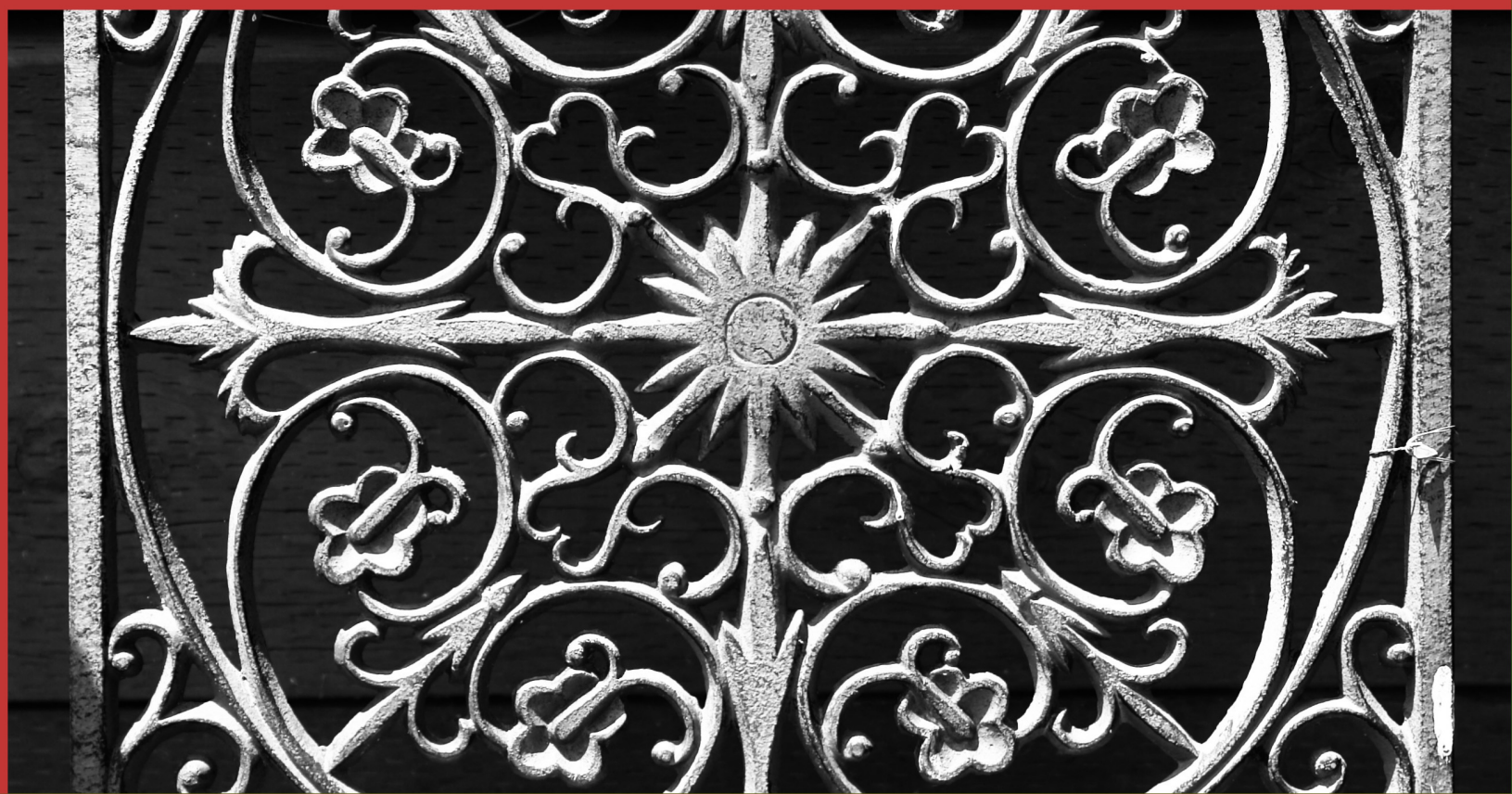
Hacking Python 3

A Guide to Scanning, Networking, Information Gathering

Sanjib Sinha



Hacking Python 3

A Guide to NMAP Scanning, Networking and Information Gathering

Sanjib Sinha

This book is for sale at http://leanpub.com/playwithpython

This version was published on 2018-03-29

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Dedication

To Kartick Paul.

Without his persistent help I couldnâ€™t have written any technology book.

Cover Design: Kaberi Sinha

Art Work: Amitakkhar Deb

Epilogue

This book requires a little bit programming knowledge. You don't have to belong to the advanced level but you need at least intermediate python knowledge to identify with few concepts. For intermediate python learners I have discussed few essential concepts in the part two.

For absolute beginners I'll advise to learn python basics and then come back to this book.

It'll not be the first time that you've learned hacking technique along with python programming language. There are many good books available in the market. They have discussed role of python in ethical hacking and I hope a little bit of research about it will not do any bad for your learning purpose.

This book has been divided in three parts.

In the first part we'll learn about the basic of networking, reconnaissance scanning and it'll also be a gentle reminder about the legal side of hacking – what you can do and what you cannot do with the help of this book.

In the second part we'll discuss and recapitulate our python knowledge. Not from the beginning. Assuming that you have intermediate python knowledge, we'll talk about few important concepts that are going to help us achieve our goal of learning ethical hacking.

In the last and final part we'll do some real hacking. We'll build python scanner that could scan any network, will do some cyber reconnaissance and gather important information about any target.

PART ONE: LEGAL SIDE, CYBER CRIME AND NETWORKING

Generally people associate the word ‘hacking’ with malicious digital activity. It is good to know that it’s not actually true. You may search the internet to find out the true meaning of the word ‘hacking’. In essence it stands for ‘creative curiosity’. When you hack a machine it means you want to know about it and want to modify about it.

Although there is a fallacy that loosely hangs about the word ‘hacker’. In general media use this word to convey a kind of negative meaning. True hackers, creative people and computer scientists reintroduce another word to counter this false legacy – cracker.

A ‘cracker’ is a criminal who uses this special knowledge for malicious activities that include stealing data from any source, tampering any system and attack it with wicked intentions.

On the other hand there are people who try to secure a system. This is called cyber security. You learn this craft because you want to learn that craft of securing a system. With this knowledge you’re not supposed to damage a system.

Legal Side of Hacking

As time goes by and we progress our old environment is also changing very fast. It has not been like before when we keep records by entering data into a big Log-Book and stack them one by one date wise. Now we keep data in computer and use special software to organize them. Many people don't go to market for buying instead they order them over the internet and payment is made by using credit or debit cards. The nature of crime has also changed as a result.

Criminals used to snatch your data physically before. They now try to seize it over the internet using computers. Now computers have become a new tool for business as well as for traditional crimes also. On the basis of which a term "CYBERLAW" comes into the fore. As an ethical hacker the first and most basic thing you should remember is "don't try to penetrate or tamper any other system without taking permission."

You may ask how I would experiment my knowledge. The answer is Virtual Box. In your virtual machine (VM) you may install as many operating systems as you want and experiment on them. Try everything on VM. Trying any virus on your virtual machine will not affect your main system. At the same time you will keep learning about the Malware, virus and every kind of possible attacks.

Few examples may give you an idea what type of computer crimes are punishable in our legal system.

If you use any software tool to generate credit card or debit card number then it is a highly punishable offense. It will invite fine of fifty thousand dollar and fifteen years of imprisonment (it varies from country to country, but it becomes stricter than ever). Setting up a bogus web site to take credit card numbers with a false promise of selling non-existent products is also highly punishable offense. Rigorous imprisonment and a hefty fine follow. I can give you several other examples that may invite troubles for you if you don't stay within law.

Remember you are an ethical hacker and you are learning hacking tools for protecting your or your client's system. For the sake of protection and defense you need to know the attack, exploit or penetration methods. Try your every single experiment on your virtual machine.

That is the RULE NUMBER ONE of ethical hacking.

Examples of Cyber Crime

As days pass by cyber criminals are changing their tactics. Very often they are caught and sometimes the trail remains untraceable with main culprits stay out of the hands of law. Here four examples of real life cyber crime instances have been given to make things clear to you. As a learner of ethical hacking you'll also know the range of thorough knowledge that is required to tackle such menaces.

Black Money and Bitcoin

Above all things, it now appears that government has thrown all its weights around the seizure of black money. IT officials claim that the amount of unaccounted money recovered since the announcement of demonetization on 8th November has crossed three thousand crores and they're counting for more. Before that government had also announced packages for black money disclosure that had ended on 30th September and till date the amount disclosed has been around 55 thousand crore rupees.

It's been one side of reality.

The other side is dark and sinister. It's not true and it can't be true that people who had amassed a huge amount of property and money without giving taxes for so many days have suddenly resurrected feeling guilty and come in line to disclose their unaccounted property. No, certainly not. The real picture is truly different and these people have tried every means to convert their black money into white and suddenly found that the classical way of conversion has been exhausted for a time being and they could not keep money in 500 and 1000 notes any longer. So they had to come up holding their hands up surrendering to the government whims cursing them viciously.

This generation of these corrupt people is sad people – in the words of Ernest Hemingway they are kind of 'lost generation' of black money hoarders – who either don't know about the recent technological boom or not believing in the intangible transaction of money and unhappily not acquainted with that. If they had tested blood in that way they would have certainly converted their black money into any available digital currency of which bitcoin is the most popular.

From the maze of dingy lanes of Khidirpur, Wattganj in Kolkata to the Delhi's Janak Puri or Model Town or Mumbai's Ghatkopar, the touts move clandestinely claiming they can convert any amount of money into bitcoin which one can get back as hard white cash after three to four working days. The exchange rate is fixed – one bitcoin equals 65 thousand rupees. The touts have their cuts. So if you give them one million black, you get back around 9,40,000 white that you can use any manner buying properties in foreign countries or so and so by exchanging bitcoin and this rate varies from one person to another.

First of all we need to know one thing. Bitcoin is a digital currency without a central authority. It has been shrouded by mystery all along because it exists only in virtual realm and is different from the physical currency like rupees, dollars or euros. In the developed world few countries including US have already legalized it and people see it as an alternative currency through which you can pay for goods and services as you do with the conventional currency. It's also termed as crypto-currency that uses cryptography for everything – creation to administration and overall security of the transaction.

To understand the mechanism of bitcoin you need to understand little bit about the dark web. Computer people also call it deep web. Why it is dark or deep? It's because, you cannot see it in

the available open layer of real web that is through Google or any available search engines and browsers. To visit a website we normally search it through Google or just type in the address on the web browser. You cannot reach to the dark web using that conventional route.

For a test, you may try this link: http://ow24et3tetp6tvmk.onion/, in your favorite search engine or you may paste it to any web browser. Google will come up with around five thousand indexes of which no one will link you to this site directly, actually they cannot. From the result of your searches you'd guess that something wrong has been associated with this link. The search results will also tell you to watch Youtube videos of how hidden Internet works and they will talk about the anonymous onion wallet but no one can take you to that web site directly. And if you type this link to the web address it says: server not found.

But that web site is there. It may sound strange but it's true that the web site is there and it's hidden in the deep or dark web. It can only be seen through the 'TOR' browser and in general, common people don't need to do that at all. To keep on the safer side of law you better not to try that.

The site is an exchanger of money – between the user and the bitcoin network where it is 'mined'. Let us see first what is there in that site first. It proudly announces the following promises: (I quote it verbatim).

```
“
Your anonymous Tor Bitcoin Wallet and Laundry
OnionWallet Features:
• Free Bitcoin Mixer! You will always get completely different Bitcoin on withdr\
awals with no "taint" to your receiving        address.
• Safe storage: we keep most of the bitcoin in secure encrypted offline storage.
• Protect your funds with a transaction PIN.
• Anonymous registration: We don’t need any private data.
• Very simple user interface, no complicated options and settings.
• Transaction fee: only 0.001 Bitcoin which is fully paid to the network.

Get started using Bitcoin in 2 simple steps:
• Register an account on OnionWallet and write down your username, password and \
optionally PIN at a secure place.
• Purchase Bitcoin to your Bitcoin address in your OnionWallet account using for\
 example one of the following exchange         services:
...
“
```

There are lots of spelling mistakes, repetitions of words and I didn't correct them keeping them as it is to get a feeling of dark web. This is a flash of dark web that handles bitcoin transaction and in the last parts of that site there are lots of links given where you can get your money transferred to Bitcoin through your normal web browsers. The touts go the designated places where operators

using Virtual Private Network or TOR to change the black money to bitcoin through these dark web sites.

I would like to say how bitcoin came to the market. There have been lots of controversies regarding the creation of this digital currency. No one knows who is behind it. He may be a single person or it might be a group of hackers. It first appeared in the year of 2009 but it rose to prominence like a rocket three years ago and since then it has been gaining ground having present market value that may vary from 5 to 10 billion dollar.

You may want to know how it works. It uses P2P technology. It stands for peer-to- peer. A conventional currency is normally issued by a central bank as part of its mandate to manage national monetary policy but bitcoin has no such central authority. It is a kind of completely decentralized currency and its all functions are carried out collectively by the network. Our government or central bank has no central policy regarding this trade but according to the intelligence bureau source the number of bitcoin users in our country stands around 25 thousand and it's increasing.

Finally, the question remains. What will happen to these people who had already stashed away unaccounted money into bitcoin or any other available cryptic digital currency after or before the surgical strike of demonetization? Do we have a system to keep a vigil on them? Probably not yet! So recovering them from the dark web seems to be a distant dream!

Biggest Data Heist

"Want to buy some profile dumps. Is there any? I'm ready to pay in Bitcoin."

After posting this message in three "TOR NETWORK" message boards, hidden in the dark web, my wait was over after nearly twenty hours when an "Anonymous" asked in a "TORCHAN" board (http://zw3crggtadila2sg.onion/): "How we know you're not an FBI scum"?

A real scummy question indeed! I wish I had a gentle reply!

It'll attract more slang designed only for the dark web if you can't prove that you're not really an "FBI scum". Other boards bore no fruit, at least not in twenty hours deadline I had set knowing that it was too short.

I sneaked into dark web to get some tips about the 'biggest data heist' that had shaken the web world few months back where one billion 'Yahoo!' user profiles were stolen and sold at least to three parties for a hefty price and according to the security farms the hoaxers were still hanging around for some final dollars. Is anyone still there lurking behind the darkness to sell some more staff? I just tried my luck to trace. I wish I had some luck and some lead.

Waiting for twenty hours is no time in dark web and you need to wait for long and take every precaution possible to hide your IP before moving towards any illegitimate transaction! Considering such transactions the year 2016 saw the record rate of increasing cyber attacks, data theft and the Yahoo data breach was the biggest among them. It is not only the biggest data heist in web history but it has some more grim implications and cataclysm for future generation who are 'digitally

connected'. The disaster has just begun. It's personal and financial. Forget about the rest of the world, in India alone the cyber crimes registered under Indian Penal Code rocketed up to nearly fifty percent in the last year! When a giant organization loses user profile database because of hacking the worst hit are those individuals who don't know the 'ABCD' of cyber security. They instantly become soft targets for Phishing attacks and social engineering and the news of individual financial losses are rarely published.

Actually anyone could buy anything in the dark web through the 'TOR NETWORK' and the insatiable appetite could titillate from 'anything' to 'anybody'. And that was the reason why after nearly three years those stolen data were resurfaced in the dark web marketplace by a black hat hacker having pseudo name 'Tressa88'.

It started the same way in one fine morning in March, last year. The date was 25.03.2016. The time – showing in the forum board was - 05.28. It must have been early morning in Russia when the black hat hacker "Tressa88" logged in and hinted about the "Yahoo!" data dumps for the first time. Primarily the language appeared as Russian indeed and it was a Russian message board running undercover – at least from its apparent mystifying appearance.

Who is 'Tressa88'? The black hat hacker claimed, 'I am a very old inhabitant of the network :)'; and that was true. Why the name 'Tressa'? Tressa88 announced in a chat, it is not his real name because that is just the name of 'a whore from Australia.' These are minor information. The director of Eastern European research and analysis for the security firm Flashpoint Intel Andrei Barysevich claimed that behind the alias Tessa88 there are actually two people, perhaps a female and a male. It's quite likely that behind this alias an underground group lurks in shadows.

"Tressa88" posted the first message in Russian and the first few lines in English translation were like this:

```
[Only registered and activated users can see links]
For a review 10% discount
Important:! All goods in EMEIL format; PASS or a HASH
Only fresh and 100% private boxes
Shell for mailing. With the criteria of sending and delivery pisma._3 $ _1
1) VK.COM _137.000.000 email accounts; pass_pfone; pass
2) MOBANGO_6.000.000 entries id: email; pass
3) MYSPACE_380.000.000 id records: mail: hash
…
```

Since we're not registered, we could not see the links that user "Tressa88" had posted and boasted about the 'dumps'. But as usual they were either 'onion' links for Bitcoin wallet or belonged to any private network and they were meant for the dark web transactions only.

If you go through the whole message, you would not probably find any mention of 'Yahoo!' data at first glance. Instead "Tress88" said that he was ready to sell user profile data from other sites like MYSPACE, MOBANGO etc. This part of the message was important as on the last line of the list he hinted about 'many other sites'.

```
1) VK.COM _137.000.000 email accounts; pass_pfone; pass
2) MOBANGO_6.000.000 entries id: email; pass
3) MYSPACE_380.000.000 id records: mail: hash
4) QIP-133.000.000 records
….
10) And many other sites. Specify in Message
```

He was ready to sell from MYSPACE and other sites including Russian social media. And the largest data belong to MYSAPCE – it was almost 4 hundred million user profile database. But there was a hint that he had something special to announce at the end of the page.

After his first post within eight hours "Tressa88" got an answer from a user called "Mr.Mongo". And then afterwards users like "Fifty", "Edgar" and "ionline" joined the conversation asking questions about getting the data and in no time that soon became sensational international news and because of that "Yahoo!" had to admit on September 22: "Yahoo believes that "at least" 500 million user accounts were stolen, which would make it the biggest breach of all time, bigger than the MySpace breach of 427 million user accounts."

Remember the list, 'Tressa88' posted in that message board. MYSPACE was among them and the number matched with the 'Yahoo!' statement.

As their conversation moved forward, "Tressa88" kept writing like this on 28th March:

```
Tessa88
03/28/2016, 06:12
Reviews have to exploit. kardklub Fak and others.
Dame forward to nick a turnip.
Garant welcome.
```

There are many other dumps, full format, and a lot of different dating mail; pass Look at the date "Tressa88" wrote his second message. It was March 28 and on September 22, after nearly six months, "Yahoo!" again published a long statement on their website: "We have confirmed, based on a recent investigation, that a copy of certain user account information was stolen from our network in late 2014 by what we believe is a state-sponsored actor. The account information may have included names, email addresses, telephone numbers, dates of birth, hashed passwords (the vast majority with bcrypt) and, in some cases, encrypted or unencrypted security questions and answers….." At the same breadth 'Yahoo!' frantically assured their users that the stolen data did not contain credit card and debit card details. Was this assurance enough to have the peace of mind? No! Any computer literate person knows that any type of profile information may invite Phishing attacks much later once black hat hackers get that profile data in their possession. The hackers will definitely use it for social engineering. Both these hacking tricks involve disguise as the hacker may appear as a friend or the user's known bank source. An innocent user will never suspect and once he clicks a link sending by the known 'bank' source, the link leads to a fake login page that collects user's login credentials and delivers them to the attackers.

By the time when the whole world came to know about the biggest data heist in the history of web the cyber security people were in tizzy. The question tormented ever cyber security personnel. The 'Yahoo!' data breach occurred in 2013. Why 'Tressa88' took so many years to announce about the 'dumps'? What happened in between? Had they social engineered millions of people in between? Had they exhausted all potentials of Phishing attacks in between?

Later it came out that the stolen profile database included plenty of American military and security personnel who used to keep their official emails as recovering emails in their "Yahoo!" accounts. Was there really any 'state actor' functioning behind when that attack took place as 'Yahoo!' claimed? Did 'Tressa88' act as a front man for any foreign spy master? Or the black hat hackers simply sold the stolen data to any 'state actor'? Mystery deepened.

On 14 December "Yahoo!" again issued a statement: "Yahoo has identified data security issues concerning certain Yahoo user accounts. Yahoo has taken steps to secure user accounts and is working closely with law enforcement."

It's too late! The damage has already been done! You'll never get the exact figure of individual victims. Common people, who usually use the same username and password combinations in various sites, fall prey easily to the cyber criminals and these innocent people have a tendency to click every links sent by the fraudsters disguised as 'friends' and 'banks' without suspecting any fraud. In the last year's RSA Conference, 'Tripwire' conducted a survey. It asked 200 security professionals to express their concerns about the state of Phishing attacks and it came out that more than half (58 percent) of respondents stated their organizations had seen an increase in Phishing attacks in the past year. 'Verizon', in its 2016 Data Breach Investigations Report, noted that the growth of Phishing attacks in both frequency and sophistication poses a significant threat to all organizations.

Now, think about the common people who don't even know that clicking a link may spell bloody disaster for them, at least financially. Let us again forget about India. Consider the case of British supermarket giant Tesco Bank. Earlier November, last year, the consumer finance wing of Tesco Bank had to freeze their online operations after as many as 20 thousand customers had their money stolen from their accounts. They admitted in their website that nearly 40 thousand accounts had been compromised and half of those had lost their money. It's really difficult to diagnose the Phishing attack in isolation when someone steals money from a British account sitting many thousand miles away in a different country.

"All-in-Once-Checker," is a hacking tool, available in the market and it can check if hacked username and password combinations from one website work on another website or not! So, there are plenty of depressing possibilities that these black hat hackers stole the data much before and exhausted all the ways of making money and after that they decided to make some 'final dollars' and appeared after three years.

'Yahoo!' claimed in their statement that they found some 'state actors' behind the whole breach. By the middle of 2016, we came to know about one more hacker – 'Peace of Mind'. A few months after Tressa88 started selling through Russian underground forums, the same data resurfaced in TRDM (The Real Deal Market) – a dark web marketplace. This time the seller's name was 'Peace of Mind' and the hacker identified as male and a clear rivalry started brewing between them as Tressa88 later

told a security farm, 'Peace_of_mind [is] a fagot who takes undue credit'. According to 'Tressa88' this 'Peace of Mind' was his accomplice whom "I shared a dump for analysis! And he started selling it." 'Peace of Mind' made same allegations about 'Tressa88', "He stole [the hacked databases] from an old buddy long ago. And he started to sell them."

'Tressa88' accused 'Peace of Mind' had been cheating him since he gave Peace few database dumps to decrypt. 'Peace of Mind' slapped back, 'Tressa88' was a thief as one of his old friend had had the dumps originally but Tressa88 stole from him and started selling it. As soon as this saber-rattling died down other hackers started pelting the TRDM with 'Denial of Service' attacks. The black hat hacker community was angry because the 'dumps' were not up to the mark. The buyers were disgusted with low quality of 'dumps'. The reason was simple. They had hoped they could gain financially from those 'dumps' by duping common innocent people.

We are not in a position to decide whether this was a pure drama and they had made this story simply to show the investigators a different line of attack. All we know now that in last two three years countless people had their money stolen without knowing the sources of the fraudulent activities. All the time through the TOR NETWORK forums people want information about other people in exchange of money. What type of information? Let's have a look at the 'INTEL EXCHANGE' forum (HTTP:// rrcc5uuudhh4oz3c.onion). A user asks, 'im trying to get some info on a person. her name is '…..' Shes from sarasota florida, got 2 kids – '….' and '…..' she did me wrong so im looking for all the info I can get on her.' (I have kept the spelling unchanged and the names withheld). Another user asks about an 'Onion' site and the forum admin warns that the site might be a FBI honey spot. But these types of people who are asking for trivial help in the underground forum are not real concerns of the civilized world. What happened after 'Tressa88' and 'Peace of Mind' had started selling the 'dumps' was the perfect storm. The black hat hackers went for a 'Denial of Service' attacks on 'The Real Deal Market' (TRDM) and tried to shut it down simply because there had been a feeling among them that they were cheated by 'Tressa88' and 'Peace of Mind'. It was enough proof of how common people were the main target of those fraudsters.

For the time being we may conclude that the financial disaster has just begun. It has not completed the full circle yet. We have to wait another five or ten years to realize its full impact. From the 'Digital India' perspective we may also hope that government would organize more awareness campaign on how to spot a Phishing attack. Otherwise we'll give these new-age cyber criminals free rein to exploit our digital dream.

Chapter 3 – Hacking and Networking

A basic knowledge about Inter-networking is extremely important if you want to learn ethical hacking. As you progress and want to go deeper it is advisable to learn more about networking. Ethical hacking and inter-networking are very closely associated. As you progress through this book you will find words like “packet”, “switch”, “router”, “modem”, “TCP/IP”, “OSI” and many more.

The very first thing you need to know is: data travel through many layers. Ethical hackers try to understand these layers. Once they have understood the movement they either want to track and block the data or they want to retrieve data.

In this chapter we would very briefly see how internetworking models work. We will look into the different types of networking models. We will also know about the devices that comprise a network.

What Does Network Mean?

A Network is a collection of devices that are connected through media. One of the main characteristics of a network is: devices contain services and resources. Devices contain Personal Computers, switches, routers and servers among others. What they do basically? They send data and get data either by switching or by routing. Actually they connect users so that users ultimately get full data instead of getting it by pieces. So the basic services these devices provide include switching, routing, addressing and data access.

We can conclude that a network primarily connects users to avail these services. That is its first job. The second job is also very important. A network always maintains a system so that the devices allow the users to share the resources more efficiently.

Now a problem arises. Not a trivial problem is this. Hardware and software manufacturers don’t know each other. They belong to different countries and share diverse cultures. When the conception of networking first came into the fore it was found that hardware and software weren’t matching. As I said before a network is a collection of devices. These devices are mainly built of hardware and software that are talking in different languages.

To solve this problem a common network model with communication functions is needed so that dissimilar devices can interoperate.

The importance of internetworking models consists of few main concepts. First they encourage interoperability. Second they provide a reference through which data will be communicated. Third they facilitate modular engineering.

There are two types of internetworking models.

They are Open Systems Interconnection (OSI) reference model and Transmission Control Protocol/Internet Protocol (TCP/IP) model. Both models are widely used today. The Open Systems Interconnection (OSI) reference model was developed by the Internet Standards Organization (ISO) and it has seven layers in all. The layers are as follows: application (layer 7), presentation (layer 6), session (layer 5), transport (layer 4), network (layer 3), data link (layer 2) and physical (layer 1).

Let us very briefly try to understand how this model works. Suppose a user tries to open a web page. The very first thing he does is sending a request to the server that is located several thousand miles away. Here the server's hard disk or hardware is the last layer (layer 1) which is termed as "physical". So user's request first knocks the "application" layer (7) which is the nearest and then it proceeds. Every process in each layer involves a complicated "bits and bytes" functioning. A Computer only understands 0 and 1. But the user does not love to see a video in 0 and 1. Let us break the process in more detail.

In the application layer (7) user interacts with the device that could be a personal computer or smart phone or anything you might guess. So the application layer basically handles the user's interaction. The name of datagram is "data". The user requests for the data and ultimately retrieves the data. What happens when the user sends requests from the layer 7? It enters into the next layer (6) presentation. The process of encapsulation starts. Data is formatted and encrypted. Next the layer 5 or session enters into the scene. This layer manages end to end communication. Suppose you type a password and log into your social media account. This layer maintains the end to end (user to server) communication so that you can remain logged into your page. Till this layer the name of datagram is "data".

To assist you to maintain your session the next three layers work very hard. They are transport (layer 4), network (layer 3), data link (layer 2) respectively. The name of the datagram of transport layer is "segment". Why this is called "segment"? It is called segment because it breaks your request into several fractions. First it adds source and destination port numbers. Next it tries to make it reliable adding sequence numbers. So in a nutshell it provides flow control, sequencing and reliability.

What happens next?

Your request enters into the layer 3 that is called network. The name of datagram is "packet". It adds source and destination IP addresses. It also looks after so that your request finds the best path to reach the destination.

Now your data request almost reaches the final stage. It enters into the layer 2 that is data link. It is nearing the end point that is server's hardware. So this layer adds source and destination Media Access Control (MAC) addresses. Next it goes through Frame Check System (FCS) processes. It checks frame by frame whether the source requests reach the right destination. That is why the datagram is known as "frame".

Now it has entered into the final destination that is layer 1 or physical. There are only bits over the physical medium. The name of the datagram is "bits and bytes".

Now we can imagine a small office with one router, two switches and few desktops, laptops, printers and servers. The router is connected to the switches and the switches are connected to the devices

like desktops, laptops, printers and servers. Here desktops, laptops, printers and servers belong to the layer 1 that is physical. The switches belong to the layer 2 that is data link and the router fits in the layer 3 that is network.

Routers are layer 3 devices and perform few definite tasks. They are packet switching, packet filtering, and path selecting and finally communicating. The task of packet switching involves the process of getting a packet to the next device. Here the next device is the switches. Packet filtering suggests in its name what it actually does. It either permits or blocks packets depending on certain criteria. Path selecting is determining the best path through the network to the destination. Communication is another important part of this layer. Routers communicate with other networks like Internet.

Between routers, layer 3 devices, and the end application physical, layer 1 devices there are switches which are layer 2 devices. In some cases switches perform the task of layer 3 devices. Switches basically deal with frame filtering and forwarding. It also maintains the connection between layer 3 and layer 1.

Chapter Seven - Regular Expressions

Searching and replacing with regular expression is equally easy and very simple in nature. To do that we will tweak our old code a little bit. We use ‘re’ module and it does the simple jobs. Regular Expression is itself a big topic. We try to understand the basic things so that we can use it in our future project.

7.1 – Using ‘re’ Module

If you want to use ‘re’ module the first step is importation. We need to import the module first and write it on the top of the code. Consider this code where we have a text file called ‘file.txt’ and it is stored in our ‘primary’ folder.

```
#!/usr/bin/python3
import re
def main():
    ReplaceWord()
    DEmarcationLine()
    MatchAndReplaceWord()

def ReplaceWord():
    try:
        files = open("../primary/file.txt")
        for line in files:
            # you can search any word
            print(re.sub('lenor|more', "######", line), end=' ')
    except FileNotFoundError as e:
        print("File was not found:", e)

def MatchAndReplaceWord():
    try:
        files = open("../primary/file.txt")
        for line in files:
            # you can search any pattern that can match and then replace with this word
            match = re.search('(len|neverm)ore', line)
            if match:
                print(line.replace(match.group(), "######"), end=' ')
    except FileNotFoundError as e:
```

```
        print("File was not found:", e)

def DEmarcationLine():
    print("**************")

if __name__ == "__main__":
    main()
```

Before we have the output, let us see what is written inside the file. The 'file.txt' in 'primary' folder has these lines:

```
first line lenore
it is nine, second line and dine
third line and nevermore over
and fourth
fifth pine line lenore
and the tremor
here is more line
and a new line
i love you
where you are staying now?
i don't know
```

As you see, these are not very meaningful sentences. Our primary concern is very simple. We write down some nonsense lines and later try to work upon it with the use of 're' module. Now we run the code and here is the output:

```
first line ######e
 it is nine, second line and dine
 third line and never##### over
 and fourth
 fifth pine line ######e
 and the tremor
 here is ###### line
 and a new line
 i love you
 where you are staying now?
 i don't know **************
first line ######
 third line and ###### over
 fifth pine line ######
```

All the words “lenore” and “nevermore” have been replaced by five hash tags - “#####”. We use two methods of “re” module that we import and write on the top of the code. These methods are “re.sub()” and “line.replace()”. We have supplied the old string and the new word. We have given five hash tags but you could have given any other word of course.

7.2 - Reusing With Regular Expressions

You have already seen how we can search and replace words in a file with the help of regular expression. Now we will try to reuse the code so that we can use them again and again. Besides, we will also try to write them in a more human readable way. Let us first write the steps. What we want to achieve is very important before achieving something. Let us have a clear idea first and the best way is writing them down.

1) We need to open a file and put into the try block to avoid getting any nasty error message. Beginners may find this ‘try block’ quite intimidating. I have not explained it before and suddenly start using it. I have done it intentionally. It is explained in the next chapter ‘Exceptions, Catching Errors’. But before that I want you to write them and get habituated to a concept that looks complex. Once you learn this ‘try block’ please revisit this code again. You will find it extremely easy! Moreover, as you progress, you will find that using ‘try block’ is always a good habit.

2) Get the pattern of the words that we want to search and using flags we can ignore case.

3) Use that ‘re’ module search method to see if that pattern matches with our line.

4) Now if it matches, then replace it with new words.

Consider this code below and read the comments. In comments I briefly explain what I am going to do.

```
1  #!/usr/bin/python3
2  import re
3
4  def main():
5      CompilerAndReplaceWord()
6
7  def CompilerAndReplaceWord():
8      try:
9          files = open("../primary/file.txt")
10         # you can search any pattern that can match ignoring the upper or lower case
11         pattern = re.compile('(len|neverm)ore', re.IGNORECASE)
12         for line in files:
13             # re module search that pattern in a line
14             if re.search(pattern, line):
15                 # we found that patetrn and now it is time to replace them with a new s\
```

```
tring
                print(pattern.sub("######", line), end=' ')
    except FileNotFoundError as e:
        print("File was not found:", e)

if __name__ == "__main__":
    main()
```

And in the output it replaces all the words “lenore” and “nevermore” with six hash tags. To do that it also checks the upper and lower case and finally replace them all.

```
first line ######
 third line and ###### over
 fifth pine line ######
 i don't know ######
```

7.3 - Searching With Regular Expressions

Regular expressions are very powerful method of matching patterns. Regular expression is a small language in itself and it can be very simple and very complex.

It is implemented in Python with ‘re’ module.

Consider this code:

```
#!/usr/bin/python3
import re
def main():
    FindWord()
    DEmarcationLine()
    MatchWord()
def FindWord():
    try:
        files = open("../primary/file.txt")
        for line in files:
            # you can search any word
            if re.search('lenor|more', line):
                print(line, end=' ')
    except FileNotFoundError as e:
        print("Fiel was not found:", e)
def MatchWord():
    try:
```

```
        files = open("../primary/file.txt")
        for line in files:
            # you can search any pattern that can match this word
            match = re.search('(len|neverm)ore', line)
            if match:
                print(match.group())
    except FileNotFoundError as e:
        print("Fiel was not found:", e)
def DEmarcationLine():
    print("**************")
if __name__ == "__main__":
    main()
```

Here we search a file called 'file.txt' that has words like 'lenor or more' and that also matches some words that ends with 'ore'. We have defined two functions to search that and we used 're' module. Let us first see what is the content inside 'file.txt'. There are some rubbish words and lines just to test our search.

```
first line lenore
it is nine, second line and dine
third line and nevermore over
and fourth
fifth pine line lenore
and the tremor
here is more line
and a new line
i love you
where you are staying now?
i don't know
```

After running our code we have found this search result.

```
first line lenore
 third line and nevermore over
 fifth pine line lenore
 **************
lenore
nevermore
lenore
```

It is a very simple regular expression example. It is beyond our scope to teach regular expression here but we can at least have some idea here. I strongly recommend you to move further. Search about

Regular Expression in Internet. You will find lot of tutorials. Learning and understanding Regular Expression is very important. Whether you become a web developer, ethical hacker, or a python programmer; Regular Expression will come to your help.

PART TWO: PYTHON AND HACKING

Python programming language helps hackers in many ways. Especially in the networking and reconnaissance scanning field it has so many things to offer.

Python can do many things in other fields too. In the socket and networking field you may have your own script or you may build a NMAP scanner to gather information. Besides, in system monitoring it has huge importance. In the advanced level of ethical hacking it can cast a magical spell. You can write your own python program for any type of security purpose.

Remember, any program written in Python or any language does issue some instructions. And they are more or less same. They are:

INPUT: Get data from keyboard or any file or any source.

OUTPUT: Display data on screen or send it to any file, device or any other source.

MATHEMATICS: Do some basic mathematical operations like add, subtract, multiply or divide. It can be complex also. It depends on your application.

CONDITIONAL EXECUTION: Check that the conditions are properly met. Like 'if that is true then do something else do some other thing.'

REPETITION: Perform some action repeatedly.

In the next chapters we'll discuss few important aspects of Python. It's important because in our future endeavor we'll deal with python modules.

If you have no basic knowledge about python, please do that first. Here you're going to know reconnaissance scanning and information gathering issues especially as we've focused on few intermediate and advanced matters.

Object in Python

Python is an object oriented language. We will discuss it later in detail. Let us think there is a class or blueprint and from this class or blueprint we can get many types of objects. Suppose Human class. It is a very complex class indeed! It has many kind of properties, many kind of actions are performed by this class. When we create an object or instance of this class, this object or instance can carry forward every single trait of this class. Remember there has always been a good human being and a bad human being.

Let us assume, a human class has two types of humans – one is good and the other is bad. In reality, it is not so simple. But to begin with our learning we start with a less complex scenario. Consider the code below:

```
#!/usr/bin/python3
class Human:
    def __init__(self, kind = "Good"):
        self.kind = kind
    def whatKind(self):
        return self.kind
def main():
    GoodHuman = Human()
    print(GoodHuman.whatKind())
    BadHuman = Human("Bad")
    print(BadHuman.whatKind())
if __name__ == "__main__":
    main()
```

And here is the output:

```
Good
Bad
```

In the above code the object is the instance of a class and encapsulates every property and methods of the class or blueprint. In the above class, we assume a sort of blueprint where every human being is good. So in the initialization method, we write this code:

```
class Human:
    def __init__(self, kind = "Good"):
        self.kind = kind
    def whatKind(self):
        return self.kind
```

Here “self” means a reference to the object. And the next parameter defines the kind of human objects we want to create.

What does this line mean?

```
def whatKind(self):
        return self.kind
```

It returns the value of what kind of human object we want to create. The next steps are quite self explanatory as it goes:

```
def main():
    GoodHuman = Human()
    print(GoodHuman.whatKind())
    BadHuman = Human("Bad")
    print(BadHuman.whatKind())
if __name__ == "__main__":
    main()
```

When we create our first object “GoodHuman”, we need not pass any value as “good” is the default value that has already been passed implicitly through the initialization process. But when we want to create “BadHuman” we need to pass the value explicitly and it returns that value.

Conditionals

In python there are two types of conditionals. They are: conditional executions and conditionals values or conditional expressions. In Conditional executions we execute or check the condition of the statement. We know between two values there could be three types of conditions. It is either less than or greater than or it is equal. Write this code:

```
def conditionals_exec():
    a, b = 1, 3
    if a < b:
        print("a is less than b")
    elif a > b:
        print("a is greater than b")
    else:
        print("a is equal to b")
conditionals_exec()
```

The output is:

```
# a is less than b
```

The output is obvious. Now you can change the value and test the code. Now try to rewrite the above statement in a different way. We can say x is either less than y or greater than y. Else it is obvious that they are equal.

```
def conditional_values():
    a, b = 1, 2
    statements = "less than " if a < b else " not less than."
    print(statements)
conditional_values()
```

These functions can be written more conveniently and neatly with the main() functions now:

```
def main():
    print("This is main function.")
    conditionals_exec()
    conditional_values()

def conditionals_exec():
    a, b = 1, 3
    if a < b:
        print("a is less than b")
    elif a > b:
        print("a is greater than b")
    else:
        print("a is equal to b")

def conditional_values():
    a, b = 1, 2
    statements = "less than " if a < b else " not less than."
    print(statements)

if __name__ == "__main__": main()
```

If we run this program now, the output will be:

```
# This is main function.
# less than
# a is less than b
```

Now we can change the place of conditional_values(), and conditionals_exec() and the output will change accordingly

```
# This is main function.
# a is less than b
# less than
```

Loops

'While loop' is the simplest form of loop in Python. But you need to understand it properly. Otherwise it can end up in eating up your memory running the infinity loop. Usually most of the jobs are done by 'for loop'. But in some special cases, you need to use 'while loop'. A basic understanding is important.

While Loops

In plain English we often say - 'While it is true it keeps on running. While it is not true it stops.' Logically same thing happens here. While a statement is true, the process is going on. You need a mechanism to stop that process. That is important. Otherwise that statement will eat up your memory.

Consider this code:

```
b = 1
while b < 50:
    print(b)
    b = b + 1
```

What does it mean? It means, the statement 'b < 50' is true until the suite or block of code is true inside it. Inside the block we wrote 'b = b + 1' and before the beginning of the while loop we defined the value of b as 1.

So in each step b progresses by adding 1 in its value and finishes at 49. In the output you will get 1 to 49.

Let us move further. Consider this code:

```
#!/usr/bin/python3
# simple fibonacci series
# sum of two numbers define the next set
a, b = 0, 1
while b < 50:
    print(b, end=' ')
    a, b = b, a + b
```

The output is quite obvious:

```
1 1 2 3 5 8 13 21 34
```

For the beginners, let us write this code in a more human readable way and it will give different output altogether:

```
#!/usr/bin/python3
a, b = 0, 1
while b < 30:
    print(b, end=' ')
    a = b
    b = a + b
```

Let us explain the steps one by one to understand it properly.

The loop starts with 1. In the first step the value of 'a' is 1. In the next step value of 'b' is 2. Now the value of 'a' is 2 so the value of 'b' is 4. Now the value of 'a' is 4 so the value of 'b' is 8(4+4). Now the value of 'a' is 8 so the value of 'b' is (8 + 8) = 16. Now the value of 'a' is 16. What will be the value of b? It will be 16 + 16 = 32. But 32 are greater than 30. So it will come out from the code suite of while loop.

The output of the above code will be:

```
1 2 4 8 16
```

Let us write the whole bunch of code in a new format:

```
#!/usr/bin/python3
# simple fibonacci series
# sum of two numbers define the next set
a, b = 0, 1
while b < 30:
    print("a = ", a, "=" , "b = ", b, "," , end=' ')
    a, b = b, a + b
print("***********")
a, b = 0, 1
while b < 30:
    print("a = ", a, "=" , "b = ", b, "," , end=' ')
    a = b
    b = a + b
```

And the output will be:

```
a =  0 and b =  1 , a =  1 and b =  1 , a =  1 and b =  2 , a =  2 and b =  3 , \
a =  3 and b =  5 , a =  5 and b =  8 , a =  8 and b =  13 , a =  13 and b =  21\
 ,
********** Lines of separation **********
a =  0 and b =  1 , a =  1 and b =  2 , a =  2 and b =  4 , a =  4 and b =  8 , \
a =  8 and b =  16 ,
```

Now hopefully, this explains how the while loops work.

For Loops

The most common loop used in Python is for loop. In fact, essentially almost all kind of looping job can be done through the 'for' loop.

There is a reason of course. With the help of for loop we can iterate through python objects and we can iterate through most of the python objects. Let us see one example:

```
#!/usr/bin/python3
songs = open('file.txt')
for lines in songs.read():
    print(lines, end='')
```

And the output of the song goes like this:

```
Yo, girl you touched me hard
your loneliness has made me weep
I am a sooo stupid nerd
I thought about the words, I could not keep
So I weep
A stupid nerd
```

We have a song written over in a file called 'file.txt' and we just iterate through this file. We could have iterated through line by line as they are indexed. Consider this code where we just used 'enumerate()' function and index value :

```
songs = open('file.txt')
for index, lines in enumerate(songs.readlines()):
    print(index, lines, end='')
```

And the output is like this:

```
0 Yo, girl you touched me hard
1 your loneliness has made me weep
2 I am a sooo stupid nerd
3 I thought about the words, I could not keep
4 So I weep
5 A stupid nerd
```

Now what does this function 'enumerate()' mean? Dictionary says: enumeration is a kind of numbering which is a numbered list. Let us consider this line of code:

```
strings = "This is a string."
# now we are going to find how many 's' is inside this string
for index, s in enumerate(strings):
    if s == 's':
        print("Hi I am 's' and I am located at position {}".format(index))
```

And we have an output:

```
Hi I am 's' and I am located at position 3
Hi I am 's' and I am located at position 6
Hi I am 's' and I am located at position 10
```

This is extremely useful. You can search any character inside any string. In Python, functions or subroutines are extremely important for reusability of codes. We can call a function for several times and pass many arguments or parameters to get different effects. Now we are going to pass one parameter inside the loops() function. Consider this code below:

```
#!/usr/bin/python3
def main():
    loops(0)
    loops()
    loops(3)

def loops(a = 4):
    for i in range(a, 6):
        print(i, " ")
    print("**************")
if __name__ == "__main__":
    main()
```

What does this code mean? In loops() function we have passed one parameter a and assigned a value 4. It is the default value. So that in future if we forget to pass any argument the code will not break. We have called that function three times inside main() function. But with three different values and one of them is NULL that is we have not passed any argument. The output changes with the new code:

```
0
1
2
3
4
5
**********
4
5
**********
3
4
5
**********
```

Now it is obvious that you can play around with this code. You can pass two arguments inside loops() function and control the range() function to get different values.

Regular Expressions

Searching and replacing with regular expression is equally easy and very simple in nature. To do that we will tweak our old code a little bit. We use ‘re’ module and it does the simple jobs. Regular Expression is itself a big topic. We try to understand the basic things so that we can use it in our future project.

Using ‘re’ Module

If you want to use ‘re’ module the first step is importation. We need to import the module first and write it on the top of the code. Consider this code where we have a text file called ‘file.txt’ and it is stored in our ‘primary’ folder.

```
#!/usr/bin/python3
import re
def main():
    ReplaceWord()
    DEmarcationLine()
    MatchAndReplaceWord()

def ReplaceWord():
    try:
        files = open("../primary/file.txt")
        for line in files:
            # you can search any word
            print(re.sub('lenor|more', "#####", line), end=' ')
    except FileNotFoundError as e:
        print("File was not found:", e)

def MatchAndReplaceWord():
    try:
        files = open("../primary/file.txt")
        for line in files:
            # you can search any pattern that can match and then replace with th\
is word
            match = re.search('(len|neverm)ore', line)
            if match:
                print(line.replace(match.group(), "#####"), end=' ')
```

```
    except FileNotFoundError as e:
        print("File was not found:", e)

def DEmarcationLine():
    print("**************")

if __name__ == "__main__":
    main()
```

Before we have the output, let us see what is written inside the file. The 'file.txt' in 'primary' folder has these lines:

```
first line lenore
it is nine, second line and dine
third line and nevermore over
and fourth
fifth pine line lenore
and the tremor
here is more line
and a new line
i love you
where you are staying now?
i don't know
```

As you see, these are not very meaningful sentences. Our primary concern is very simple. We write down some nonsense lines and later try to work upon it with the use of 're' module. Now we run the code and here is the output:

```
first line #####e
 it is nine, second line and dine
 third line and never##### over
 and fourth
 fifth pine line #####e
 and the tremor
 here is ##### line
 and a new line
 i love you
 where you are staying now?
 i don't know **************
first line #####
 third line and ##### over
 fifth pine line #####
```

All the words “lenore” and “nevermore” have been replaced by five hash tags - “#####”. We use two methods of “re” module that we import and write on the top of the code. These methods are “re.sub()” and “line.replace()”. We have supplied the old string and the new word. We have given five hash tags but you could have given any other word of course.

Reusing With Regular Expressions

You have already seen how we can search and replace words in a file with the help of regular expression. Now we will try to reuse the code so that we can use them again and again. Besides, we will also try to write them in a more human readable way.

Let us first write the steps. What we want to achieve is very important before achieving something. Let us have a clear idea first and the best way is writing them down.

1) We need to open a file and put into the try block to avoid getting any nasty error message. Beginners may find this ‘try block’ quite intimidating. I have not explained it before and suddenly start using it. I have done it intentionally. It is explained in the next chapter ‘Exceptions, Catching Errors’. But before that I want you to write them and get habituated to a concept that looks complex. Once you learn this ‘try block’ please revisit this code again. You will find it extremely easy! Moreover, as you progress, you will find that using ‘try block’ is always a good habit.

2) Get the pattern of the words that we want to search and using flags we can ignore case.

3) Use that ‘re’ module search method to see if that pattern matches with our line.

4) Now if it matches, then replace it with new words. Consider this code below and read the comments. In comments I briefly explain what I am going to do.

```
1  #!/usr/bin/python3
2  import re
3
4  def main():
5      CompilerAndReplaceWord()
6
7  def CompilerAndReplaceWord():
8      try:
9          files = open("../primary/file.txt")
10         # you can search any pattern that can match ignoring the upper or lower \
11 case
12         pattern = re.compile('(len|neverm)ore', re.IGNORECASE)
13         for line in files:
14             # re module search that pattern in a line
15             if re.search(pattern, line):
16                 # we found that patetrn and now it is time to replace them with \
```

```
a new string
                print(pattern.sub("######", line), end=' ')
    except FileNotFoundError as e:
        print("File was not found:", e)

if __name__ == "__main__":
    main()
```

And in the output it replaces all the words “lenore” and “nevermore” with six hash tags. To do that it also checks the upper and lower case and finally replace them all.

```
first line ######
 third line and ###### over
 fifth pine line ######
 i don't know ######
```

Search With Regular Expressions

Regular expressions are very powerful method of matching patterns. Regular expression is a small language in itself and it can be very simple and very complex. It is implemented in Python with ‘re’ module.

Consider this code:

```
#!/usr/bin/python3
import re
def main():
    FindWord()
    DEmarcationLine()
    MatchWord()
def FindWord():
    try:
        files = open("../primary/file.txt")
        for line in files:
            # you can search any word
            if re.search('lenor|more', line):
                print(line, end=' ')
    except FileNotFoundError as e:
        print("Fiel was not found:", e)
def MatchWord():
    try:
```

```
        files = open("../primary/file.txt")
        for line in files:
            # you can search any pattern that can match this word
            match = re.search('(len|neverm)ore', line)
            if match:
                print(match.group())
    except FileNotFoundError as e:
        print("Fiel was not found:", e)
def DEmarcationLine():
    print("**************")
if __name__ == "__main__":
    main()
```

Here we search a file called 'file.txt' that has words like 'lenor or more' and that also matches some words that ends with 'ore'. We have defined two functions to search that and we use - 're' module. Let us first see the content inside 'file.txt'. There are some rubbish words and lines just to test our search.

```
first line lenore
it is nine, second line and dine
third line and nevermore over
and fourth
fifth pine line lenore
and the tremor
here is more line
and a new line
i love you
where you are staying now?
i don't know
```

After running our code we have found this search result.

```
first line lenore
 third line and nevermore over
 fifth pine line lenore
 **************
lenore
nevermore
lenore
```

It is a very simple regular expression example. It is beyond our scope to teach regular expression here but we can at least have some idea here. I strongly recommend you to move further. Search about

Regular Expression in Internet. You will find lot of tutorials. Learning and understanding Regular Expression is very important. Whether you become a web developer, ethical hacker, or a python programmer; Regular Expression will come to your help.

Exceptions, Errors

I hope you have already written lot of codes. If you had really done that, you would have encountered one or two errors. There are two distinguishable kinds of errors. The first is 'SyntaxError'. It means, you have error in your syntax. Consider this code:

```
>>> for i in range(10) print(i)
SyntaxError: invalid syntax
```

As you see, I forgot to use ':' in for loop. It is a syntax error. Another error is 'Exceptions'. It means you write a code perfectly. There are no syntactical errors. But you forget to define a variable. Let us consider these lines of code:

```
>>> 10 * x
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    10 * x
NameError: name 'x' is not defined
>>> 10 / 0
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    10 / 0
ZeroDivisionError: division by zero
>>> '2' + 2
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    '2' + 2
TypeError: Can't convert 'int' object to str implicitly
>>> inputs = input("Please enter a number.")
Please enter a number.
>>> inputs + 2
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    inputs + 2
TypeError: Can't convert 'int' object to str implicitly
>>> inputs = input("Please enter a number.")
Please enter a number.12
>>> inputs - 10
```

```
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    inputs - 10
TypeError: unsupported operand type(s) for -: 'str' and 'int'
>>> int(inputs) - 10
2
>>>
```

As you see there are lot of different kind of errors. And in the last line we have come out from the error and get a perfect output. In the last error we get a 'TypeError'. We tried to subtract an integer from a string object. In the last step we converted that string input integer and the subtraction took place smoothly.

It is always good to catch those errors and get a nice output. The usage of 'try block' phrase has been taken place before. Now the time comes when we learn how we use those blocks to catch errors. Write down the code below in your text editor and save it as 'CatchError.py'.

```
#!/usr/bin/python3
def main():
    FileRead()
    DemarcationLine()
    LineStrip()
    DemarcationLine()
    CheckFileExtension()
def ReadFile(filename):
    files = open(filename)
    lines = files.readlines()
    for index, line in enumerate(lines):
        print(index, "=", line)
def StripFile(filename):
    files = open(filename)
    for lines in files:print(lines.strip())
def RaisingError(filename):
    if filename.endswith(".txt"):
        lines = open(filename)
        for line in lines:print(line.strip())
    else:
        raise ValueError("File must end with .txt")
def FileRead():
    try:
        ReadFile("../primary/files.txt") # path is okay, it reads file
    except IOError as e:
        print("Could not open file:", e)
```

```
def LineStrip():
    try:
        StripFile("primary/files.txt")
    except IOError as e:
        print("Could not open file:", e) # it will give error
def CheckFileExtension():
    try:
        RaisingError("../primary/file.rtf")
    except IOError as e:
        print("Could not open file:", e)
    except ValueError as e:
        print("Bad Filename:", e)
def DemarcationLine():
    print("******************")
if __name__ == "__main__":
    main()
```

Run this file and you get this output:

```
Could not open file: [Errno 2] No such file or directory: '../primary/files.txt'
******************
Could not open file: [Errno 2] No such file or directory: 'primary/files.txt'
******************
Bad Filename: File must end with .txt
```

As an exercise, try to write this code with ‘Try’ and ‘Except’ and catch if there is any error.

```
#!/usr/bin/python3
def main():
    GetARangeOfNumber()
def GetARangeOfNumber():
    for index in IteratingStepByStep(1,123, 7):
        print(index, end=' ')
def IteratingStepByStep(start, stop, step):
    number = start
    while number <= stop:
        yield number
        number += step
if __name__ == "__main__":
    main()
```

Functions

Let us first define the function and try to know why function is being used in Python. Consider this code:

```
#!/usr/bin/python3
def main():
    print("This is main function.")
if __name__ == "__main__":
    main()
```

And the output is:

```
This is main function.
```

What does that mean? First of all, let us understand what function does mean. A function is used in any programming language to reuse code. Programmers are lazy and so they don't want to write again and again. And it is not a good idea to write same thing again and again. So the concept of reusability comes in and we use function to do that.

You may consider a very simple example. Suppose we want to use a demarcation line again and again. Will you write like this again and again?

```
print(“**********”)
```

Or you will write a function and call it when it is necessary? Like this:

```
def DemarcationLine():
        print(“*********”)
DemarcationLine():
DemarcationLine():
DemarcationLine():
```

Each time you call the function 'DemarcationLine()' it will print a demarcation line. Now let us come to the first question. It is always a good practice to write functions inside main() function and you can call them any time. The flow control not necessarily follows downward. You can test it:

```
def AnotherFunction():
    print("I am another function.")
def TestFunction():
    print("I am going to call another function.")
    AnotherFunction()
TestFunction()
```

It will print without any problem and give you this output:

```
I am going to call another function.
I am anotheer function.
```

Now we will write the above code differently.

```
def TestFunction():
    print("I am going to call another function.")
    AnotherFunction()

TestFunction()

def AnotherFunction():
    print("I am another function.")
```

A little bit change in the position. We have not defined AnotherFunction() before TestFunction() and for that reason, it will give an error output:

```
I am going to call another function.
Traceback (most recent call last):
  File "/home/hagudu/PycharmProjects/FirstPythonProject/functions/defining_funct\
ions.py", line 17, in <module>
    TestFunction()
  File "/home/hagudu/PycharmProjects/FirstPythonProject/functions/defining_funct\
ions.py", line 15, in TestFunction
    AnotherFunction()
NameError: name 'AnotherFunction' is not defined
```

So each time you call a function inside another function, you need to define it before. But this problem can be solved if you define main() function first. Now consider this code:

```
#!/usr/bin/python3
def main():
    TestFunction()
def TestFunction():
    print("I am going to call another function.")
    AnotherFunction()
def AnotherFunction():
    print("I am another function.")
if __name__ == "__main__":
    main()
```

And here is the output:

```
I am going to call another function.
I am another function.
```

Now see, we did not bother about the position any more because all the functions are under main() function. Much more flexibilities are now being added when you are using main() function like this. Another great advantage of using function is passing parameters or arguments through it.

```
#!/usr/bin/python3
def main():
    PassingParameters(1,2,3)
def PassingParameters(argument1, argument2, argument3):
    print("Here is our arguments:", argument1, argument2, argument3)
if __name__ == "__main__":
    main()
```

And the output is:

```
Here is our arguments: 1 2 3
```

We have passed three parameters or arguments and get the output as expected. But what happens if we forget to pass any argument? We don't want to get any nasty error message. We can manage that by two ways:

```
#!/usr/bin/python3
def main():
    PassingParameters(1)
def PassingParameters(argument1, argument2 = 4, argument3 = 6):
    print("Here is our arguments:", argument1, argument2, argument3)
if __name__ == "__main__":
    main()
```

And the output:

```
Here is our arguments: 1 4 6
```

It is called passing default values. We have passed two default values and when we actually call the function, it takes that default value. Now we can override these default values any time. Consider this one:

```
#!/usr/bin/python3
def main():
    PassingParameters(1, 10, 14)
def PassingParameters(argument1, argument2 = 4, argument3 = 6):
    print("Here is our arguments:", argument1, argument2, argument3)
if __name__ == "__main__":
    main()
```

And the output:

```
Here is our arguments: 1 10 14
```

We have overwritten the default values by passing new values and the output has changed accordingly. We can write this code this way also:

```
#!/usr/bin/python3
def main():
    PassingParameters(1)
def PassingParameters(argument1, argument2 = None, argument3 = 6):
    if argument2 == None:
        print("Here is our arguments:", argument1, argument2, argument3)
    else:
        print("Here is our arguments:", argument1, argument2, argument3)
if __name__ == "__main__":
    main()
```

And the output:

```
Here is our arguments: 1 None 6
```

What happens if we pass a new value for argument2? Consider this code:

```
#!/usr/bin/python3
def main():
    PassingParameters(1, 12)
def PassingParameters(argument1, argument2 = None, argument3 = 6):
    if argument2 == None:
        print("Here is our arguments:", argument1, argument2, argument3)
    else:
        print("Here is our arguments:", argument1, argument2, argument3)
if __name__ == "__main__":
    main()
```

And the output:

```
Here is our arguments: 1 12 6
```

In the next section we will see how lists of arguments work in a function.

Return Values

In Python a function can return any value. It can return any type of data – string, integer, object; anything. Let us return an object. Consider this code:

```
#!/usr/bin/python3
def main():
    for index in ReturnValues():
        print(index, end=" ")
def ReturnValues():
    #return "Returning string."
    #return 56
    return range(10)
if __name__ == "__main__":
    main()
```

And the output

```
0 1 2 3 4 5 6 7 8 9
```

We have returned range() object and got the value in our main() function.

Generate Functions

In Python we can generate functions. Let us explain it by step by step. Consider this code first:

```
#!/usr/bin/python3
def main():
    RangeFunctions()
def RangeFunctions():
    for i in range(10):
        print(i, end=' ')
if __name__ == "__main__":
    main()
```

And the output is quite obvious:

```
0 1 2 3 4 5 6 7 8 9
```

You have probably found that the function RangeFunctions() has a limitation. It stops at 9 as the range is mentioned as 10. What can I do to include this number? Let us write RangeFunctions() this way:

```
#!/usr/bin/python3
def main():
    for index in RangeFunctions(0, 10, 1):
        print(index, end=' ')

def RangeFunctions(start, stop, step):
    i = start
    while i <= stop:
        yield i
        i += step
if __name__ == "__main__":
    main()
```

And here is the output:

```
0 1 2 3 4 5 6 7 8 9 10
```

Here we have used the ‘yield’ keyword. It is done because; we have imagined that the code will progress step by step like we play a tape. After yielding one step it will stop and start from there and again starts and goes one step. You can just start from any point; stop at any point and progress by any step. If we write like this:

```
for index in RangeFunctions(15, 1025, 102):
        print(index, end=' ')
```

The output will be: 15 117 219 321 423 525 627 729 831 933. As you have seen that we can set the value any argument as default. So we can write this function like this:

```
def AnotherRangeFunctions(start = 0, stop, step = 1):
    i = start
    while i <= stop:
        yield i
        i += step
```

And we may try to get the output by

```
for index in AnotherRangeFunctions(25):
    print(index, end=' ')
```

But it gives us an error message:

```
File "/home/hagudu/PycharmProjects/FirstPythonProject/functions/generate-functio\
ns.py", line 18
    def AnotherRangeFunctions(start = 0, stop, step = 1):
SyntaxError: non-default argument follows default argument
```

Python does not support this. Can we solve this problem? So that we can pass any number of arguments and control it without having any error message? Consider this code:

```
def AnotherRangeFunctions(*args):
    numberOfArguments = len(args)

    if numberOfArguments < 1: raise TypeError('At least one argument is required\
.')
    elif numberOfArguments == 1:
        stop = args[0]
        start = 0
        step = 1
    elif numberOfArguments == 2:
        # start and stop will be tuple
        (start, stop) = args
        step = 1
    elif numberOfArguments == 3:
        # all start and stop and step will be tuple
        (start, stop, step) = args

    i = start
    while i <= stop:
        yield i
        i += step
```

Write down every line and take note side by side. Add comments where you feel that an explanation is necessary.

Lists of Arguments

In Python sometimes you need arbitrary number of arguments and you have to name them. Let us write this code:

```
#!/usr/bin/python3
def main():
    PassingListsOfArguments(1, 2, 3, 5, 7, 45, 98, 56, 4356, 90876543)
    PassingAnotherListsOfArguments(1, 2, 3, 5, 7, 45, 98, 76, 987654, 3245, 2345\
, 98760)

def PassingListsOfArguments(arg1, arg2, arg3, arg4, *args):
    print(arg1, arg2, arg3, arg4, args)

def PassingAnotherListsOfArguments(param1, param2, *params):
    print(param1, param2)
```

```
12      for index in params:
13          if index == 76:
14              x = 10
15              y = index + x
16              print("We are going to add 10 with", index, "and the new value is:",\
17   y)
18              continue
19          print(index, end=' ')
20
21  if __name__ == "__main__":
22      main()
```

And the output goes like that:

```
1  1 2 3 5 (7, 45, 98, 56, 4356, 90876543)
2  1 2 3 5 7 45 98 We are going to add 10 with 76 and the new value is: 86
3  987654 3245 2345 98760
```

In our code *args or *params mean lists of arguments. You can pass any number of arguments through them. Consider this line:

```
1  def PassingListsOfArguments(arg1, arg2, arg3, arg4, *args):
```

It means you need to pass four arguments first. That is compulsory. After that number of arguments may vary. But the arbitrary number of arguments comes out as 'tuple'. See the output of this function:

1 2 3 5 (7, 45, 98, 56, 4356, 90876543)

The later part is obviously a tuple and you can iterate through it.

9.4 - Named Arguments

Sometimes it is important to use named arguments in Python. And we get those named arguments in a dictionary format. Consider this code:

!/usr/bin/python3

def main(): NamedArguments(name = ‘Sanjib’, address = ‘Pluto’, hobby = “Gardening”) def NamedArguments(**kwargs): for key in kwargs: print(key, “=”, kwargs[key]) if **name** == “**main**”: main()

And the output

hobby = Gardening name = Sanjib address = Pluto

As it is a dictionary output, it is not ordered. You can sort it alphabetically. Let us consider a fairly long code where we can use every kind of passing arguments.

!/usr/bin/python3

def main(): NamedArguments(name = ‘Sanjib’, address = ‘Pluto’, hobby = “Gardening”) DemarcationLine() AnotherNamedArguments(‘Hi’, 1235, 1,2,3, one = 1, two = 2, three = 3)

def NamedArguments(**kwargs): for key in kwargs: print(key, “=”, kwargs[key])

def AnotherNamedArguments(arg1, arg2, *args, **kwargs): print(arg1, arg2) for index in args: print(index, end=’ ‘) DemarcationLine() for keys in kwargs: print(keys, “=”, kwargs[keys])

def DemarcationLine(): print(“*********”)

if **name** == “**main**”: main()

Here is the output:

hobby = Gardening address = Pluto name = Sanjib **** Hi 1235 1 2 3 **** three = 3 two = 2 one = 1

Classes

If you are a complete beginner, you probably for the first time hear about "Object Oriented Programming and Class." Let us give a brief introduction to Object Oriented Programming (OOP).

Object Oriented Methodology

It is based on real world programming. An object is a representation of a real world entity. If there is an object, there must be a class or blueprint behind it. In that class the behavior of that object is designed or described in detail. These details consist of all the properties and actions that the object performs. There could be many types of objects coming from different classes and they might have relationships. It could be very complicated but you can always break those objects from one another and make some changes. The advantage of Object Orientation is that when you work on a part of a big, complicated project the other part remains unaffected. Our goal is simple. We want to join different objects to create big, complicated software. At the same time we want to make the relations of those objects as loose as possible.

A car object is built of many other objects like tyre, wheel, engine, accelerator etcetera. If you get a flat tyre does the engine stop? It is interrelated and depends on one another. But finally you can work on them individually without affecting other. That is Object Orientation.

Consider an object 'GoodHuman'. This object must be different from another object 'BadHuman'. Both come from 'Human' class. Now these two objects might have interrelationships and data interactions. Can you imagine how many kinds of properties and methods are there in 'Human' class? It could be very complex. Imagine a situation where a 'BadHuman' does something ugly. At the same time a 'GoodHuman' does some good thing. Whoever does whatever thing the life goes on and that is also Object Orientation.

Object Orientation is a type of methodology used for building software applications. An Object Oriented program consists of classes, objects and methods. The Object Oriented methodology in software development revolves around a single concept called the object. You can develop software by breaking the application into component objects. These objects interact with each other when the whole application is put together. An object is a combination of messages and data. The object receives and sends messages and those messages contain data or information.

You (an object) interact with your Television (another object) via messages sent through a remote controller (another object).

Consider another real world example of a football. A football has a boundary. It has a specific defined property like bouncing. You can direct or apply few specific actions by kicking it or throwing it.

An object has a state. It may display behavior. It has a unique ID.

The difference between an object and class is subtle but important. Whereas a class is an abstract concept, an object is a concrete entity. From a class objects with specific properties can be created or instantiated. That is why an object is often called an instance of a class.

One of the major features of Object Oriented Programming is 'Polymorphism'. Polymorphism is the capability of something to assume different forms. In Object Oriented Programming polymorphism is the property that a message can mean different things depending on the objects receiving it. The message 'Accelerate' means one thing if it sent to an object "OldCar". But it means different thing to if it is sent to the object "NewCar". It is a natural concept that can be applied to objects. It also means that similar objects often accept the same message but do different things.

Consider a web page. It is an object. There are billions of such objects around us. When you send a request to an object like web page you actually apply a verb "GET" to a noun "WebPage". Now every "WebPage" object does not behave the same way when the "GET" verb is applied. Someone open up a PDF file, someone simply shows some texts and pictures and someone may harm your computer. When you double-click a file. It may execute if it is an executable file. Or it may opens up in a text editor if it is a text file. The message is same. That is "Double Click". But the behaviour displayed by the file object depends on the object itself.

This is Polymorphism. You will learn it heart as you progress through this chapter. The advantage of Python classes is that it provides all the standard features of Object Oriented Programming. It has the class inheritance mechanism. That allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name. Objects can contain arbitrary amounts and kinds of data.

Finally remember, in Python everything is an object. It means there is an abstraction or encapsulation behind it. Your need to understand the abstraction first and then you create your own abstraction.

Classes and Objects

You can not understand theory unless you implement that concept into the real world. Let us see what we have learned.

1) Classes are what you create your own object.

2) A class is a blueprint for an object.

3) An object is an instance of a class.

Let us see how we can build a class and later create a few instances from it. Consider this code:

```
#!/usr/bin/python3
class Robot:
    def __init__(self):
        pass
    def WalkLikeARobot(self):
        print("walks like a robot.")
    def CareLikeARobot(self):
        print("takes care like a robot.")
robu1 = Robot()
print(type(robu1))
print(id(robu1))
robu2 = Robot()
print(type(robu2))
print(id(robu2))
del robu2
def main():
    robu = Robot()
    print(type(robu))
    print(id(robu))
if __name__ == "__main__":
    main()
```

In this code we have class definition of “Robot”. Here ‘class’ is the key word. Next to it is a “:” sign which means a class definition will follow a suite or block of codes. After we have defined the class “Robot”, we have three methods.

And they are:

```
def __init__(self):
        pass
    def WalkLikeARobot(self):
        print("walks like a robot.")
    def CareLikeARobot(self):
        print("takes care like a robot.")
```

The first one is the special method. When a class is instantiated, this method will be called first. “**init**” means initialization. The class is initialized. Other two methods follow it. Those methods are self explanatory. Methods are action verbs. When we create a robot object and we call those methods, we actually tell them to do something. In our class we defined what they will do.

In this code we created there robot objects. And finally we did not tell them to do anything. We have just seen how they are different from one another. We have tested their type and ID. Look, each object has different ID. So this is a major point. Each object or instance crated from a class, has their own individuality.

Now see the output:

```
<class '__main__.Robot'>
140445354614624
<class '__main__.Robot'>
140445354668160
<class '__main__.Robot'>
140445354668160
```

The next lines of code are a little bit longer but I strongly suggest that you write it on your own text editor and run the program to see that you get the same output.

```
#!/usr/bin/python3
class Robots:
    def __init__(self):
        pass
    def WalkLikeARobot(self, style):
        self.style = style
        return self.style
    def CareLikeARobot(self):
        print("takes care like a robot.")
class Humans:
    def __init__(self, nature = "good"):
        self.nature = nature
    def GoodHumanBeing(self):
        print("need not repeat, a good human being is always", self.nature)
    def BadHUmanBeing(self):
        self.nature = "need not repeat, bad human being is always bad."
        print(self.nature)
    def WalkLikeARobot(self, style):
        self.style = style
        return self.style
def main():
    robu = Robots()
    robu.CareLikeARobot()
    print(robu.WalkLikeARobot("walks like a robot"))
    GoodMan = Humans()
    print(GoodMan.nature)
    GoodMan.GoodHumanBeing()
    BadMan = Humans()
    BadMan.nature = "bad"
    print(BadMan.nature)
```

```
    BadMan.BadHUmanBeing()
    print(BadMan.WalkLikeARobot("he is human but walks like a robot"))
if __name__ == "__main__":
    main()
```

In the above snippet of code, we have two classes. One is “Robot”, that we wrote earlier. The other class is “Human”. In the “Human” class, we have defined this special method like this:

```
def __init__(self, nature = "good"):
        self.nature = nature
```

What does this mean? It means when we create a human instance of this class, we assume that the nature of human object will by default be good. Unfortunately it does not happen in the real world. Keeping that in our mind we also write this line “self.nature = nature”. It means self nature or the nature of the instance will be good if we do not explicitly mention that it “Bad” or something else.

In the following steps, when we create a bad human instance, we explicitly change the nature. Remember each method is the action part of that object. An object is a noun and it does something. In any software application it follows the same rule. An example of Polymorphism is also there. In both classes - “Robot” ans “Human”, we have defined a method:

```
def WalkLikeARobot(self, style):
        self.style = style
        return self.style
```

When we apply this same verb to the different Robot and Human object, it displays different behaviour. If you run this code, it gives us an output like this:

```
takes care like a robot.
walks like a robot
good
need not repeat, a good human being is always good
bad
need not repeat, bad human being is always bad.
he is human but walks like a robot
```

When a Robot instance walks like a robot, it displays: walks like a robot; but when an instance of Human walks like a robot, it displays: he is human but walks like a robot. This is nothing but a simple example of Polymorphism. When the same verb applies to two different objects, depending on the nature of the object it gives different output.

Actually we change this behavior by passing two different arguments. Suppose, instead of a single argument, we pass a dictionary of values. See how the power is magnified. Consider a simple code below:

```
print(type(BadMan.WalkLikeARobot(dict(one=1, two=2))))
st = BadMan.WalkLikeARobot(dict(one=1, two=2))
for keys in sorted(st):
    print(keys, st[keys])
ws = BadMan.WalkLikeARobot({'one':56, 'two':2})
for keys in sorted(ws):
    print(keys, ws[keys])
```

Here is the output:

```
<class 'dict'>
one 1
two 2
one 56
two 2
```

You can add more key, value pairs to this dictionary and run this code to see what happen.

Write a Game “Good VS Bad”

So far we have learned many things. I hope you have written the codes and tested them and it executed perfectly. Now the time has come to write a simple game in Python. It is a game called “Good Vs Bad”. The game is simple. But as a beginner you may find this code a bit longer. Write them down. Try to add more features.

If you are in Linux environment, save this file as ‘good-vs-bad.py’ and change the file executable by running this command:

```
Sudo chmod +x  good-vs-bad.py
```

And then run it on your terminal like this:

```
./ good-vs-bad.py
```

If you are in Windows, run the IDLE and save the file as ‘good-vs-bad.py’. Press F5 and play the game.

In the background the code shows and you may play the game on Python Shell. The code is like this:

```
#!/usr/bin/python3
class Robots:
    def __init__(self):
        pass
    def WalkLikeARobot(self, WalkingStyle):
        self.WalkingStyle = WalkingStyle
        return self.WalkingStyle
    def CareLikeARobot(self):
        print("takes care like a robot.")
class Humans:
    def __init__(self, nature = "good"):
        self.nature = nature
    def GoodHumanBeing(self):
        print("need not repeat, a good human being is always", self.nature)
    def BadHUmanBeing(self):
        self.nature = "need not repeat, bad human being is always bad."
        print(self.nature)
    def WalkLikeARobot(self, WalkingStyle):
        self.WalkingStyle = WalkingStyle
        return self.WalkingStyle
def main():
    robu = Robots()
    # robu.CareLikeARobot()
    # print(robu.WalkLikeARobot("A robot walks like a robot and nothing happens.\
"))
    GoodMan = Humans()
    # print(GoodMan.nature)
    # GoodMan.GoodHumanBeing()
    BadMan = Humans()
    # BadMan.nature = "bad"
    # print(BadMan.nature)
    # BadMan.BadHUmanBeing()
    # print(BadMan.WalkLikeARobot("he is human but walks like a robot"))
    # when a bad man wlaks like a robot many things happen
    WhenABadManWalksLikeARobot = BadMan.WalkLikeARobot(dict(change = 'he becomes\
 a monster inside',
                                act = 'he kills fellow people',
                                feel = 'he enjoys torturing animals',
                                care = 'he cares for none',
                                look = 'he looks a normal human being',
                                state = 'finally he destroys himself'))
    # there are lot of actions that take place
```

```
    print("What happens when a Bad Man walks like a Robot?")
    change = input("Tell us what kind of change may take place inside him?\n Cho\
ose between 'monster' and 'angel',"
                   "and type here...>>>>")
    WhenABadManWalksLikeARobot['change'] = change
    reward = 0
    if change == 'monster':
        print("You have won the first round:", change)
        reward = 1000
        print("You have won ", reward, "points.")
        print("What does he do? :", WhenABadManWalksLikeARobot['act'])
        change = input("Now tell us what the monster feels inside while killing \
people?\n Choose between 'great' and 'sad',"
                   "and type here...>>>>")
        WhenABadManWalksLikeARobot['change'] = change
        if change == 'great':
            print("You have won the second round:")
            reward = 10000
            print("You have won ", reward, "points.")
            print("What he feels inside? :", WhenABadManWalksLikeARobot['feel'])
            change = input("Tell us does the monster care for anyone?\n Choose b\
etween 'yes' and 'no',"
                   "and type here...>>>>")
            WhenABadManWalksLikeARobot['change'] = change
            if change == 'no':
                print("You have won the third round:")
                reward = 100000
                print("You have won ", reward, "points.")
                print("What he feels inside? :", WhenABadManWalksLikeARobot['car\
e'])
                change = input("Tell us does the monster look like a normal huma\
n being?\n Choose between 'yes' and 'no',"
                   "and type here...>>>>")
                WhenABadManWalksLikeARobot['change'] = change
                if change == 'yes':
                    print("You have won the fourth round:")
                    reward = 1000000
                    print("You have won ", reward, "points.")
                    print("What does he look like? :", WhenABadManWalksLikeARobo\
t['look'])
                    change = input("Tell us what happens to the monster finally?\
 Does he destroy himself\n Choose between 'yes' and 'no',"
```

```
                                    "and type here...>>>>")
                    WhenABadManWalksLikeARobot['change'] = change
                    if change == 'yes':
                        print("You have won the fifth round:")
                        reward = 100000000
                        print("You have won Jackpot.", reward, "points.")
                    else:
                        print("You have changed the course of game. It ends here\
. You have lost", reward - 100000, "points.")
                else:
                    print("You have changed the course of game. It ends here. Yo\
u have lost", reward - 1000, "points.")
            else:
                print("You have changed the course of game. It ends here. You ha\
ve lost", reward - 100, "points.")
        else:
            print("You have changed the course of game. It ends here. You have l\
ost", reward - 10, "points.")
    else:
        print("You have changed the course of game. It ends here and you have wo\
n no point.")
if __name__ == "__main__":
    main()
```

And the output on your Python Shell looks like this:

```
What happens when a Bad Man walks like a Robot?
Tell us what kind of change may take place inside him?
 Choose between 'monster' and 'angel',and type here...>>>>monster
You have won the first round: monster
You have won  1000 points.
What does he do? : he kills fellow people
Now tell us what the monster feels inside while killing people?
 Choose between 'great' and 'sad',and type here...>>>>great
You have won the second round:
You have won  10000 points.
What he feels inside? : he enjoys torturing animals
Tell us does the monster care for anyone?
 Choose between 'yes' and 'no',and type here...>>>>no
You have won the third round:
You have won  100000 points.
What he feels inside? : he cares for none
```

```
Tell us does the monster look like a normal human being?
 Choose between 'yes' and 'no',and type here...>>>>yes
You have won the fourth round:
You have won  1000000 points.
What does he look like? : he looks a normal human being
Tell us what happens to the monster finally? Does he destroy himself
 Choose between 'yes' and 'no',and type here...>>>>yes
You have won the fifth round:
You have won Jackpot. 100000000 points.
```

Since I had written the code, I won the game. But there are few tricks. In those tricky parts, if you failed and supplied wrong inputs you would lose.

Primary Class and Object

Now primary class and object should no longer be difficult. You can write a Human class and pass one default argument like “kind” in the initialization process. You can set it as “good”. Now if you want to create a good human being you need not pass any extra argument. In the next step, when you apply a verb like “BeingHuman()” to the good human being, it is by default good. If you want to create a bad human being you can change that deafult argument and make it bad.

```
#!/usr/bin/python3
class Human:
    def __init__(self, kind = "good"):
        self.kind = kind
    def BeingHuman(self):
        return self.kind
def main():
    good = Human()
    bad = Human("bad")
    print(good.BeingHuman ())
    print(bad.BeingHuman ())
if __name__ == "__main__":
    main()
```

The output is quite obvious:

```
good
bad
```

There are few things you need to understand. Why we use ‘self’? What does that mean? Consider the code below.

```
#!/usr/bin/python3
class MySelf:
    def __init__(self, name, quantity):
        self.name = name
        self.quantity = quantity
    def Eat(self):
        print(self.name, "eats", self.quantity, "bananas each day.")
def main():
    hagu = MySelf("Hagu", 2)
    mutu = MySelf("Mutu", 3)
    hagu.Eat()
    mutu.Eat()
if __name__ == "__main__":
    main()
```

In this code of class "MySelf" we have two methods. One is the special constructor method "**init**" and the other is "Eat()". You notice that each method has a special argument "self". Actually it references to the object that is going to be created. When we write a class, we assume that instances will be created. In this case, we created two objects "hagu" and "mutu". When we apply the verb "Eat()" or call the method, to the objects, it is as though they pass through the method. We set the names and and the numbers of bananas they eat. And the output of this code is like this:

```
Hagu eats 2 bananas each day.
Mutu eats 3 bananas each day.
```

But we need more concrete examples. We want to connect to our databases from our applications. To do that we need a class where we will have methods and prpperties that will connect to databases. Suppose we have two different set up. We have a MySQL database and besides we want to create a SqLite connection. To do that we can write two separate classes and set the connection in the constructor part or initialization method. So that when we create an instance the connection to the database is set up automatically.

Consider the code:

```
#!/usr/bin/python3
import sqlite3
import mysql.connector
from mysql.connector import Error
class MySQLiteConnection:
    def __init__(self):
        db = sqlite3.connect('testdb.db')
        db.commit()
        print("Connected to SqLite3")
class MyMySQLConnection:
    def __init__(self):
        try:
        ### you can either use a dictionary object or you can connect directly #\
##
        ### using a dictioanry object ###
            kwargs = dict(host = 'localhost', database = 'python_mysql', user = \
'root', password = 'pass')
            conn = mysql.connector.connect(**kwargs)
        ### connecting directly ###
            connection = mysql.connector.connect(host = 'localhost',
                                                 database = 'python_mysql',
                                                 user = 'root',
                                                 password = 'pass')
            if connection.is_connected():
                print("Connected to MySQL from 'conneection' object")
            # if conn.is_connected():
            #     print("Connected from 'conn' object")
        except Error as e:
            print(e)
        finally:
            connection.close()
def main():
    ConnectToMySQL = MyMySQLConnection()
    ConenctToSqLite = MySQLiteConnection()
if __name__ == "__main__":
    main()
```

We create two instances or objects of MyMySQLConnection() and MySqLiteConnection() classes and put them into two separate variables. Connections are being set up and in the output section we see this:

```
Connected to MySQL from 'connection' object
Connected to SqLite3
```

But this is extremely simple example and written badly. We should develop this code so that each instance of MySqlConnection and SqLiteConnection classes can not only connect to the database but also reteive data from a table.

Let us replace our old codse with this:

```
#!/usr/bin/python3

import sqlite3
import mysql.connector
from mysql.connector import MySQLConnection, Error


class MySQLiteConnection:
    def __init__(self, db = sqlite3.connect('test.db')):
        self.db = db
        db.row_factory = sqlite3.Row
        print("Connected to SqLite3")
    def Retrieve(self):
        print("Retreiving values from table test1 of SqLite database test")
        read = self.db.execute('select * from test1 order by i1')
        for row in read:
            print(row['t1'])
class MyMySQLConnection:
    def __init__(self,  kwargs = dict(host = 'localhost', database = 'testdb', u\
ser = 'root', password = 'pass')):
        try:
        ### you can either use a dictionary object or you can connect directly #\
##
        ### using a dictioanry object ###
            self.kwargs = kwargs
            conn = mysql.connector.connect(**kwargs)

            if conn.is_connected():
                print("Connected to MySql database testdb from 'conn' object")
        except Error as e:
            print(e)
        finally:
            conn.close()
    def Retrieve(self):
```

```
35          print("Retreiving records from MySql database testdb.")
36          try:
37              conn = MySQLConnection(**self.kwargs)
38              cursor = conn.cursor()
39              cursor.execute("SELECT * FROM EMPLOYEE")
40              rows = cursor.fetchall()
41              print('Total Row(s):', cursor.rowcount)
42              for row in rows:
43                  print("First Name = ", row[0])
44                  print("Second Name = ", row[1])
45                  print("Age = ", row[2])
46                  print("Sex = ", row[3])
47                  print("Salary = ", row[4])
48          except Error as e:
49              print(e)
50          finally:
51              cursor.close()
52              conn.close()
53  def main():
54      ConnectToMySQL = MyMySQLConnection()
55      ConnectToMySQL.Retrieve()
56      ConenctToSqLite = MySQLiteConnection()
57      ConenctToSqLite.Retrieve()
58  if __name__ == "__main__":
59      main()
```

We have connected to each database with the initialization process and then apply one verb "Retrieve()" to each object. We have also imported many database modules that you have not learned yet.

You will learn them in due process. But our purpose is served. We create two separate database objects. One is MySQL connection object and another is SqLite Connection object. After that with those objects we are able to retrieve separate data from two different tables.

First look at the output:

```
Connected to MySql database testdb from 'conn' object
Retreiving records from MySql database testdb.
Total Row(s): 3
First Name =  Mac
Second Name =  Mohan
Age =  20
Sex =  M
Salary =  2000.0
First Name =  Mac
Second Name =  Mohan
Age =  20
Sex =  M
Salary =  2000.0
First Name =  Mac
Second Name =  Mohan
Age =  20
Sex =  M
Salary =  2000.0
Connected to SqLite3
Retreiving values from table test1 of SqLite database test
Babu
Mana
Bappa
Babua
Anju
Patai
GasaBuddhu
Tapas
```

The output says, the MySQL database “testdb” has a table called “Employee” and there are several rows like name, sex, salary etc. Secondly we have a SqLite3 database “test1” which has a table called “test1” which has many rows that contain few names.

Accessing Object Data

When an object is created from a class it is quite obvious that it will have some kind of data. The question is how we can access that data? Whet is the proper way? We must access that data in a way so that we can keep a track of that. Consider this code below:

```
#!/usr/bin/python3
class Human:
    def __init__(self, height = 5.08):
        self.height = height
def main():
    ramu = Human()
    print(ramu.height)
    ramu.height = 5.11 # it is called side effect and hard to track
    print(ramu.height)
if __name__ == "__main__":
    main()
```

In this code we see Human class with a default height. Which is 5.08. When we create an object, this height is set automatically unless we change it or mention it explicitly. We can also set any property outside that object. In the next line we have written ramu.height = 5.11. We can set any object property like this. But this is called side effect and it is very hard to track. So we need to do that in a more structured manner. How we can do that? Let us see the output of this code first.

```
5.08
5.11
```

You see the height changes and we don't know what the proper height of object "ramu" is. To solve this problem, the accessor method is implemented. The accessor methods are methods that first set the value and next through that method you can get the value.

```
#!/usr/bin/python3
class Human:
    def __init__(self):
        pass
    # accessor
    def set_height(self, height):
        self.height = height
    def get_height(self):
        return self.height
def main():
    ramu = Human()
    # ramu.height = 5.11 # it is called side effect and hard to track
    ramu.set_height(5.12)
    print(ramu.get_height())
if __name__ == "__main__":
    main()
```

And the output is:

```
5.12
```

But still we lack at some point. Like we want to add more flexibilities so that with less amount of code we can get lot of jobs done.

```
#!/usr/bin/python3
class Human:
    def __init__(self, **kwargs):
        self.variables = kwargs
    def set_manyVariables(self, **kwargs):
        self.variables = kwargs
    def set_variables(self, key, value):
        self.variables[key] = value
    def get_variables(self, key):
        return self.variables.get(key, None)
def main():
    mana = Human(name = 'Mana')
    print("Object Mana's name:", mana.variables['name'])
    ManaName = mana.variables['name']
    mana.set_variables('class', 'two')
    print(ManaName, "reads at class", mana.get_variables('class'))
    mana.set_manyVariables(school = 'balika school', height = 4.54)
    print(ManaName, "has height of", mana.variables['height'], "and her school's\
 name is", mana.variables['school'])
    babu = Human(name = 'Babu', student_of = 'Class Three', reads_at = ' Balak S\
chool', height = 5.21)
    BabuName = babu.variables['name']
    print(BabuName, "he is a student of", babu.variables['student_of'], "and he \
reads at",
          babu.variables['reads_at'], "and his height is", babu.variables['heigh\
t'])
if __name__ == "__main__":
    main()
```

In this code snippet we have many options open to us. We have set our variables in a dictionary format. After that we can get the value through the key.

```
Object Mana's name: Mana
Mana reads at class two
Mana has height of 4.54 and her school's name is balika school
Babu he is a student of Class Three and he reads at Balak School and his height \
is 5.21
```

This is not the only method to tackle object data. As you progress you will see a lot of different examples of handling data.

Polymorphism

Polymorphism is a very important concept in object oriented programming. The basic thing is when we apply same verb on two different objects, depending on the objects, they react differently. When we put up an old house on sale it fetches a certain value. But when we put up a new house on sale it fetches more price and value. So in this case when we apply "sale" method or "sale" verb to different objects they behave differently.

```
#!/usrbin/python3
class Table:
    def __init__(self):
        pass
    def ItHolds(self):
        print("A table holds books, writing pads on it.")
    def YouCanWriteOnit(self):
        print("You can write on a table.")

class Book:
    def __init__(self):
        pass
    def ItHelps(self):
        print("A book helps us to know something new.")

def main():
    MyTable = Table()
    MyBook = Book()
    MyTable.ItHolds()
    MyTable.YouCanWriteOnit()
    MyBook.ItHelps()
if __name__ == "__main__":
    main()
```

These are quite simple classes and the output is also very simple.

```
A table holds things on it.
You can write on a table.
A book helps us to know something new.
```

This output may change drastically when apply same verbs or methods to the objects of “Table” and “Book” classes. Consider the following codes.

```
#!/usrbin/python3
class Table:
    def __init__(self):
        pass

    def Get(self):
        print("Please get me that table.")
    def Put(self):
        print("Please put the table on the corner of the room.")
    def Destroy(self):
        print("Some people came and they did not want us to read and write. They\
 destroted the table.")
class Book:
    def __init__(self):
        pass
     def Get(self):
        print("Please get me that book.")
    def Put(self):
        print("We put some new books on the table.")
    def Destroy(self):
        print("Some people came and they did not want us to read and write. They\
 destroyed the book.")
def main():
    MyTable = Table()
    MyBook = Book()
    InMistake(MyBook)
    Intentionally(MyTable)
def InMistake(Table):
    Table.Get()
    Table.Put()
    Table.Destroy()
def Intentionally(Book):
    Book.Get()
    Book.Put()
    Book.Destroy()
```

```
36 if __name__ == "__main__":
37     main()
```

There are three methods : Get, Put and Destroy. You see how the table and book objects react differently to those methods.

```
1 Please get me that book.
2 We put some new books on the table.
3 Some people came and they did not want us to read and write. They destroyed the \
4 book.
5 Please get me that table.
6 Please put the table on the corner of the room.
7 Some people came and they did not want us to read and write. They destroyed the \
8 table.
```

Using Generators

In Python a generator object is used in a context where iteration is necessary. Normally in this case, we rely on two methods: def **init**(self, *args) and def **iter**(self). We set the logic in the constructor method and iterate through it by the def **iter**(self) function.

```
1  #!/usr/bin/python3
2  class InclusiveRange:
3      def __init__(self, *args):
4          numberOfArguments = len(args)
5          if numberOfArguments < 1: raise TypeError('At least one argument is requ\
6  ired.')
7          elif numberOfArguments == 1:
8              self.stop = args[0]
9              self.start = 0
10             self.step = 1
11         elif numberOfArguments == 2:
12             # start and stop will be tuple
13             (self.start, stop) = args
14             self.step = 1
15         elif numberOfArguments == 3:
16             # all start and stop and step will be tuple
17             (self.start, self.stop, self.step) = args
18         else: raise TypeError("Maximum three arguments. You gave {}".format(numb\
19 erOfArguments))
20
```

```
    def __iter__(self):
        i = self.start
        while i <= self.stop:
            yield i
            i += self.step

def main():
    ranges = InclusiveRange(5, 210, 10)
    for x in ranges:
        print(x, end=' ')
if __name__ == "__main__":
    main()
```

This code means you can control the range of iteration. We start from 5 and then end at 210. In each step we progress by 10.

```
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145 155 165 175 185 195 205
```

We can get the same effect without using those methods. We can simply write this way.

```
    ## the function below is perfectly working also but that is not a generator \
##

    def RangeFunctions(self, *args):
         numberOfArguments = len(args)
         if numberOfArguments < 1: raise TypeError('At least one argument is req\
uired.')
         elif numberOfArguments == 1:
             self.stop = args[0]
             self.start = 0
             self.step = 1
         elif numberOfArguments == 2:
             # start and stop will be tuple
             (self.start, stop) = args
             self.step = 1
         elif numberOfArguments == 3:
             # all start and stop and step will be tuple
             (self.start, self.stop, self.step) = args
       else: raise TypeError("Maximum three arguments. You gave {}".format(numbe\
rOfArguments))

         i = self.start
```

```
23          while i <= self.stop:
24              yield i
25              i += self.step
```

Decorator

Decorators are special functions that return functions. Normally to set a property of object we usually get it through another function.

```
1  #!/usr/bin/python3
2  class Dog:
3
4      def __init__(self, **kwargs):
5          self.properties = kwargs
6      def get_properties(self):
7          return self.properties
8      def set_properties(self, key):
9          self.properties.get(key, None)
10
11 def main():
12     lucky = Dog(nature = 'obedient')
13     print(lucky.properties.get('nature'))
14
15 if __name__ == "__main__":
16     main()
```

The output is quite simple.

```
1  obedient
```

In Python ‘Decorator’ is simply a method by which we decorate an accessor method for variable. And the function starts behaving like a property. The beauty of this decorator is, you can use the function as property and after creating the object you can control the property – setting and getting it. See the following code.

```
#!/usr/bin/python3
class Dog:

    def __init__(self, **kwargs):
        self.properties = kwargs
    @property
    def Color(self):
        return self.properties.get('color', None)
    @Color.setter
    def Color(self, color):
        self.properties['color'] = color
    @Color.deleter
    def Color(self):
        del self.properties['color']
def main():
    lucky = Dog()
# now we are going to use the decorator function as a normal property
    lucky.Color = 'black and yellow'
    print(lucky.Color)

if __name__ == "__main__":
    main()
```

The output is as expected:

```
black and yellow
```

It is a very simple example where we see that an usual syntax of function can be written as a property syntax. It is more convenient when we use this decorator method in saving files inside a database.

In the last chapter, when we will see the web application of “Flask”, we will see how we can use this decorator to route our web pages.

File Input, Output

Python has some built in functions for dealing with files. You can open a file and read what is inside. You can write a file. That file could be a text file or a picture.

Each time we use the open() method and pass the mode as an argument. For reading a file we write ‘r’ and for write we use ‘w’. Let us consider a code where in an object we read a file and write it on another file using another object in the next step.

```
infile = open('files.txt', 'r')
outfile = open('new.txt', 'w')
for line in infile:
    print(line, file=outfile)
print("Done")
```

If we copy this way the file size is increased in the new text file. Now we have a comparatively large file. ‘Files.txt’ is now ‘5.4 KB’ and the ‘new.txt’ is only 134 bytes.

If we copy by the old way the new file becomes ‘5.7 KB’, a little bit larger than the former one. But Python has the technique to copy by buffer so that the buffer size remains intact.

Now we are going to write contents of ‘files.txt’ into ‘new.txt’, but not by the old way. The new code is :

```
BufferSize = 500000
infile = open('files.txt', 'r')
outfile = open('new.txt', 'w')
buffer = infile.read(BufferSize)
while len(buffer):
    outfile.write(buffer)
    print("It is copying, it might take some time...please wait....", end='')
    buffer = infile.read(BufferSize)
print()
print("Copying Done.")
```

The output is as expected.

```
It is copying, it might take some time...please wait....
Copying Done.
```

Reading and writing binary file is same. All you need to do is you have to change the mode from ‘r’ to ‘rb’ and change mode ‘w’ to ‘wb’. That is it. Your code looks like that:

```
BufferSize = 5000000
infile = open('home.jpg', 'rb')
outfile = open('newimageofHome.jpg', 'wb')
buffer = infile.read(BufferSize)
while len(buffer):
    outfile.write(buffer)
    print("It is copying an image, it might take some time...please wait....", e\
nd='')
    buffer = infile.read(BufferSize)
print()
print("Copying Done.")
```

Containers

In Python tuples and lists are array types. Tuples are immutable but lists are mutable. Tuples are used with comma operator and you can iterate through the tuple quite easily. As tuples are immutable, you can not add or update the value of tuple. In lists, you can update or add new value quite easily. Open up your terminal in Linux and IDLE in Windows. Write down the code below and see the output yourself. Please read the comments that are attached with the code.

```
#!/usr/bin/python3
tuples1 = 1, 2, 3, 4
print(type(tuples1))
print(id(tuples1))
tuples2 = (1, 2, 3, 4)
print(type(tuples2))
print(id(tuples2))
print(tuples1[0])
print(tuples2[0])
# it will give the last item
print(tuples2[-1])
print(type(tuples1[0]))
print(type(tuples2[0]))
print(id(tuples1[0]))
print(id(tuples2[0]))
# tuple is immutable we can not change any value
# 'tuple' object does not support item assignment
# tuples2[0] = 120
# print(tuples2)
# to make an integer tuple you need to add comma separator
IsItTuple = (1)
print(type(IsItTuple))
IsItTuple = (1,)
print(type(IsItTuple))
# let us see how list behaves
list1 = [1, 2, 3, 4]
print(type(list1))
print(id(list1))
# first item
print(list1[0])
# last item
```

```
print(list1[-1])
# we can change the value of a list item
list1[0] = 120
print(list1) # output: [120, 2, 3, 4]
```

The output is like this:

```
<class 'tuple'>
139794725901080
<class 'tuple'>
139794725900920
1
1
4
<class 'int'>
<class 'int'>
10455040
10455040
<class 'int'>
<class 'tuple'>
<class 'list'>
139794725273480
1
4
[120, 2, 3, 4]
```

Tuple and List Object

Let us open up our terminal and test how tuples and lists work together.

```
root@kali:~# python3
Python 3.4.4 (default, Jan  5 2016, 15:35:18)
[GCC 5.3.1 20160101] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> t = (1,2,3,4)
>>> t
(1, 2, 3, 4)
>>> t[0]
1
>>> t = tuple(range(25))
>>> type(t)
```

```
<class 'tuple'>
>>> 50 in t
False
>>> 10 in t
True
>>> for i in t:print(i)
...
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
>>> l = list(range(20))
>>> type(l)
<class 'list'>
>>> for i in l:
... print(i)
  File "<stdin>", line 2
    print(i)
        ^
IndentationError: expected an indented block
>>> for i in l:print(i)
```

```
...
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
>>> l[2]
2
>>> 50 in l
False
>>> 12 in l
True
>>> t[0] = 25
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> l[0] = 25
>>> print(l)
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> t.append(50)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> l.append(120)
>>> print(l
l         lambda    len(      license(  list(     locals(
>>> print(l)
```

```
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 120]
>>> t.count()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: count() takes exactly one argument (0 given)
>>> t.count(5)
1
>>> l.append(25)
>>> l.count(25)
2
>>> t.index(10)
10
>>> l.index(10)
10
>>> l.extend(range(25))
>>> for i in l:print(i)
...
25
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
120
25
0
1
2
```

```
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
>>> l.insert(0, 4656)
>>> l[0]
4656
>>> l.insert(12, 147)
>>> l.index(12)
14
>>> l[12]
147
>>> l.remove(12)
>>> l[12]
147
>>> print(l)
[4656, 25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11, 13, 14, 15, 16, 17, 18, 19, 1\
20, 25, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20\
, 21, 22, 23, 24]
>>> l.remove(12)
>>> print(l)
[4656, 25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11, 13, 14, 15, 16, 17, 18, 19, 1\
20, 25, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21\
, 22, 23, 24]
```

```
>>> l.pop(0)
4656
>>> print(l)
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11, 13, 14, 15, 16, 17, 18, 19, 120, 25\
, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, \
23, 24]
>>> l.pop()
24
>>> print(l)
[25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 147, 11, 13, 14, 15, 16, 17, 18, 19, 120, 25\
, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, \
23]
>>>
```

Write down the same code and see how they work in your machine. Errors may come out as happened in the above code. But remember, each error will help you to learn few new things.

Dictionary Object

Like you have tested tuples and lists, you can test the dictionary object and see how it works.

```
root@kali:~# python3
Python 3.4.4 (default, Jan  5 2016, 15:35:18)
[GCC 5.3.1 20160101] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = {'one':1, 'two':2, 'three':3}
>>> type(x)
<class 'dict'>
>>> y = dict(four = 4, five = 5, six = 6)
>>> type(y)
<class 'dict'>
>>> z = dict(seven = 7, eight = 8, nine = 9, **x, **y)
  File "<stdin>", line 1
    z = dict(seven = 7, eight = 8, nine = 9, **x, **y)
                                               ^
SyntaxError: invalid syntax
>>> z = dict(seven = 7, eight = 8, nine = 9, **x)
>>> type(z)
<class 'dict'>
>>> print(z)
{'eight': 8, 'two': 2, 'nine': 9, 'one': 1, 'seven': 7, 'three': 3}
```

```
>>> for i in z:print(i)
...
eight
two
nine
one
seven
three
>>> for key, value in z.items():print(key, value)
...
eight 8
two 2
nine 9
one 1
seven 7
three 3
>>> for key, value in z.items():
...   if key == two:
...     print(value)
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
NameError: name 'two' is not defined
>>> z.pop()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: pop expected at least 1 arguments, got 0
>>> z.pop(three)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'three' is not defined
>>> z.pop('three')
3
>>> for i in z:print(i)
...
eight
two
nine
one
seven
>>> for key, value in z.items():
...   if key == 'nine':
```

```
...     print(value)
...
9
>>>
```

More you spend time with tuples, lists and dictionaries, more you learn about Python. There is lot of built in functions and you can use those functions quite easily to get more out of your code. Another key concept of dictionary is â€œkey⇒valueâ€ pair. As you progress further and learn more languages along with Python, you will find that each language uses this concept taking it further to solve major problems. Especially the web frameworks use this concept very heavily.

Module

In Python when you leave the shell or terminal or Python interpreter, the script is lost. After all you don't write programs to lose at the end of the day. It may be a simple calculator program. But you want to use it again. Another important thing is you need to use your one code in your other code. You may want to use other people's code also.

To solve this dilemma the concept of "Module" comes in.

You write a simple calculator program and save the file as "cal.py". If you are in the root directory of your project you can easily use your calculator in your other program. Once you write a Python code and save it with a name that name becomes a module.

In this case, "cal" becomes a module. Now you can "import" that "cal" module into any other code or module. In the large Python library there are tons of modules. You can always import them and use them. Consider the code below. In this code we have imported three modules. The first is "sys" or system specific module. The second one is "os" or operating system specific module and the third one is "urllib" which means a library that is URL specific. You notice that we write "urllib.request" the "DOT" notation means we actually call something called "request" from the Python URL library. Actually the web architecture primarily depends upon two things – request and respond. Here we are going to request something from a URL.

```
1  #!/usr/bin/python3
2  import sys, os, urllib.request
3  def main():
4      print("This is Python Version : {}.{}.{}".format(*sys.version_info))
5      # os module
6      print(os.name)
7      print(os.getenv('PATH'))
8      print(os.getcwd())
9
10     #urllib module
11     page = urllib.request.urlopen('http://arshinagar.in/')
12     for line in page:
13         print(str(line, encoding='utf-8'), end='')
14 if __name__ == "__main__":
15     main()
```

You see that in the first part of the code we have used the "sys" module and wanted to know the version of Python our system is using. The second part is all about the operating system. It gives us name, path and many other things.

And in the last part we are requesting a web page. Let us see the output in a Linux Debian distribution like Ubuntu first. The first line is the version and the second line is about the operating system which is “posix”. Third line is the environment path and the fourth line is the actual path where this file is stored.

From the fifth line you see the “urllib.request” starts working in and fetches the whole index page from a web site. I have used one web site of my friend. I do not print out the whole HTML output as it would take lots of space. Go through each line and see how different module work.

```
This is Python Version : 3.4.3
posix
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc\
al/games
/home/hagudu/PycharmProjects/FirstPythonProject/modules
<!DOCTYPE html>
<html  lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width" />
<meta name="viewport" content="initial-scale=1.0" />
<meta name="HandheldFriendly" content="true"/>
<link rel="profile" href="http://gmpg.org/xfn/11" />
<link rel="pingback" href="http://www.arshinagar.in/xmlrpc.php" />
<title>Arshinagar &#8211; Just another WordPress site</title>
<link rel="alternate" type="application/rss+xml" title="Arshinagar &raquo; Feed"\
 href="http://www.arshinagar.in/feed/" />
<link rel="alternate" type="application/rss+xml" title="Arshinagar &raquo; Comme\
nts Feed" href="http://www.arshinagar.in/comments/feed/" />
//the details are removed for brevity
Process finished with exit code 0
```

Now we can try this same code in Windows and compare the output.

```
This is Python Version : 3.4.4
nt
C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\Microso\
ft SQL Server\90\Tools\binn\
D:\pthon-files-fromwindows
```

In this output you see the Python version has been changed. The operating system is not “posix” anymore. It is “nt” now. The environment path and the file path are also poles apart. I removed the “urllib.request” module output for concision.

We can see more module examples here.

```
#!/usr/bin/python3
import sys, os, urllib.request, random, datetime
def main():
    print("This is Python Version : {}.{}.{}".format(*sys.version_info))

    # random module
    print(random.randint(1, 1000))
    x = list(range(25))
    print(x)
    random.shuffle(x)
    print(x)
    random.shuffle(x)
    print(x)
    random.shuffle(x)
    print(x)
    PresentTime = datetime.datetime.now()
    print(PresentTime)
    print(PresentTime.year, PresentTime.month, PresentTime.day, PresentTime.hour\
, PresentTime.minute, PresentTime.second, PresentTime.microsecond)

if __name__ == "__main__":
    main()
```

In this code we add two more modules. They are “random” and “datetime”. We get the output below to see how they work.

```
This is Python Version : 3.4.3
366
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2\
2, 23, 24]
[23, 6, 22, 3, 7, 19, 10, 16, 8, 12, 15, 21, 11, 17, 9, 13, 4, 14, 24, 18, 0, 2,\
 1, 20, 5]
[0, 8, 21, 5, 13, 3, 2, 18, 24, 12, 4, 19, 14, 17, 20, 10, 11, 22, 15, 9, 6, 23,\
 1, 7, 16]
[11, 6, 23, 14, 9, 7, 3, 5, 15, 2, 19, 0, 16, 24, 21, 12, 4, 13, 22, 20, 10, 8, \
1, 17, 18]
//here is the outout of date and time module
2016-03-23 08:34:37.253888
2016 3 23 8 34 37 253888
```

Each time you run the code, you get a new number as the “random” module produces new numbers always. To get more idea you need to go through the Python Standard Library in the official python

web site or download the Python 3.4.4 Documentation. It is available in many types including simple text file or PDF version. You can check the “date time” module page in Python Standard Library in the documentation page. Now you can go back to your old codes and see them again in a new light. Now you will easily understand why we have used MYSQL connector module or configuration parser module.

Debugging, UnitTest

You have progressed a lot and now you can tackle more complex issues. In the process of coding you must have found or seen many type of errors. It is quite obvious. The seasoned programmers also do mistakes. You have also learned to catch your errors. But situation may come when you need to update your code. It might happen. You need to modify or add a few lines in your code. It may either work or it may fail. In your new lines of code there could be "syntactical" errors. There could be "runtime" errors. Usually Python interpreter tries to guide you in such cases. Generally it points out where the error is occurred. But not always.

In such cases the "UnitTest" module comes to your help.

In Python standard library you get a lot of information about this module. You may also search about "nose" tool over the internet that also do something similar. The basic concept is, you have a code repository at somewhere and you have a separate unit testing schedule. It is an automated test. Suppose we have a folder called "MyTest/BrainAndSoul". Inside this folder we have a Python file called "saytimedate.py". It is a very simple file that will tell us the version of Python and the present time and date. To get that output, we need two modules: "sys" and "datetime". In two methods we have get those outputs. To ge the output all we need to do is call them under "main()" function. We exactly do that.

At the same time we have two separate methods that begin with the word "test". The methods are "test_PyVar()" and "test_main()".

```
1  #!/usr/bin/python3
2  # coding=utf-8
3  import sys, datetime
4  def PyVer():
5      print("This is Python Version : {}.{}.{}".format(*sys.version_info))
6  def PyTime():
7      PresentTime = datetime.datetime.now()
8      print(PresentTime)
9      print(PresentTime.year, PresentTime.month, PresentTime.day, PresentTime.hour\
10 , PresentTime.minute,
11             PresentTime.second, PresentTime.microsecond)
12     #print(obj)
13 def main():
14     PyVer()
15     PyTime()
16 def test_Pyvar():
17     PyVer()
```

```
18 def test_Main():
19     PyTime()
20 if __name__ == "__main__":
21     main()
```

When you run this code your main() function calls the two methods defined inside it. And the output below is quite expected.

```
1 This is Python Version : 3.4.2
2 2016-04-22 23:30:30.435691
3 2016 4 22 23 30 30 435691
```

Now in a completely separate folder we would like to run the "unittest" module and see whether this code passes or fails. Since we have already run the code and got a successful output, so we can safely say that this code will pass the test.

The name of our unit testing code is "TestUnitTest.py" and the code is like this:

```
1  #!/usr/bin/python3
2  # coding=utf-8
3  import MyProject.BrainAndSoul.saytimedate
4  import unittest
5  class SayTiemDate(unittest.TestCase):
6      def setUP(self):
7          pass
8      def test_Version(self): self.assertEqual(MyProject.BrainAndSoul.saytimedate.\
9  PyVer(), MyProject.BrainAndSoul.saytimedate.test_Pyvar())
10
11     def test_Time(self): self.assertEqual(MyProject.BrainAndSoul.saytimedate.mai\
12 n(), MyProject.BrainAndSoul.saytimedate.test_Main())
13
14 if __name__ == "__main__":
15     unittest.main()
```

What does this code say? As you see, there are two methods "test_Time()" and "test_Version()". We have not passed any argument. Both the methods call one default method from the "unittest" module. And that is "assertEqual()". Through this method we have passed two methods that we have defined earlier in the "MyTest/BrainAndSoul" folder. Inside that folder we have a Python file called "saytimedate.py". We are now comparing two methods through our "unittest" module. Finally it gives a nice output like this if everything runs properly.

```
Testing started at 8:58 PM ...
This is Python Version : 3.4.2
This is Python Version : 3.4.2
Process finished with exit code 0
```

If we run that code again we may get an output like this:

```
This is Python Version : 3.4.2
2016-04-23 05:47:23.608853
2016 4 23 5 47 23 608853
2016-04-23 05:47:23.608951
2016 4 23 5 47 23 608951
This is Python Version : 3.4.2
This is Python Version : 3.4.2
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK

Process finished with exit code 0
```

Now for testing purpose we change in our source code and do some mistakes intentionally to see whether our “unittest” module fails or not.

If there is any error, the output will change and gives an error message something like this:

```
This is Python Version : 3.4.2
2016-04-23 05:51:45.994547
2016 4 23 5 51 45 994547
This is Python Version : 3.4.2
This is Python Version : 3.4.2
E.
======================================================================
ERROR: test_Time (__main__.SayTiemDate)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/ss/FirstPythonProjects/PlayWithPython-master/MyTest/TestUnitTest.p\
y", line 17, in test_Time
    self.assertEqual(MyProject.BrainAndSoul.saytimedate.main(), MyProject.BrainA\
ndSoul.saytimedate.test_Main())
  File "/home/ss/FirstPythonProjects/MyProject/BrainAndSoul/saytimedate.py", lin\
```

```
e 20, in main
    PyTime()
  File "/home/ss/FirstPythonProjects/MyProject/BrainAndSoul/saytimedate.py", lin\
e 15, in PyTime
    print(obj)
NameError: name 'obj' is not defined

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (errors=1)

Process finished with exit code 1
```

Now you can try to run more unit testing module. Here is another example where the test is successful.

PART THREE: PYTHON AND SECURITY ANALYSIS, RECONNAISSANCE SCANNER

In this part we will build some reconnaissance scanner through which we could gather information about any website. And that job includes whois analysis and checking the activity of ‘robots.txt’ file. Normally webmasters use ‘robots.txt’ file to decline directory or file access. Gathering ‘whois’ information actually stands for passive information gathering.

Python has a huge trove of tools for ethical hacking purpose. To name a few there are socket networking and NMAP scanning and many more. We’ll also learn about them in great detail in the following chapters.

Socket and Networking

This chapter is kind of introduction to the advanced concepts of Python. Since this is the final chapter, I would like to tell you about what you can go from here. You can either build web applications with the help of Python. You can do some Security Networking staff. Finally like me, you can choose the interesting path of Ethical Hacking. All these things and more you can do through Python and many more.

Let us have how we can apply our basic knowledge of Python in Socket and Networking. Write down this code on your IDE and see what output you get.

```
# coding=utf-8
import socket
print(socket.gethostbyname("www.mesanjib.wordpress.com"))
print(socket.gethostbyname("www.sanjib.pythonanywhere.com"))
```

The output is like this in my machine. You can test any other web site to get their addresses. This is tip of the iceberg. Lot of things are inside. It is better to see everything is inside that I can not tell you as I feel you should concentrate on trying to write basic concepts of Python more and more.

```
192.0.78.12
50.19.109.98
```

Process finished with exit code 0

In the further study of the relationship between Ethical Hacking and Python 3, youâ€™ll find this socket and networking concepts extremely useful. Let us move further to the Part three pf the book where weâ€™ll learn about anonymity.

An Ethical Hacker should always remain anonymous.

Why? Let us see.

Importing Nmap Module

Nmap (Network Mapper) is a security scanner. It was originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich). This tool is particularly used to discover hosts and services on a computer network. While finding the hosts and services it creates a “map” of the network. For this reason it has been widely called ‘Nmap’ or you can also call it ‘Network Mapper’. It is regarded as an essential tool in your pursuit to be a good and competent ethical hacker.

To get the best results, Nmap usually sends specially crafted packets to the target host and then analyzes the responses and finds what ports are open. It also assesses the vulnerability of a computer network.

This software widely used by the hackers has number of features. It actually probes computer networks, discovering hosts and services. It also detects operating system and it decides the vulnerability of the systems by finding the open ports.

Python actually extends these features so that you can easily do more advanced service detection, vulnerability detection and other things.

Let us first check whether ‘Nmap’ module of python has already been installed in our system or not by issuing a simple command on the terminal.

```
nmap
```

It gives us a long listing which is very important. Many things you can learn from this listing as it says about the version, usages and at the end it also says where you can get the manual for more reading. The output is too long and may seem boring but each line is important if you want to master NMAP scanning. Remember, NMAP scanning is a part of reconnaissance which covers almost ninety percent of any type of hacking activities.

```
Nmap 6.40 ( http://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
```

```
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
  -sC: equivalent to --script=default
  --script=<Lua scripts>: <Lua scripts> is a comma separated list of
          directories, script-files or script-categories
  --script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
  --script-args-file=filename: provide NSE script args in a file
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --script-help=<Lua scripts>: Show help about scripts.
          <Lua scripts> is a comma separted list of script-files or
```

```
53            script-categories.
54  OS DETECTION:
55    -O: Enable OS detection
56    --osscan-limit: Limit OS detection to promising targets
57    --osscan-guess: Guess OS more aggressively
58  TIMING AND PERFORMANCE:
59    Options which take <time> are in seconds, or append 'ms' (milliseconds),
60    's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
61    -T<0-5>: Set timing template (higher is faster)
62    --min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
63    --min-parallelism/max-parallelism <numprobes>: Probe parallelization
64    --min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
65        probe round trip time.
66    --max-retries <tries>: Caps number of port scan probe retransmissions.
67    --host-timeout <time>: Give up on target after this long
68    --scan-delay/--max-scan-delay <time>: Adjust delay between probes
69    --min-rate <number>: Send packets no slower than <number> per second
70    --max-rate <number>: Send packets no faster than <number> per second
71  FIREWALL/IDS EVASION AND SPOOFING:
72    -f; --mtu <val>: fragment packets (optionally w/given MTU)
73    -D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
74    -S <IP_Address>: Spoof source address
75    -e <iface>: Use specified interface
76    -g/--source-port <portnum>: Use given port number
77    --data-length <num>: Append random data to sent packets
78    --ip-options <options>: Send packets with specified ip options
79    --ttl <val>: Set IP time-to-live field
80    --spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
81    --badsum: Send packets with a bogus TCP/UDP/SCTP checksum
82  OUTPUT:
83    -oN/-oX/-oS/-oG <file>: Output scan in normal, XML, s|<rIpt kIddi3,
84       and Grepable format, respectively, to the given filename.
85    -oA <basename>: Output in the three major formats at once
86    -v: Increase verbosity level (use -vv or more for greater effect)
87    -d: Increase debugging level (use -dd or more for greater effect)
88    --reason: Display the reason a port is in a particular state
89    --open: Only show open (or possibly open) ports
90    --packet-trace: Show all packets sent and received
91    --iflist: Print host interfaces and routes (for debugging)
92    --log-errors: Log errors/warnings to the normal-format output file
93    --append-output: Append to rather than clobber specified output files
94    --resume <filename>: Resume an aborted scan
```

```
  --stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
  --webxml: Reference stylesheet from Nmap.Org for more portable XML
  --no-stylesheet: Prevent associating of XSL stylesheet w/XML output
MISC:
  -6: Enable IPv6 scanning
  -A: Enable OS detection, version detection, script scanning, and traceroute
  --datadir <dirname>: Specify custom Nmap data file location
  --send-eth/--send-ip: Send using raw ethernet frames or IP packets
  --privileged: Assume that the user is fully privileged
  --unprivileged: Assume the user lacks raw socket privileges
  -V: Print version number
  -h: Print this help summary page.
EXAMPLES:
  nmap -v -A scanme.nmap.org
  nmap -v -sn 192.168.0.0/16 10.0.0.0/8
  nmap -v -iR 10000 -Pn -p 80
SEE THE MAN PAGE (http://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
```

You can get more about Network Mapper in the internet. Please follow these links.

```
http://nmap.org/
http://nmap.org/book/man.html
https://nmap.org/book/inst-other-platforms.html
https://nmap.org/book/inst-windows.html
https://nmap.org/book/vscan.html
```

If in your 'Linux' version of default operating system you don't get this listing you can install 'Nmap' by issuing a simple command.

```
sudo apt-get install nmap
```

In your virtual machine if you run kali Linux, you'll find that 'Nmap' has already been installed. Now after this installation part is over we can very quickly have a short python script to see how our 'Nmap' module is working.

You've already learned how to use 'nano' text editor on your terminal. So open it up with this command:

```
sudo nano test.py
```

It will first ask for your root password and then open up the nano text editor on your terminal. Write a short script like this:

```
#!/usr/bin/python
import nmap
nm = nmap.PortScannerAsync()
def callback_result(host, scan_result):
    print ('------------------')
    print (host, scan_result)
nm.scan('127.0.0.1', arguments="-O -v", callback=callback_result)
while nm.still_scanning():
    print("Waiting >>>")
    nm.wait(2)
nm1 = nmap.PortScanner()
a = nm1.nmap_version()
print (a)
```

If you run your ‘test.py’ script, you’d get this output:

```
Waiting >>>
------------------
('127.0.0.1', None)
(6, 40)
```

It’s your localhost address. But we are interested about the remote target. Run up the kali Linux in your Virual Box and open the ‘Tor’ browser. Search ‘what is my ip address’.

It will give you an anonymous IP address all the time. Each time you search that IP address changes. In your case it may come out as:

```
x.x.xx.xxx
ISP: Some Internet LTD
```

It’s usually too far from your original location! Anyway, you can test the IP and see the result. But it’s a good practice to test the IP of http://nmap.org

Nmap Network Scanner

Now we're ready to do more network testing using python scripts. And this time we'll try to build up a more robust scanner and we'll also try to detect the open ports and see if there are any vulnerabilities.

Let us write the python script first. And after that we'll see the output. Let us change the 'test.py' script to this:

```
 #!/usr/bin/python
import nmap
nm = nmap.PortScanner()
print (nm.nmap_version())
nm.scan('x.x.xx.xxx', '1-1024', '-v')
print(nm.scaninfo())
print(nm.csv())
```

Here '-v' stands for version and the '1-1024' stands for the range of the port numbers. It's a very small script but see the power of it in the output.

```
hagudu@hagudu-H81M-S1:~$ ./test.py
(6, 40)
{'tcp': {'services': '1-1024', 'method': 'connect'}}
host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;reason;ve\
rsion;conf;cpe
x.x.xx.xxx;host3.x0x;PTR;tcp;22;ssh;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;25;smtp;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;53;domain;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;80;http;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;137;netbios-ns;filtered;;;no-response;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;138;netbios-dgm;filtered;;;no-response;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;139;netbios-ssn;filtered;;;no-response;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;445;microsoft-ds;filtered;;;no-response;;3;
```

It shows that all together four ports are open. They are: 22, 25, 53 and 80. And the others are filtered.

Before going to test another port and this time we can show the IP as it's of http://nmap.org, let us have a very quick facts about the port terminology. You can also find the legal side of scanning explained here: https://nmap.org/book/legal-issues.html.

Port is an addressable network location. It's ideally implemented inside the operating system and this OS helps us to discriminate web traffic. This traffic is destined for different applications or services, like some for 'mail', some for 'HTTP' and so and so.

Next we're interested about the Port scanning. In one word, it's a type of process and this process usually tries to connect to a number of sequential ports, as you have just seen in the above output. We want to know which ports are open and what services and operating system are behind them.

Let us scan another IP address (http://nmap.org) and in doing that we have changed the python script a little bit.

```
1 #!/usr/bin/python
2 import nmap
3 nm = nmap.PortScanner()
4 print (nm.nmap_version())
5 nm.scan('192.168.146.1', '1-1024', '-v')
6 print(nm.scaninfo())
7 print(nm.csv())
```

The output is like this:

```
1 (6, 40)
2 {'tcp': {'services': '1-1024', 'method': 'connect'}}
3 host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;reason;ve\
4 rsion;conf;cpe
5 192.168.146.1;;;tcp;25;smtp;open;;;syn-ack;;3;
6 192.168.146.1;;;tcp;53;domain;open;;;syn-ack;;3;
7 192.168.146.1;;;tcp;80;http;open;;;syn-ack;;3;
```

The open ports are 25, 53 and 80. There are no filtered ports showing on this machine. Let us get all hosts from that IP with a little change in our previous script. This time we reduce the range so that our program won't run for long.

```
1 #!/usr/bin/python
2 import nmap
3 nm = nmap.PortScanner()
4 print (nm.nmap_version())
5 nm.scan('192.168.146.1', '22-455', '-v --version-all')
6 print(nm.all_hosts())
```

We have changed the number of ports in line number five. We also removed last two lines and want to see if we can get more data from that machine. The output shows that there is only one host.

```
(6, 40)
{'tcp': {'services': '22-455', 'method': 'connect'}}
['192.168.146.1']
```

Let us change and go back to the previous IP and see the output.

```
#!/usr/bin/python
import nmap
nm = nmap.PortScanner()
print (nm.nmap_version())
nm.scan('x.x.xx.xxx', '22-455', '-v --version-all')
print(nm.all_hosts())
```

Nothing changes. The output tells us about the only one host. There are more to come.

As we want more information we should ideally change our 'test.py' code.

```
#!/usr/bin/python
import nmap
nm = nmap.PortScanner()
print (nm.nmap_version())
nm.scan('192.168.146.1', '22-1024', '-v --version-all')
print (nm.scanstats())
print (nm['192.168.146.1'].state())
print (nm['192.168.146.1'].all_protocols())
print (nm['192.168.146.1']['tcp'].keys())
```

This time the output is more verbose.

```
(6, 40)
{'uphosts': '1', 'timestr': 'Mon Oct  3 09:53:35 2016', 'downhosts': '0', 'total\
hosts': '1', 'elapsed': '5.73'}
up
['tcp']
[80, 25, 53]
```

You see that one host is up.

There is no down-hosts and the number of total host is 1 as expected. We also see the exact time when the scan is being executed and the time elapsed. Let us dig a bit further.

We have used the port range '1-1024'. Normally ports below 1024 are associated with Linux and Unix like services. These operating systems are considered to be vital for essential network functions. For that reason you must have root privileges to assign services to these types of OS.

If you want to go beyond 1024, there are either ‘registered’ or ‘private’ ports. Ports between 49152 and 65535 are supposed to be for private use.

Let us consider the first output and try to understand what port is used for what purposes.

```
x.x.xx.xxx;host3.x0x;PTR;tcp;22;ssh;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;25;smtp;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;53;domain;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;80;http;open;;;syn-ack;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;137;netbios-ns;filtered;;;no-response;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;138;netbios-dgm;filtered;;;no-response;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;139;netbios-ssn;filtered;;;no-response;;3;
x.x.xx.xxx;host3.x0x;PTR;tcp;445;microsoft-ds;filtered;;;no-response;;3;
```

Port 22 is used for ‘SSH’. It stands for ‘Secure Socket Shell’. It’s a network protocol with which administrators access a remote computer in a secure way.

```
Port 25 is for SMTP or mail.
Port 53 stands for DNS Services.
Port 80 is for web traffic.
Port 137, 138 and 139 are grabbed by Microsoft for transporting their NetBIOS pr\
otocol over IP based LAN and WAN networks.
Lastly the port 445 is used for Microsoft Directory Services. For further readin\
g about this port you may find this link interesting: https://www.grc.com/port_4\
45.htm.
```

TLD Scanner

Our first step will be creating a general python file that will simply create a directory and save the 'gathered information'. You can create any python file and may name it 'nmapping.py'. The code is simple enough to follow.

```
#!/usr/bin/python3

import os

def create_dir(directory):
    if not os.path.exists(directory):
        os.makedirs(directory)

def write_file(path, data):
    f = open(path, 'w')
    f.write(data)
    f.close()
```

The first function creates a directory and inside the function it simply checks whether the directory already exists or not. And in the second function it's writing the data in a path. Here path is 'open a file'.

In the next step we'll do some real hacking with python and extract top level domain name of the websites with the help of python modules.

Assuming that you possess intermediate python knowledge and a hacking beginner I must tell you about the top level domain name.

What is top level domain name?

In any Internet system there is a hierarchy. There are top levels and lower levels. Top-level domains (TLD) are domains that stay at the highest level in the hierarchical Domain Name System of the Internet. In the name space system TLD stays in the root zone.

The last label of a fully qualified domain name is the top-level domain name. For example, in the domain name www.example.com, the top-level domain is 'example.com'.

You may ask which organizations maintain these top-level and low-level domain names. Just for the sake of your knowledge – the responsibility has been entrusted to some specific organizations by the Internet Corporation for Assigned Names and Numbers (ICANN). ICANN operates the Internet Assigned Numbers Authority (IANA), and is in charge of maintaining the DNS root zone.

In any domain name there are many parts. Consider this example: https://www.example.com/writing. This domain name starts with a protocol which is 'https' and it might have been 'http' also. Sometimes you may find 'ftp'.

The Hypertext Transfer Protocol (HTTP)/(HTTPS)is an application protocol. It is the foundation of data communication for the World Wide Web.

What does the term hypertext mean?

It is nothing but structured text that uses logical links (hyperlinks) between nodes containing text only. HTTP is the protocol to exchange or transfer hypertext.

So a domain name starts with HTTP or HTTPS (which is much safer) and next comes the 'www' which stands for 'world wide web' and then at the end we get top level domain name that is 'example.com'.

If you have any directory or sub-directory it just follows the top level domain name.

Let us start with 'whois' search for a website. This could be the beginning or the first step of our reconnaissance scanning effort. Any type of information gathering in the ethical hacking territory is incomplete without 'whois' search.

Here is a gentle reminder for all the hacking beginners. Before going for information gathering or any kind of reconnaissance scanning you should be anonymous. It's a good practice. Anonymity plays a big role in ethical hacking and I've written books on it. There are other good books also. So you'd better open up your virtual machine and anonymous Kali Linux terminal before starting this job.

Our task is building a python file that will extract the top-level domain name or TLD. In python there is a module called 'tld'. If your python is updated you will always have it. If not, then you can install it through this command:

```
sudo apt-get install python3-pip
```

If you've already installed it like me then you'll have an output like this:

```
Reading package lists... Done

Building dependency tree

Reading state information... Done

python3-pip is already the newest version.

The following packages were automatically installed and are no longer required:

  bsdtar ruby-childprocess ruby-erubis ruby-ffi ruby-i18n ruby-log4r
```

```
12
13    ruby-net-scp ruby-net-ssh
14
15  Use 'apt-get autoremove' to remove them.
16
17  0 upgraded, 0 newly installed, 0 to remove and 496 not upgraded.
```

The next command will be like this:

```
1  hagudu@hagudu-H81M-S1:~$ sudo pip3 install tld
2
3  [sudo] password for hagudu:
4
5  Requirement already satisfied (use --upgrade to upgrade): tld in /usr/local/lib/\
6  python3.4/dist-packages
7
8  Cleaning up...
```

It has already been installed so nothing comes up actually. This 'tld' module of 'python 3' is extremely important for getting top-level domain name through your python file. Next we'll write our simple python file - 'domain.py', which brings us any top-level domain name easily.

```
1  #!/usr/bin/python3
2
3  from tld import get_tld
4
5  def GetDomainName(url):
6
7      DomainName = get_tld(url)
8
9      return DomainName
10
11 print(GetDomainName('https://google.com'))
```

Here We're trying to get the top-level domain name of 'https://google.com'. And here is the output:

```
1  /usr/bin/python3.4 /home/hagudu/PycharmProjects/FirstPythonProject/Nmap/domain.py
2  google.com
3
4  Process finished with exit code 0
```

As you see, through the 'tld' module we have easily extracted the top-level domain name of any website. The python function is very simple. We've set one parameter - 'url', in the main function and passed any original website URL through it to get the actual TLD.

Get IP Address

It appears that getting top level domain name (TLD) is not very complicated and you can do that by yourself through your python code.

Now, we'll see how we can get the IP address of any web site. Apparently it also appears very easy for the guys who know a little bit about linux commands.

Just open your Ubuntu terminal and type any top level domain name with the command: host. You'll get the following output.

```
hagudu@hagudu-H81M-S1:~$ host google.com
google.com has address 216.58.203.174
google.com has IPv6 address 2404:6800:4009:807::200e
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 10 aspmx.l.google.com.
```

As we see here Google is showing its IP address as some numbers. Before we know about getting the IP address let's find out what does that actual mean?

We also keenly want to study these two lines:

```
google.com has address 216.58.203.174
google.com has IPv6 address 2404:6800:4009:807::200e
```

It says Google has one address and that is in combination of four blocks. In the next line we find another IPv6 address which is completely different.

First of all, the abbreviation IP stands for internet protocol. In the network chapter we've discussed about it. We also knew about the TCP/IP protocol. On this TCP/IP network billions of computers and other electronic devices are connected. There should be a mechanism to identify them. You can imagine those devices as single nodes.

Administrators generally set up and control the IP addressing system for their local networks (LAN). An IP address can be private – especially when you use on a local area network (LAN). It becomes public when you use it on the Internet or other wide area network (WAN).

On the other hand Internet is a vast network handling many IP addresses and those are managed by the service providers and a central allocation system. In the global context The IP addresses are

managed by the Internet Assigned Numbers Authority (IANA). There are also five regional Internet registries (RIR) and they are responsible in their selected territories and they assign IP addresses to individual users and Internet service providers.

Now we need to know the difference between IPv4 and IPv6. in our Google IP address output what you've seen in the first line is the IPv4 IP address. IPv4 addresses consist of four bytes (32 bits) and IPv6 addresses are 16 bytes (128 bits) long. You're intelligent enough to guess that IPv6 has been introduced because with IPv4 it seemed that the number of allocation of addresses would end one day. So we needed to shift to 128 bits long numbers.

You've seen that through linux command it's not difficult to get IP address of any web site. But it looks clumsy and if we want to store them in a text file or any database, it also ends up in a very cumbersome manner.

Let's build a python IP address extractor. It'll store the IP address individually without any extra lines.

To do that we'll write a code like this:

```
#!/usr/bin/python3

import os

def GetIPAddress(url):

    LinuxCommand = "host " + url
    StartProcess = os.popen(LinuxCommand)
    results = str(StartProcess.read())

    marking = results.find('has address') + 12

    return results[marking:].splitlines()[0]

print(GetIPAddress('google.com'))
```

Let's understand each line. First we have imported the 'os' module. The name is quite literary. It stands for operating system. Any operating system will have a terminal process. We're going to use that process to extract our IP address in a clear and concise way.

First we define a function through which we can pass the URL. We've passed the parameter. Next in a variable we get that command fully.

```
LinuxCommand = "host " + url
```

Actually when we write the same command on the terminal what we get? You've seen the output before. Now from that output we'd like to omit everything except the first 12 characters. That is the IPv4 address. So our next step will be easier. We'll start the process, read the process and finally we would like to stop the reading after 12 characters.

Let's run the program.

The output is like this:

```
1 /usr/bin/python3.4 /home/hagudu/PycharmProjects/FirstPythonProject/Nmap/ipaddres\
2 s.py
3 216.58.199.142
4
5 Process finished with exit code 0
```

When we used the terminal and tried to get the IP address of Google it was different. Now it's different. But you can be sure that this is the authentic IP address of the TLD or top level domain name of Google.

Our python program splits the lines and picks up the top level domain name with this line:

```
1 return results[marking:].splitlines()[0]
```

Whois Search

Who are you?

Quite a simple query. But you can interpret it in different ways. A human being has many identity marks. Name, address, contact number etc are few to name in this context. Real thing is you need to have something so that other human beings can identify you and finally can communicate with you.

Internet is a vast place. If you only think about the space it has acquired in a very short period of time, you find it just amazing. It has already proved that our future space of communication is going to be Internet. You may give it any name but finally it is nothing but internet or web space.

As time passes by, the numbers of web sites grow and with it problem of finding new addresses mature into a real crisis. Creating more digital space is no problem. The real problem is identifying them to protect us from cyber criminals. If needed, legal establishment should be able to identify any type of web owner.

“WHOIS” search enables us to do this. Like other internet protocols, it’s also a protocol, that is widely used for querying databases that store the registered web owners. Internet resource is usually assigned to various persons. Domain name, IP address block and other major information are stored in a database.

It was first established in the early 1980s. For more than twenty five years this querying protocol has been running and faced with much criticism.

You cannot compare 1980’s world of internet with the present day. In the infrastructure and administration of internet many changes have taken place. Billions of web users, diversified interests clash with each other. Technology has been changed a lot. To address the present day needs developers are working and developing a new protocol, the Internet Registry Information Service (IRIS). But “whois” search does not lose its relevance.

Consider a simple code in Python.

```
import os

def GetWhoisInfo(url):

    command = "whois " + url
    process = os.popen(command)
    results = str(process.read())

    return results
```

```
10
11 print(GetWhoisInfo('nmap.org'))
```

And consider this simple output.

```
1  {
2    "updated_date": "2016-12-15 22:16:41",
3    "status": "clientTransferProhibited https://icann.org/epp#clientTransferProhib\
4  ited",
5    "name": "Domain Hostmaster",
6    "dnssec": "unsigned",
7    "city": "Seattle",
8    "expiration_date": "DATE TIME",
9    "zipcode": "========",
10   "domain_name": "NMAP.ORG",
11   "country": "US",
12   "whois_server": null,
13   "state": "WA",
14   "registrar": "==========.",
15   "referral_url": null,
16   "address": "===============",
17   "name_servers": [
18     "NS1.LINODE.COM",
19     "NS2.LINODE.COM",
20     "NS3.LINODE.COM",
21     "NS4.LINODE.COM",
22     "NS5.LINODE.COM"
23   ],
24   "org": "==========",
25   "creation_date": "1999-01-18 05:00:00",
26   "emails": "==============="
27 }
```

We have had “whois” query result of https://www.nmap.org.

There is nothing illegal in getting ‘whois’ query. How you use this information – that is really important. If somebody notes down the expiration date and waits for the right time to register it up before the owner, then the intention is definitely beyond law. For that reason I’ve obliterated few information here – specially the addresses and email part. Although any one can search it down.

You may find it little difficult for python 3 to run ‘whois’ module. For that purpose I have also run it in my terminal with python 2.7 just to show the power of python default functions.

If you don’t have python ‘whois’ module by default you can install it easily. The next terminal output shows you how you can do that. Finally we have a ‘whois’ search in our terminal.

```
hagudu@hagudu-H81M-S1:~$ hg clone https://bitbucket.org/richardpenman/pywhois
The program 'hg' is currently not installed. You can install it by typing:
sudo apt-get install mercurial
hagudu@hagudu-H81M-S1:~$ sudo apt-get install mercurial
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  bsdtar ruby-childprocess ruby-erubis ruby-ffi ruby-i18n ruby-log4r
  ruby-net-scp ruby-net-ssh
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  mercurial-common
Suggested packages:
  qct kdiff3 kdiff3-qt kompare meld tkcvs mgdiff
The following NEW packages will be installed:
  mercurial mercurial-common
0 upgraded, 2 newly installed, 0 to remove and 496 not upgraded.
Need to get 1,561 kB of archives.
After this operation, 8,518 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu/ trusty-updates/universe mercurial-com\
mon all 2.8.2-1ubuntu1.3 [1,520 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu/ trusty-updates/universe mercurial amd\
64 2.8.2-1ubuntu1.3 [41.1 kB]
Fetched 1,561 kB in 3s (437 kB/s)
Selecting previously unselected package mercurial-common.
(Reading database ... 230139 files and directories currently installed.)
Preparing to unpack .../mercurial-common_2.8.2-1ubuntu1.3_all.deb ...
Unpacking mercurial-common (2.8.2-1ubuntu1.3) ...
Selecting previously unselected package mercurial.
Preparing to unpack .../mercurial_2.8.2-1ubuntu1.3_amd64.deb ...
Unpacking mercurial (2.8.2-1ubuntu1.3) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up mercurial-common (2.8.2-1ubuntu1.3) ...
Setting up mercurial (2.8.2-1ubuntu1.3) ...

Creating config file /etc/mercurial/hgrc.d/hgext.rc with new version
hagudu@hagudu-H81M-S1:~$ hg clone https://bitbucket.org/richardpenman/pywhois
destination directory: pywhois
requesting all changes
adding changesets
```

```
adding manifests
adding file changes
added 125 changesets with 198 changes to 34 files
updating to branch default
28 files updated, 0 files merged, 0 files removed, 0 files unresolved
hagudu@hagudu-H81M-S1:~$ pip install futures
Requirement already satisfied (use --upgrade to upgrade): futures in /usr/local/\
lib/python2.7/dist-packages
Cleaning up...
hagudu@hagudu-H81M-S1:~$ python2
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import whois
(6, 40)
{'uphosts': '1', 'timestr': 'Sat Mar 11 09:41:28 2017', 'downhosts': '0', 'total\
hosts': '1', 'elapsed': '5.67'}
up
['tcp']
[80, 25, 53]
>>> w = whois.whois('webscraping.com')
>>> w.expiration_date
[datetime.datetime(2020, 6, 26, 0, 0), datetime.datetime(2020, 6, 26, 18, 1, 19)]
>>> w = whois.whois('nmap.org')
>>> w
{u'updated_date': datetime.datetime(2016, 12, 15, 22, 16, 41), u'status': u'clie\
ntTransferProhibited https://icann.org/epp#clientTransferProhibited', u'name': u\
'Domain Hostmaster', u'dnssec': u'unsigned', u'city': u'Seattle', u'expiration_d\
ate': datetime.datetime(), u'zipcode': u'', u'domain_name': u'NMAP.ORG', u'count\
ry': u'US', u'whois_server': None, u'state': u'WA', u'registrar': u.', u'referra\
l_url': None, u'address': u'', u'name_servers': [u'NS1.LINODE.COM', u'NS2.LINODE\
.COM', u'NS3.LINODE.COM', u'NS4.LINODE.COM', u'NS5.LINODE.COM'], u'org': u'Insec\
ure.Com LLC', u'creation_date': datetime.datetime(1999, 1, 18, 5, 0), u'emails':\
 u''}
>>>
```

I have deleted few places in the output for security purpose. But I repeat reading this information is not illegal. If you try to use it in a malicious manner, that is illegal. In the terminal output you've probably noticed that I have downloaded few packages to run this 'whois' module perfectly.

Now we can have more functions in Python 2.7. See the next output where we read the same 'whois' information in a text mode.

```
    >>> w.text
                u"Domain Name: NMAP.ORG\r\nRegistry Domain ID: D3106402-LROR\r\n\
Registrar WHOIS Server:\r\nRegistrar URL: http://www.fabulous.com\r\nUpdated Dat\
e: 2016-12-15T22:16:41Z\r\nCreation Date: 1999-01-18T05:00:00Z\r\nRegistry Expir\
y Date: 2023-01-18T05:00:00Z\r\nRegistrar: Fabulous.com Pty Ltd.\r\nRegistrar IA\
NA ID: 411\r\nRegistrar Abuse Contact Email:\r\nRegistrar Abuse Contact Phone:\r\
\nDomain Status: clientTransferProhibited https://icann.org/epp#clientTransferPr\
ohibited\r\nRegistry Registrant ID: ==========\r\nRegistrant Name: Domain Hostma\
ster\r\nRegistrant Organization: Insecure.Com LLC\r\nRegistrant Street: ========\
\r\nRegistrant City: =========\r\nRegistrant State/Province: WA\r\nRegistrant Po\
stal Code: =========\r\nRegistrant Country: US\r\nRegistrant Phone: ========\r\n\
Registrant Phone Ext:\r\nRegistrant Fax:\r\nRegistrant Fax Ext:\r\nRegistrant Em\
ail: ============\r\nRegistry Admin ID: ==========\r\nAdmin Name: Domain Hostmas\
ter\r\nAdmin Organization: Insecure.Com LLC\r\nAdmin Street: ==========\r\nAdmin\
 City: =====\r\nAdmin State/Province: WA\r\nAdmin Postal Code: --------\r\nAdmin\
 Country: US\r\nAdmin Phone: +1.4159159337\r\nAdmin Phone Ext:\r\nAdmin Fax:\r\n\
Admin Fax Ext:\r\nAdmin Email: =========\r\nRegistry Tech ID: 9447a1ab61e1d3e9\r\
\nTech Name: Domain Hostmaster\r\nTech Organization: Insecure.Com LLC\r\nTech St\
reet: ==========\r\nTech City: ========\r\nTech State/Province: WA\r\nTech Posta\
l Code: 98104-2205\r\nTech Country: US\r\nTech Phone: =========\r\nTech Phone Ex\
t:\r\nTech Fax:\r\nTech Fax Ext:\r\nTech Email: ========= \r\nName Server: NS1.L\
INODE.COM\r\nName Server: NS2.LINODE.COM\r\nName Server: NS3.LINODE.COM\r\nName \
Server: NS4.LINODE.COM\r\nName Server: NS5.LINODE.COM\r\nDNSSEC: unsigned\r\nURL\
 of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/\r\n>>\
> Last update of WHOIS database: 2017-03-11T04:14:08Z <<<\r\n\r\nFor more inform\
ation on Whois status codes, please visit https://icann.org/epp\r\n\r\nAccess to\
 Public Interest Registry WHOIS information is provided to assist persons in det\
ermining the contents of a domain name registration record in the Public Interes\
t Registry registry database. The data in this record is provided by Public Inte\
rest Registry for informational purposes only, and Public Interest Registry does\
 not guarantee its accuracy. This service is intended only for query-based acces\
s. You agree that you will use this data only for lawful purposes and that, unde\
r no circumstances will you use this data to: (a) allow, enable, or otherwise su\
pport the transmission by e-mail, telephone, or facsimile of mass unsolicited, c\
ommercial advertising or solicitations to entities other than the data recipient\
's own existing customers; or (b) enable high volume, automated, electronic proc\
esses that send queries or data to the systems of Registry Operator, a Registrar\
, or Afilias except as reasonably necessary to register domain names or modify e\
xisting registrations. All rights reserved. Public Interest Registry reserves th\
e right to modify these terms at any time. By submitting this query, you agree t\
o abide by this policy.\r\n"
```

This is more powerful search and has come up with more information including the warning in the

last part.

It also shows the update date as: Last update of WHOIS database: 2017-03-11T04:14:08Z.

In our last search we've got the last update date as: "updated_date": "2016-12-15 22:16:41".

And the last part, that is also very important to read as far as legality of any scanning and information gathering stands for. You need to be aware that you should always stay within law.

"WHOIS" scanning is extremely important but you cannot expect that accuracy will be guaranteed. Sometimes the wholesale registrar comes in between you and the web owner.

It may happen that the main registrar goes out of business. To confirm your 'whois' search's accuracy it's mandatory to check that the resource is at least registered with ICANN.

Privacy is a big concern. And for that reason The Expert Working Group (EWG) of the Internet Corporation for Assigned Names and Numbers (ICANN) has recommended that WHOIS should be scrapped. It recommends a system where information would be kept secret from most Internet users. They opine for disclosure of information that are to the degree of "permissible purposes."

NMAP Port Scan

Have you seen the film “The Matrix Reloaded”? Well, if you had seen you would have probably recalled the scene where the character Trinity was seen using NMAP to hack the system of a power plant.

A film is a film. We’re in reality. Life is not a film!

NMAP port scanning is an essential tool for ethical hackers and computer security people. The creator of NMAP is Gordon Lyon, and in his about page he writes: ‘… I often go by Fyodor on the Internet. I run the Internet security resource sites Insecure.Org, Nmap.Org, SecLists.Org, and SecTools.Org. I also wrote and maintain the Nmap Security Scanner.’

Lyon is a celebrated ethical hacker and he writes fine prose also. You may be interested to read his chapter of “Stealing the Network: How to Own a Continent” here: http://insecure.org/stc/. What NMAP does?

It basically scans a system and many more. It gives us reports on the services running on a system. It discovers hosts and ports and it also detects operating system running on a system. It tells us about the open and filtered ports. In a nutshell, it provides a lot of important information.

This is not an end to it. NMAP is a vast topic and if you visit their web site and you must do it as a student of ethical hacking; you’ll find that these features can be extended to more advanced security scanning. It includes vulnerability tests, advanced service detections.

Before going further on this topic let’s do a simple NMAP scanning on an IP address. How about doing it on Wikipedia?

Now finding IP address is a very simple job. You have your python code at your hand. Well, we’ve run it and found that it shows: 91.198.174.192.

Open your Ubuntu terminal and type:

```
1  nmap -F 91.198.174.192
```

It gives us a not very long output.

```
Starting Nmap 6.40 ( http://nmap.org ) at 2017-03-09 09:56 IST
Nmap scan report for text-lb.esams.wikimedia.org (91.198.174.192)
Host is up (0.18s latency).
Not shown: 87 closed ports
PORT     STATE    SERVICE
22/tcp   filtered ssh
25/tcp   open     smtp
53/tcp   open     domain
80/tcp   open     http
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
179/tcp  filtered bgp
443/tcp  open     https
445/tcp  filtered microsoft-ds
5060/tcp filtered sip
5666/tcp filtered nrpe
8008/tcp open     http
9100/tcp open     jetdirect

Nmap done: 1 IP address (1 host up) scanned in 3.12 seconds
```

We get a lot of information as it appears which ports are open and which type of operating system they are running. Come to the lines. It clearly tells us how many ports are closed. It also says which port is open. It says about the Microsoft added transport protocol and many more things. Look at the last line: '9100/tcp open jetdirect'.

TCP Port 9100 is commonly used by printer manufacturers. What is Jetdirect? HP Jetdirect is the name of a technology sold by Hewlett-Packard that allows computer printers to be directly attached to a Local Area Network. You're gathering more and more information. From one point of data you can converge on many points of data.

Since there are HTTP/HTTPS, you can guess that database is running on the system. Bunch of good information come along this way.

Now we'd like to NMAP port scan on its web site also.

The same procedure will be followed. Know the IP address first and run the NMAP command on your Linux terminal.

It gives us output like this:

```
hagudu@hagudu-H81M-S1:~$ nmap -F 45.33.49.119

Starting Nmap 6.40 ( http://nmap.org ) at 2017-03-09 09:58 IST
Nmap scan report for ack.nmap.org (45.33.49.119)
Host is up (0.00095s latency).
Not shown: 97 filtered ports
PORT   STATE SERVICE
25/tcp open  smtp
53/tcp open  domain
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 3.29 seconds
```

It says 97 filtered ports are not shown. Port 80 and SMTP port are also open.

Now we are going to create our python library on NMAP scripting.

We need the module 'os' to do that. And the script will look like this:

```
#!/usr/bin/python3

import os

def GetNMAP(options, ip):

    command = "nmap " + options + " " + ip

    process = os.popen(command)

    results = str(process.read())

    return results


print(GetNMAP('-F', '54.186.250.79'))
```

As you see, python has its own library of functions and modules to do the job. Just pass the command and you'll get the same result. This time we scan another server of Amazon and the result is like this:

```
Nmap scan report for ec2-54-186-250-79.us-west-2.compute.amazonaws.com (54.186.2\
50.79)
Host is up (0.023s latency).
Not shown: 96 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 4.14 seconds
```

Now we'd like to summarize our learning at this point. First of all there are few territories where NMAP port scan is not legal. You better get the permission to do it in such cases.

Otherwise, there are lots of features that include host discovery. Through NMAP we can identify hosts on a network. We could list them as they can respond to the TCP/IP requests. We also know which ports are open.

At the same time we can scan the port, detect the versions. We can detect the operating system and hardware characteristics of the network devices.

In the advanced level we can interact with the target by using NMAP scripting engine and LUA programming language. LUA means Moon in Portuguese. If you are interested please visit their site: https://www.lua.org.

Robots Exclusion

Checking all information about a web portal is one of the main activities that usually ethical hackers do. In a broad term reconnaissance stands for gathering information from every possible source.

The activity is not limited to only port scanning or email tracking or assembling target's addresses. Your target must have some private directories which he or she wants to keep outside of search engine searches.

To do that webmasters usually keep one 'robots.txt' file inside the web root. This is called robots exclusion standard. It is also known as the robots exclusion protocol or simply robots.txt.

In a normal course search engines crawl any web site to index and place them in their respective search listings. Keeping a 'robots.txt' file in the web root is a standard used by all websites to communicate with web crawlers and other web robots. Through this standard a website generally informs the web robot about the restricted areas. They tell the robots, "Hey, don't crawl this part of my site. I don't want to be crawled in this section." Listening to this request the web robot doesn't crawl and process that part.

Why web robots are used?

They are used for categorizing the web sites. Reading up to this you might have thought that all robots are friendly. They're indexing a web site with good intentions.

No! That is a misconception. There are good robots and bad robots.

Not all robots cooperate with the set standard. There are email harvesters, spambot, malware, and other robots that scan for security vulnerabilities in a web portal. These robots usually try to stay in the restricted zones specifically. Instead of staying out they keep hanging around in hope of gain or benefit.

How this 'robots.txt' file works?

Suppose you're a site owner and you want web robots not to crawl the "private" directory. To do that you need to place a text file 'robots.txt' with instructions in the root directory. It's something like this: http://www.example.com/robots.txt.

When a web robot comes to your site it usually searches for the 'robots.txt' file. Then it reads the instructions doing the needful. If it's a good robot then it'll never crawl the directory "private". If the 'robots.txt' file doesn't exist, the web robots travel the entire site indexing each page, categorizing them accordingly. For the sub-domains you need to maintain separate 'robots.txt' files like this – http://www.sanjib.example.com/robots.txt.

As an ethical hacker you need to know the restricted parts of a web site. How we can do that? Python has the answer. Let's write the code first and then we'll dissect the meaning.

```
#!/usr/bin/python3
import urllib.request
import io

def GetRobots(url):

    if url.endswith("/"):

        path = url

    else:

        path = url + "/"

    requestingData = urllib.request.urlopen(path + "robots.txt", data=None)

    data = io.TextIOWrapper(requestingData, encoding="utf 8")

    return data.read()

print(GetRobots("http://www.arshinagar.in"))
```

As you see, we've imported two modules. With the help of module functions we've built a simple function that first checks whether we want to run after a sub-domain or not. After that it reads the data and we get a print output. For instance to begin with we've checked one simple site that uses wordpress content management system.

The output says that wordpress wants to be crawled everywhere except the Administration part. Here is the output:

```
/usr/bin/python3.4 /home/hagudu/PycharmProjects/FirstPythonProject/Nmap/robots.py
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
Process finished with exit code 0
```

Let's change the last line of our get robots function and instead of a simple site let's put a comparatively heavy site like https://www.reddit.com.

Change the last line of code to:

```
print(GetRobots("https://www.reddit.com"))
```

Now look at the output. It's changed drastically.

```
/usr/bin/python3.4 /home/hagudu/PycharmProjects/FirstPythonProject/Nmap/robots.py

# 80legs
User-agent: 008
Disallow: /

# 80legs' new crawler
User-agent: voltron
Disallow: /

User-Agent: bender
Disallow: /my_shiny_metal_ass

User-Agent: Gort
Disallow: /earth

User-agent: MJ12bot
Disallow: /

User-agent: PiplBot
Disallow: /

User-Agent: *
Disallow: /*.json
Disallow: /*.json-compact
Disallow: /*.json-html
Disallow: /*.xml
Disallow: /*.rss
Disallow: /*.i
Disallow: /*.embed
Disallow: /*/comments/*?*sort=
Disallow: /r/*/comments/*/*/c*
Disallow: /comments/*/*/c*
Disallow: /r/*/submit
Disallow: /message/compose*
Disallow: /api
Disallow: /post
Disallow: /submit
Disallow: /goto
Disallow: /*after=
Disallow: /*before=
Disallow: /domain/*t=
```

```
Disallow: /login
Disallow: /reddits/search
Disallow: /search
Disallow: /r/*/search
Disallow: /r/*/user/
Disallow: /gold?
Allow: /partner_api/
Allow: /
Allow: /sitemaps/*.xml

Sitemap: https://www.reddit.com/sitemaps/subreddit-sitemaps.xml
Sitemap: https://www.reddit.com/sitemaps/comment-page-sitemaps.xml

Process finished with exit code 0
```

The difference between these two outputs clearly shows how tough it is to scan a heavy site like https://www.reddit.com.

You may be astonished with this type of line:

```
User-Agent: Gort
Disallow: /earth
```

Reddit wants user agent robot ‘Gort’ not to crawl the ‘earth’ directory. The web site owner or the web master in its robots exclusion protocol uses the term “User Agent”. When he writes “User-agent: *” in the ‘robots.txt’ file, it means all robots can visit to all these sections. The “Disallow: /private” tells the robot that it should not visit any pages on the private directory.

Just for more information, ‘Gort’ stands for the interstellar police. It’s really humorous when we find that https://www.reddit.com directs the user agent ‘Gort’ not to crawl the directory ‘earth’. On earth we have our policing system. Why should the interstellar police come here?

Prologue

In this book python programming language has played a major role and you see it in almost every chapter. Readers with intermediate or advanced python knowledge will find it simple to grasp the core idea of this book. If you're a beginner I'd suggest you to acquire preliminary python knowledge first.

A good programming knowledge always helps you gain the necessary confidence for being an ethical hacker. Moreover you have the correct mind set which is also very important. Understanding the flow of logic in computer science is particularly essential. Besides, it builds up your skill of thinking logically. The logical steps in any programming language help us think reasonably in real life. In ethical hacking it's imperative.

Ethical hacking is a very vast topic and it includes many things that you may not cover altogether at once. You need not worry about it. Few basic things like Linux administration and programming and knowledge of Python programming language will always help you; so try to master them slowly. Don't lose your heart at the very beginning.

I again repeat, ethical hacking has many branches and you'll definitely find your own point of interest one day and if you approach the subject with ultimate passion you'll learn it one day.

Keep learning and enjoy the process, share your knowledge and help your fellow learners.

For any query, you feel free to contact me at sanjib12sinha@gmail.com and my twitter handle is @sanjibsinha. You may send a direct message. I also need your assistance in finding errors.

I'll love to hear from you. Best of luck.