represented as "1" or "0"—"True" or "False."

```
import pandas as pd
```

```
df = pd.read_csv('~/Downloads/listings.csv')  
df = pd.get_dummies(df, columns = ['neighbourhood_group',  
'neighbourhood'])  
df.head()
```

Run the code in Jupyter Notebook.

One-hot encoding expands the dataframe horizontally with the addition of new columns. While expanding the dataset isn't a major issue, you can streamline the dataframe and enjoy faster processing speed using a parameter to remove expandable columns. Using the logic of deduction, this parameter reduces one column for each original variable.

The Python interpreter can deduct the true value of each variable without referring to the expendable columns, based on the false argument of other

variables. In statistics, this concept is known as multicollinearity and describes the ability to predict a variable based on the value of other variables.

To remove expandable columns in Python we can add the parameter **drop_first=True**, which removes the first column for each variable.

```
df = pd.get_dummies(df, columns = ['neighbourhood_group', 'neighbourhood'], drop_first = True)
```

Drop Missing Values

Another common but more complicated data scrubbing problem is deciding what to do with missing values. There are numerous quick fixes to this problem, but let's first review the code for inspecting missing values.

df.isnull().sum()

Inspecting missing values using `isnull().sum()`

Using this method, we can obtain a general overview of missing values for each feature. From here, we can see that four variables contain missing values, which is high in the case of **last_review** (3908) and **reviews_per_month** (3914). While this won't be necessary for use with all algorithms, there are several options we can consider to patch up these missing values. The first approach is to fill the missing values with the average value for that variable using the **fillna** method.

```
df['reviews_per_month'].fillna((df['reviews_per_month'].mean()), inplace=True)
```

This line of code replaces the missing values for the variable **reviews_per_month** with the mean (average) value of that variable, which is 1.135525 for this variable.

We can also use the **fillna** method to approximate missing values with the mode (the most common value in the dataset for that variable type). The mode represents the single most common variable value available in the dataset.

```
df['reviews_per_month'].fillna(df['reviews_per_month'].mode(), inplace = True)
```

In the case of our dataset, the mode value for this variable is ‘NAN’ (Not a Number), and there isn’t a reliable mode value we can use. This is common when variable values are expressed as a floating-point number rather than an integer (whole number).

Also, the mean method does not apply to non-numeric data such as strings—as these values can’t be aggregated to the mean. One-hot encoded variables and Boolean variables expressed as 0 or 1 should also not be filled using the mean method. For variables expressed as 0 or 1, it’s not appropriate to aggregate these values to say 0.5 or 0.75 as these values change the meaning of the variable type.

To fill missing values with a customized value, such as ‘0’, we can specify that target value inside the parentheses.

```
df['reviews_per_month'].fillna(0)
```

A more drastic measure is to remove missing values altogether. This action becomes necessary when the mean and mode aren’t reliable and finding an artificial value is not feasible as well. There are two primary methods for removing missing values. The second method is the dropna method which automatically removes columns or rows that contain missing values.

```
df.dropna(axis = 0, how = 'any', thresh = None, subset = None, inplace = True)
```

As datasets typically have more rows than columns, it’s best to drop rows rather than columns as this helps to retain more of the original data.

In summary, the management of missing values is highly dependent on their data type and frequency.

Dimension Reduction

Dimension reduction, known also as descending dimension algorithms, is a method of transforming data to a lower dimension, which can help to lessen computational resources and visualize patterns in the data.

Dimensions are the number of variables describing the data, such as the city of residence, country of residence, age, and gender of a user. Up to four variables can be plotted on a scatterplot but three-dimensional and two-dimensional plots are easiest for human eyes to interpret.

The goal of descending dimension algorithms is to arrive on a minimal set of variables that mimic the distribution of the original dataset's variables. Reducing the number of variables makes it easier to recognize patterns, including natural groupings as well as outliers and anomalies.

It's vital to note that dimension reduction isn't a case of deleting columns but, rather, mathematically transforming information contained in those columns in such a way that the information is captured using fewer variables (columns). If, for example, we look at house prices, we might find multiple correlated variables (such as house area and postcode) that we can merge into a new variable that adequately represents those two variables. By applying dimension reduction before running the core algorithm, the model will run faster, consume less computational resources, and may actually provide more accurate predictions.

Another side benefit of this technique is the opportunity to visualize multidimensional data. Given the maximum number of plottable dimensions for a scatterplot is four, and two or three dimensions is ideal (the fourth dimension is time), descending dimension algorithms can be used to streamline a dataset with more than four dimensions into four or fewer variables and project the synthetic variables onto the visual workspace of a scatter plot.

Three-dimensional scatter plot with three variables

The ability to visualize relationships and patterns on a scatter plot is useful for both explanatory and exploratory graphics. Exploratory graphics are typically generated on-the-fly to aid internal understanding when analysis is in progress, whereas explanatory graphics are delivered to an external audience in the post-analysis stage.

Bear in mind, though, that streamlining a multidimensional dataset into four or fewer variables (suitable for a scatterplot) isn't a prerequisite for machine learning. Models that analyze the input of 20 variables, for instance, can't be visualized on a scatter plot but can still be processed by the model to identify patterns and aid decision-making by producing a binary output, such as 1 (True) and 0 (False) or another form of output.

Pre-model Algorithms

As an extension of the data scrubbing process, unsupervised learning algorithms are sometimes used in advance of a supervised learning algorithm to prepare the data for prediction modeling. In this way, unsupervised algorithms are used to clean or reshape the data rather than derive actionable insight.

Principal Component Analysis

One of the most popular dimension reduction techniques is principal component analysis (PCA). Known also as general factor analysis, PCA is useful for dramatically reducing data complexity and visualizing multidimensional data in fewer dimensions. The practical goal of PCA is to find a low-dimensional representation of the dataset that preserves as much of the original variation as possible. Rather than removing individual features from the dataset, PCA recreates dimensions as a linear combination of features called components, and then ranks components that contribute most to patterns in the data, allowing you to drop components that have the least impact on data variability.

PCA deconstructed

Above are four horizontal axes. The first two axes measure the x and y values denoted in the original data plot. The second two axes measure the distance from x and y values when rotated 90 degrees.

These new axes provide us with the first two components of this dataset. The new x-axis serves as Principal Component 1 (PC 1) and the new y-axis as Principal Component 2 (PC 2). Using PC1 and PC2 (seen on the third and fourth axis), we can see a new range of variance among the data points. The variance in PC1 has expanded in comparison to the original x values (seen in the first axis). Meanwhile, the variance in PC2 has shrunk significantly as all the data points are close to zero and virtually stacked on top of each other.

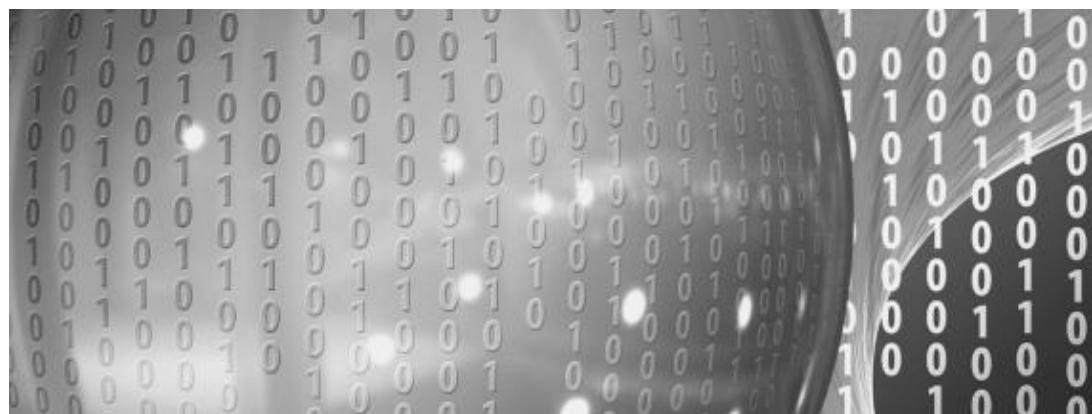
Given that PC2 contributes the least to overall variance, we can focus our attention on studying the variance contained in PC1. While PC1 doesn't contain 100% of the original information plotted on the scatterplot, it captures the variable relationship that has the most impact on data patterns while minimizing computational resources.

Chapter 13 Summary

- Data scrubbing is the pre-analysis process of manipulating data.
- Enhancing the compatibility of the data to meet targeted answers is achieved by removing incompatible variables.
- Efficient and effective data science requires variables' data type to match the conditions of the model's algorithm. One-hot encoding can be applied to establish this match.
- Machine learning does not necessarily require a streamlining of a multidimensional dataset into four or fewer variables.
- Reshaping and cleaning data is necessary in machine learning. Unsupervised algorithms can be employed to fill this role.

Chapter 14:

Syntax



We have just gotten in touch with the various types of data in Python.

Now, we should find out which commands can be used.

First, we will look into the indentation; second, we will explain the control flow statements – if, for, while, break, continue, and pass.

Indentation

The indentation is one of the most characterizing syntactic elements of Python.

Let's look at a classic example:

```
def perm(l):
    # Compute the list of all permutations of l
    if len(l) <= 1:
        return [l]
```

```
r = []
for i in range(len(l)):
    s = l[:i] + l[i+1:]
    p = perm(s)
    for x in p:
        r.append(l[i:i+1] + x)
return r
```

As you can see, it is easy to see where the code ends and where the following code line starts.

The indentation is used to indicate a block of code. Obviously you must indent each block with the same amount of spaces. In Python, we use indentations to indicate to what block a particular string of code belongs.

Another great feature is that the universal block closing rules are universal, so we do not need to specify END, ENDIF, FI, parentheses or brackets, WEND, ENDWHILE, NEXT, LOOP, END PROCEDURE OR END FUNCTION.

We only need to respect the INDENTATION.

The IF statement

What is a CONTROL FLOW STATEMENT ? It's a feature that allows the program to make decisions, depending on the situation, of reading the code in a particular order. We can control the statement execution using some control flow tools.

The IF instruction is probably the most used command among all programming languages.

We can use IF to check a condition.

Here is an example of how IF works:

We run a block of statement only if the condition IF is true; otherwise, we operate another statement.

Example of IF statement syntax:

```
x = int(input("Please enter an integer: "))
```

```
Please enter an integer: 42
```

```
if x < 0:
```

```
    x = 0
```

```
    print('Negative changed to zero')
```

```
elif x == 0:
```

```
    print('Zero')
```

```
elif x == 1:
```

```
    print ('Single')
```

```
else:
```

```
    print('more')
```

More

We can observe ":" that we used to end the test that controls the program flow.

You can insert as many ELIF AS you need; in the example above, the last Elif is not obligatory.

The FOR statement

The For instruction allows us to define iterations. To decide how many iterations we need and on which element we have to perform the iteration, we must use any object we can consider.

Traditionally used to repeat a string of code an exact number of times.

We have already seen many sequence examples: lists, tuple, strings. These are all data we can use to iterate with For:

```
for c in "abc":
```

```
    Print c
```

```
A  
B  
C  
>>>
```

How can we execute a simple iteration a certain number of times without having the right sequence?

We could use “range”

```
for n range (4):
```

```
    Print n
```

```
    1  
    2  
    3  
    4
```

```
>>>
```

*With the command “range”, we can also choose the start and the end of an iteration.

“For” also allows us to execute a cycle on more variables at the same time.

This is possible due to the fact that the “items” method in a dictionary gives a list of TUPLES composed by their own key and value.

“FOR” establishes a cycle on every element in this list and gives to the variables KEY and VALUE the values we have in each TUPLE.

The WHILE statement

The WHILE instruction is similar to the FOR instruction. The only difference is that the iteration is not based on a fixed sequence but executes the operation as long as the given condition is true.

Here is an example of how While works:

```
x = 1
```

```
while x < 7:
```

```
    Print x
```

```
    X += 1
```

```
    1
```

```
    2
```

```
    3
```

```
    4
```

```
    5
```

```
    6
```

```
>>>
```

In the example above, (the increment statement), is executed until the count remains below 7.

If a condition remains True, we have a so-called “Infinite Loop”. In that case, you should press CTRL + C to close the program.

The Break and Continue

Break and Continue are used to stop an iteration to pass to the next one.

You will often use BREAK in FOR LOOP and WHILE LOOP as in these classic examples:

For var in sequence:

```
# codes inside for loop
```

If condition :

```
    Break
```

```
# codes inside for loop
```

```
#codes outside for loop  
While test expression:  
# codes inside while loop  
If condition:
```

 Break

```
# codes inside the loop  
# codes outside while loop
```

Now that we have gotten in touch with the instructions BREAK and CONTINUE, let's step back to FOR and WHILE because they have another peculiarity, the instruction: ELSE.

It was an IF clause, but "ELSE" is also very useful with cycles due to the fact that all the instructions we will write in ELSE will be executed at the cycle end (if we do not end the cycle first with a BREAK).

Example of if – else

Num = 8

If num >= 5:

 Print("positive or zero")

Else

 Print("negative number")

*try with different numbers as well

With this program, you can check if the number is positive, negative, or zero.

Chapter 14 summary

- In Python, indentation is employed as one of the primary characterizing syntax elements.
- Decision-making in the Python program is allowed primarily by the CONTROL FLOW STATEMENT.
- Iterations can be achieved through the FOR statement. The WHILE statement is similar to the FOR instruction. However, the iteration is not based on a fixed sequence.
- Iteration is stopped by the ‘Break and Continue.’

Chapter 15:

Auto Learning Myths



The different myths of machine learning according to Pedro Domingos can be summarized below:

Machine learning can only discover correlations, not cause-and-effect relationships.

One of the main myths about machine learning is that it can not discover cause-and-effect relationships. One of the most common types of machine learning is experimenting with different actions and observing the consequences - the essence of causal discovery. The ability of artificial intelligence to discover the prediction of cause-and-effect relationships is used in the e-commerce industry. For example, an e-commerce website can try many different ways to present a product and choose the one that drives the most purchases. You have probably participated in thousands of these experiments without knowing it. And cause-and-effect relationships can be discovered even in certain situations where the experiments are irrelevant, and all that the computer can do is examine past data.

Machine learning cannot predict events never seen before.

This is commonly called the black swan's myth. It is often misunderstood that if something had never happened before, your predicted probability should be zero - what else could it be? In contrast, machine learning is the art of predicting rare events with great accuracy. If A is among the causes of B and B is among the causes of C, A can lead to C, even if we have never seen it before. Every day spam filters correctly report new spam created. Black swans and the real estate crisis of 2008 were widely anticipated - not just the imperfect risk models that most banks used at the time.

The more data you possess, the more likely you are to make meaningful predictions. Supposedly, the more phone-based the NSA's analysis, the more likely it was to designate an innocent person as a potential terrorist because he accidentally equated a terrorist detection rule. Exploiting more attributes of the same entities may increase the risk of hallucination, but machine learning experts are very good at minimizing it. On the other hand, extracting more entities with the same set of attributes decreases the risk because the rules learned in this way will be better managed. And some learning algorithms can find models involving multiple entities, which makes them even more robust: a person filming the New York City Hall may not be suspicious, and another purchase of large amounts of ammonium nitrate may not be; but if the two are in communication, the FBI should perhaps take a look, just to make sure it's not a bomb plan.

Machine learning ignores pre-existing knowledge. Experts in many areas of machine learning first sought the "blank" approach to the learning algorithms they know. True knowledge is the result of a long process of reasoning and experimentation, which you can not imitate by running a generic algorithm over a database. But not all learning algorithms begin with a blank; some use the data to refine a pre-existing body of knowledge, which can be quite complex, provided that it is coded so that the computer can understand it.

Models learned by computers are incomprehensible to humans. This is, of course, a subject of concern. How can we believe a learning algorithm's recommendations if it is a black box? Some types of models are very difficult to understand, such as the deep neural networks that are responsible for some of the most exceptional successes of machine learning

(like recognizing cats in YouTube videos). But others are quite understandable, as the diagnostic rule of skin cancer that we have seen previously.

All of these myths are pessimistic in that they assume that machine learning is more limited than it is. But there are also some optimistic myths:

The simpler models are more precise. This belief is sometimes related to Occam's razor, but the razor says that simpler explanations are preferable, not why. They are preferable because they are easier to understand, memorize and reason. Sometimes, the simplest assumption compatible with the data is less precise for the prediction than a more complicated hypothesis. Some of the most powerful learning algorithms generate models that seem to be designed for free - sometimes even added after perfectly adjusting the data - but that's how they outperform the less powerful ones.

Models discovered by computers can be obtained using the nominal value. If a learning algorithm indicates the rule for the diagnosis of skin cancer, the rule is very precise (in that almost all the moles that correspond to it are tumors), that does not necessarily mean that you should believe it. A slight change in the data can cause the algorithm to induce a very different - but equally accurate - rule. Only reliably induced rules, despite random variations in the data, can be interpreted as such, rather than just predictive tools.

Machine learning will someday give rise to superhuman intelligence. According to daily reports of AI's progress, it's easy to believe that computers are about to see, talk and reason, just like us, after which they will quickly leave us dusty. We have come to a long distance in the first 50 years of AL, and machine learning is the main cause of its recent success, but we still have a long way to go. Computers can do a lot of very fine jobs, but they still have no sense, and no one knows how to teach them.

As a result, machine learning is both more powerful and less powerful than we suppose. What we do depends on us, provided we start with a precise understanding.

Chapter 15 summary

- The novelty and disruptive nature of machine learning have led to significant auto learning myths.
- Machine learning myths:
 - Machine learning cannot be predictive in its analysis.
 - Machine learning can only establish correlations, not cause-and-effect interactions.
 - Machine learning is not based on pre-existing knowledge and bias.
 - Models learned by computers are beyond the comprehension of humans.
 - Machines will someday in future lead to superhuman intelligence.
 - The simpler the models the more precision they offer.

Conclusion

The next step is to make the best use of your new-found wisdom on the cutting-edge technologies of today that have generated the powerhouse, that is the Silicon Valley. The day machines will be able to operate independently with no human assistance whatsoever may not be as far as you think. Today, machine learning and artificial intelligence have given rise to sophisticated machines that can study human behavior and activity to identify underlying human behavioral patterns and precisely predict what products and services consumers are interested in.

Businesses with an eye on the future are gradually turning into Technology companies under the façade of their intended business model. The smart and savvy customers today can be easily swayed by modern companies with a whimsical edge offering consumers a unique, rich, and engaging experience. It is getting increasingly challenging for traditional businesses to retain their customers without adopting one or more of the cutting-edge technologies explained in this book.

With an understanding of how much is at stake for your business and how you can position yourself to not only retain existing customers but also interest more prospective customers that will eventually be converted to a paying consumer, you are poised to take your business to new heights. Start with one step at a time and understand the current status and challenges of your business, then mindfully use the power of data science technologies like artificial intelligence, machine learning, data mining and big data

analytics, in your company to be able to enhance human abilities without undermining the human intelligence and creating new high paying and rewarding jobs.