COURSE GUIDE

CYB 224
PYTHON FOR CYBER SECURITY

Course Team M. A. Bagiwa (PhD) (Course Writer)
Prof. Joe Essien (Content Editor)

NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
University Village
Plot 91, Cadastral Zone
Nnamdi Azikiwe Expressway
Jabi, Abuja

Lagos Office
14/16 Ahmadu Bello Way
Victoria Island, Lagos

E-mail : centralinfo@nou.edu.ng
URL: www.nou.edu.ng

Printed 2024

ISBN: 978-978-786-230-8

CONTENTS **PAGE**

INTRODUCTION

Welcome **to CYB224 – Python for Cyber security.** CYB224 – Python for Cyber security is a 2-credit unit. The course is a core course in second semester. It will take you 13 weeks to complete the course. You are to spend 65 hours of study for a period of 13 weeks while the first week is for orientation and the last week is for end of semester examination. The credit earned in this course is part of the requirement for graduation.
You will receive the course material which you can read online or download and read off-line. The online course material is integrated in the Learning Management System (LMS). All activities in this course will be held in the LMS. All you need to know in this course is presented in the following sub-headings.

COURSE COMPETENCIES

- Using Python libraries such as Scapy, Nmap, and Requests for network and web security assessments.
- Automating log parsing and analysis for intrusion detection.

COURSE OBJECTIVES

- To develop scripts to automate repetitive security tasks, such as log analysis and vulnerability scanning.
- To use Python to capture, decode, and analyze network packets to detect anomalies and potential threats.
- To use Python libraries to implement and understand encryption, decryption, hashing, and digital signatures
- To create Python scripts to assist in the rapid response and investigation of security incidents.

WORKING THROUGH THIS COURSE

The course is divided into modules and units. The modules are derived from the course competencies and objectives. The competencies will guide you on the skills you will gain at the end of this course. So, as you work through the course, reflect on the competencies to ensure mastery.

Each study unit has introduction, intended learning outcomes, the main content, conclusion, summary and references/further readings. The introduction will tell you the expectations in the study unit. Read and note the intended learning outcomes (ILOs). The intended learning outcomes tell you what you should be able to do at the completion of each study unit. So, you can evaluate your learning at the end of each

unit to ensure you have achieved the intended learning outcomes. To meet the intended learning outcomes, knowledge is presented in texts, video and links arranged into modules and units. Click on the links as may be directed but where you are reading the text off line, you will have to copy and paste the link address into a browser. You can download the audios and videos to view offline. You can also print or download the texts and save in your computer or external drive. The conclusion gives you the theme of the knowledge you are taking away from the unit. Unit summaries are presented in downloadable audios and videos.

There are two main forms of assessments – the formative and the summative. The formative assessments will help you monitor your learning. This is presented as in-text questions, discussion forums and Self-Assessment Exercises.

The summative assessments would be used by the university to evaluate your academic performance. This will be given as Computer Base Test (CBT) which serve as continuous assessment and final examinations. A minimum of three computer base test will be given with only one final examination at the end of the semester. You are required to take all the computer base tests and the final examination.

The main content is the body of knowledge in the unit. Self-assessment exercises are embedded in the content which helps you to evaluate your mastery of the competencies. The conclusion gives you the takeaway while the summary is a brief of the knowledge presented in the unit. The final part is the further readings. This takes you to where you can read more on the knowledge or topic presented in the unit. The modules and units are presented as follows:

There are eleven units in this course. Each unit represents a week of study. The modules and units are presented as follows:

STUDY UNITS

Module 1 Introduction to Python and Cyber Security
Unit 1 Overview of Python Programming Language
Unit 2 Installing Python and setting up the environment
Unit 3 Basic Python Syntax
Unit 4 Overview of Strings in Python

Module 2 Working with Networks and Sockets
Unit 1 Introduction to Networking Concepts
Unit 2 Socket Programming with Python

REFERENCES AND FURTHER READINGS

Al Sweigart (2015). *Automate the Boring Stuff with Python: Practical Programming for Total Beginners*. No Starch Press.

Provides a foundational understanding of Python programming with practical examples.

Justin Seitz (2014). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

Focuses on Python's application in penetration testing and cybersecurity tasks.

TJ O'Connor (2012). *Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers, and Security Engineers*. Syngress.

Offers a variety of practical Python scripts for security professionals.

Zaid Sabih (2018). *Learn Python & Ethical Hacking From Scratch*. Packt Publishing.

Combines Python programming with ethical hacking techniques.

Sybex (2018). *Python for Cybersecurity: Using Python for Cyber Offense and Defense*. Sybex.

Details Python's role in both offensive and defensive cybersecurity operations.

Gus Khawaja (2018). *Practical Python and OpenCV: An Introductory, Example-Driven Guide to Image Processing and Computer Vision*. PyImageSearch.

Discusses the use of Python in image processing and computer vision, relevant for security applications like facial recognition.

Christopher Duffy (2020). *Python Penetration Testing Essentials*. Packt Publishing.

Introduces penetration testing concepts and tools using Python.
Sanjib Sinha (2019). *Beginning Ethical Hacking with Python*. Apress.

Covers ethical hacking fundamentals and practical Python applications.

Brett Slatkin (2015). *Effective Python: 59 Specific Ways to Write Better Python*. Addison-Wesley Professional.

Offers insights on writing more efficient and effective Python code, useful for security scripting.

Khrais, H. (2018). Python for Offensive PenTest: A practical guide to ethical hacking and penetration testing using Python. Packt Publishing Ltd.

Offers a practical guidance on using Python for offensive security, including creating your own tools and scripts for penetration testing.

ONLINE COURSES AND TUTORIALS

Coursera: https://www.coursera.org/
Udemy: https://www.udemy.com/
Pluralsight: https://www.pluralsight.com/
Cybrary: https://www.cybrary.it/
GitHub - Python Security
Google Scholar - Python for Cybersecurity Research Papers
LinkedIn Learning - Learning Python for Cybersecurity
Coursera - Python for Cybersecurity Specialization
SecurityTube - Python for Security
Python.org - Python Documentation

PRESENTATION SCHEDULE

The weekly activities are presented in Table 1 while the required hours of study and the activities are presented in Table 2. This will guide your study time. You may spend more time in completing each module or unit.

Table 1: Weekly Activities

Week	Activity
1	Orientation and course guide

2	Module1Unit1
3	Module1Unit2
4	Module1Unit3
5	Module1Unit4
6	Module2Unit1
7	Module2Unit2
8	Module3Unit1
9	Module3Unit2
10	Module4Unit1
11	Module4Unit2
12	Module4Units3
13	Revision and response to questionnaire
14	Revision and response to questionnaire
15	Examination

The activities in Table I include facilitation hours (synchronous and asynchronous), assignments, mini projects, and laboratory practical. How do you know the hours to spend on each? A guide is presented in Table 2.

Table2: Required Minimum Hours ofStudy

S/N	Activity	Hour per Week	Hour per Semester
1	Synchronous Facilitation (Video Conferencing)	2	26
2	Asynchronous Facilitation (Read and respond to posts including facilitator's comment, self-study)	4	52
3	Assignments, mini-project, laboratory practical and portfolios	1	13
	Total	7	91

ASSESSMENT

There are four main forms of assessments in this course that will be scored. These are Continuous Assessments and the final examination. The continuous assessment shall be in three-fold as indicated in Table 3. **The computer-based assessments will be given in accordance to university academic calendar. The timing must be strictly adhered to**. The Computer Based Assessments shall be scored a maximum of 10% each, while your participation in miniprojects presentation and your portfolio presentation shall be scored maximum of 40% if you meet

75% participation. Therefore, the maximum score for continuous assessment shall be 20% which shall form part of the final grade.

Table3presentsthemodeyouwillbeassessed. Table 3: Assessment

S/N	Method of Assessment	Score(%)
1	Portfolios	10
2	Mini Projects with presentation	30
3	Assignments	20
4	Final Examination	40
Total		100

PORTFOLIO

A portfolio has been created for you tagged**—My Portfolio**. With the use of Microsoft Word, state the knowledge you gained in every Module and in not more than three sentences explain how you were able to apply the knowledge to solve problems or challenges in your context or how you intend to apply the knowledge. Use this Table format:

Application of Knowledge Gained

Module	Topic	Knowledge Gained	Application of Knowledge Gained

You may be required to present your portfolio to a constituted panel.

MINI PROJECTS WITH PRESENTATION

You are to work on the project according to specification. You may be required to defend your project. You will receive feedback on your project defense or after scoring. This project is different from your thesis.

ASSIGNMENTS

Take the assignment and click on the submission button to submit. The assignment will be scored, and you will receive feedback.

EXAMINATION

Finally, the examination will help to test the cognitive domain. The test items will be mostly application, and evaluation test items that will lead to creation of new knowledge/idea.

HOW TO GET THE MOST FROM THE COURSE

To get the most in this course, you need to have a personal laptop and regular and stable internet. The use of mobile phone only may not give you the desirable environment to work. This will give you adequate opportunity to learn anywhere you are in the world. Use the Intended Learning Outcomes (ILOs) to guide your self-study in the course. At the end of every unit, examine yourself with the ILOs and see if you have achieved what you need to achieve.

To succeed in this course, you must install the recommended software and diligently work through the course step by step, starting with the program orientation. Plagiarism or impersonation are serious offences that could lead to the termination of your studentship, and all submissions will be checked for plagiarism. It is essential to complete all assessments according to the given instructions and to dedicate time daily to your studies.

Carefully work through each unit and make your notes. Join the online real time facilitation as scheduled. Where you missed the scheduled online real time facilitation, go through the recorded facilitation session at your own free time. Each real time facilitation session will be video recorded and posted on the platform.

In addition to the real time facilitation, watch the video and audio recorded summary in each unit. The video/audio summaries are directed to salient part in each unit. You can assess the audio and videos by clicking on the links in the text or through the course page.

Work through all self-assessment exercises. Finally, obey the rules in the class.

FACILITATION

There will be two forms of facilitation – synchronous and asynchronous. The synchronous will be held through video conferencing according to weekly schedule. **During the synchronous facilitation**:

- There will be two hours of online real time contact per week making a total of 26 hours for thirteen weeks of study time.
- At the end of each video conferencing, the video will be uploaded for view at your pace.
- You are to read the course material and do other assignments as may be given before video conferencing time.
- The facilitator will concentrate on main themes.
- The facilitator will take you through the course guide in the first lecture at the start date of facilitation.

For the asynchronous facilitation, your facilitator will:

- Present the theme for the week.
- Direct and summarise forum discussions.
- Coordinate activities in the platform.
- Score and grade activities when need be.
- Support you to learn. In this regard personal mails may be sent.
- Send you videos and audio lectures, and podcasts if need be.

Read all the comments and notes of your facilitator especially on your assignments, participate in forum discussions. This will give you opportunity to socialize with others in the course and build your skill for teamwork. You can raise any challenge encountered during your study. To gain the maximum benefit from course facilitation, prepare a list of questions before the synchronous session. You will learn a lot from participating actively in the discussions.

Finally, respond to the questionnaire. This will help ACETEL to know your areas of challenges and how to improve on them for the review of the course materials and lectures.

LEARNER SUPPORT

You will receive the following support:

- Technical Support: There will be contact number(s), email address and chatbot on the Learning Management System where you can chat or send message to get assistance and guidance any time during the course.
- 24/7 communication: You can send personal mail to your facilitator and the centre at any time of the day. You will receive answer to you mails within 24 hours. There is also opportunity for personal or group chats at any time of the day with those that are online.
- You will receive guidance and feedback on your assessments, academic progress, and receive help to resolve challenges facing your studies.

ICE BREAKER

You are welcome to CYB 224, Python for Cyber Security, a two-unit course. Please upload your profile such as picture, workplace address, GSM number and other details on your wall. What are your expectations in this course? How do you see Python playing a role in your cyber security journey? Additionally, what specific skills or topics are you most excited to explore in this course?

MAIN COURSE

CONTENTS

MODULE1 INTRODUCTION TO PYTHON AND CYBER SECURITY

UNIT 1 OVERVIEW OF PYTHON PROGRAMMING LANGUAGE

Unit Structure

1.0 Introduction

Python programming language is well-known for being readable and simple to use. It is a general-purpose language that can be used for a variety of applications, including web development, data analysis, machine learning and Cyber security. Cyber security is the discipline of defending programs, networks, and systems from online threats. These cyber-attacks typically target customers with the intent of extorting money or accessing, altering, or destroying sensitive data, as well as disrupting regular corporate operations. The history of Python and its importance to cyber security is the focus of this unit.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- explain the history of Python Programming Language
- explain what is cyber security
- explain the usage and application of python in cyber security.

3.0 Main Content

3.1 History of Python

Python is very famous for its simple programming syntax, code readability and English like commands making its usage for coding easier and efficient. As a result of this, python has become the most popular and preferred language for coding. Python is an interpreted high-level programming language developed by Guido van Rossum at the National Research Institute for computer science in the Netherlands. Python was named after a show "Monty pythons flying circus". Python was invented by its creator in the late 1980s as a successor to the ABC language. In 1994 version 1.0 of Python was released with features such as exception handling and Lambda. Version 2.0 was released in the year 2000 with more features like list comprehension and garbage collection system. Python 3.0was released on the 3rd December 2008. Python 2.x and 3.x are the most used versions as of today. Python 3.7.4 is the latest stable version released in December 2019. Python is mostly used for development, scripting and software testing. Top IT companies like Google, Facebook, Instagram, and Spotify among others Python. Major applications of the use of Python can be seen in machine learning artificial intelligence data Science and IoT. Python also offer many libraries such as tensor flow and Django.

3.2 What is Cyber Security?

Cyber security is the act of protecting our network and devices from unauthorized access. Unauthorized access can refer to small or big cyber-attacks and cyber threats. There are various types of cyber-attacks that one can fall prey to such as Phishing, malware attacks, password attacks and many more. A few ways to implement cyber security is to define clear boundaries using network security control devices like firewalls, intrusion detection systems and carrying out security testing using libraries provided by programming languages such as python. At any point in time CIA which is confidentiality, integrity and availability are implemented in an organisation to make sure that user information are secured. To learn more on what is cyber security, you can visit (https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security)

In-text Question
What are some common types of cyber-attacks mentioned in the text?

Answer
The common types of cyber-attacks mentioned are phishing, malware attacks, and password attacks.

3.3 Why Python for Cyber security?

Python programming language is now widely used in cyber security because of several reasons which include:

- **Versatility:** The unique versatility of python in terms of flexible libraries and framework for numerous cyber security tasks such as network and port scan, penetration testing, malware analysis and cryptography makes it a good choice for cyber security professionals.
- **Ease of Learning and Usage:** The simple syntax and readability of python has made it more accessible to beginners and developers.
- **Rapid Development**: Python extensive libraries and packages developed by third party enhanced its usage for rapid development of cyber security tools and scripts.
- **Support by Communities:** Python has large and active community developers which include many cyber security professionals.
- **Cross-Platform Compatibility**: Python can be used across platform which means a python code can execute on different operating systems without modification.

In summary the versatility, simplicity, rapid development capabilities, community support and cross-platform capability make python a very good choice for many cyber security professionals and organisations.

4.0 Self-Assessment Exercise(S)

1. What are some specific ways in which Python can be used to enhance network security and protect against cyber-attacks like phishing and malware?

Answer:

Python enhances network security by utilizing libraries like Scapy for packet analysis and detection of anomalies, and cryptography modules for robust encryption to safeguard against phishing and malware attacks. It also supports automation of security processes, enabling rapid response and mitigation of potential threats in real-time.

2. Who invented python and where was it invented?

Answer:

Python is an interpreted high-level programming language developed by Guido van Rossum at the National Research Institute for computer science in the Netherlands.

3. State and explain why python programming language is a good choice for cyber security professionals.

Answer:

Python programming language is now widely used in cyber security because of several reasons which include:

- **Versatility:** The unique versatility of python in terms of flexible libraries and framework for numerous cyber security tasks such as network and port scan, penetration testing, malware analysis and cryptography makes it a good choice for cyber security professionals.
- **Ease of Learning and Usage:** The simple syntax and readability of python has made it more accessible to beginners and developers.
- **Rapid Development**: Python extensive libraries and packages developed by third party enhanced its usage for rapid development of cyber security tools and scripts.
- **Support by Communities:** Python has large and active community developers which include many cyber security professionals.
- **Cross-Platform Compatibility**: Python can be used across platform which means a python code can execute on different operating systems without modification.
- **Topic for Discussion**: Introduction to Cybersecurity Fundamentals and Python Applications

5.0 Conclusion

In conclusion, Python stands out as a versatile and powerful programming language renowned for its simplicity and readability, making it an ideal choice for beginners and experienced developers alike. Its extensive standard library and rich ecosystem of third-party

packages support a wide array of applications, including data analysis, web development, automation, and cybersecurity. Python's dynamic typing and interpreted nature enhance development speed, while its object-oriented and functional programming paradigms offer flexibility in coding approaches. Emphasizing clean syntax and readability, Python fosters efficient collaboration and maintenance of codebases. As Python continues to evolve with regular updates and community contributions, its adaptability and broad applicability ensure its relevance across various domains, including cybersecurity, reinforcing its status as a cornerstone in modern programming languages.

This unit has introduced you the history and evolution of Python programming language, the definition and concept of cyber security and why Python may be an ideal choice for cyber security professionals when developing tools and scripts for network and port scan, penetration testing, malware analysis and cryptography.

6.0 Summary

In this unit, you have learnt that python is very famous for its simple programming syntax, code readability and English like commands making its usage for coding easier and efficient. You have also learnt the definition of cyber security and why may be an ideal choice for cyber security professionals

7.0 References/Further Reading

https://www.institutedata.com/blog/how-to-use-python-for-cyber-security/

Python for Cyber security: Using Python for Cyber Offense and Defense (2002) 1st Edition by Howard E. Poston III

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.

https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security

UNIT 2 INSTALLING PYTHON AND SETTING UP THE ENVIRONMENT

Unit Structure

1.0 Introduction

In the previous unit, you have learnt that python is very famous for its simple programming syntax, code readability and English like commands making its usage for coding easier and efficient. This unit will take you through the installation and the setup of python environment for your projects. Setting up Python and configuring the development environment are crucial steps for anyone embarking on programming or software development. Installing Python involves downloading the latest version from the official Python website and following straightforward installation prompts tailored to different operating systems like Windows, macOS, or Linux. Once Python is installed, setting up the environment typically includes configuring a text editor or an Integrated Development Environment (IDE) like PyCharm or VS Code, which streamline coding with features such as syntax highlighting, debugging tools, and integrated terminal access. This initial setup not only ensures that Python is ready for use but also lays the foundation for efficient and productive programming, whether for web development, data analysis, or cybersecurity applications.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- explain the usage of Python compiler
- install PyCharm IDE
- setup Python Environment in PyCharm IDE

3.0 Main Content

3.1 Python Interpreter

Python programming language is well-known for its simplicity, readability and ease of use. Python is an interpreted language. However, Python can also be compiled as well. The difference between interpretation and compilation of python is on how the code is executed and translated into machine code. Interpreted languages are executed line after line by an interpreter without the need for an intermediate compilation Step while compiled languages translate the entire code into machine instruction before execution making execution faster. An interpreter is a software program that executes an instruction written in high level language directly without converting it into machine code. Figure 1 one is a diagram showing the interpretation process of python programming language.

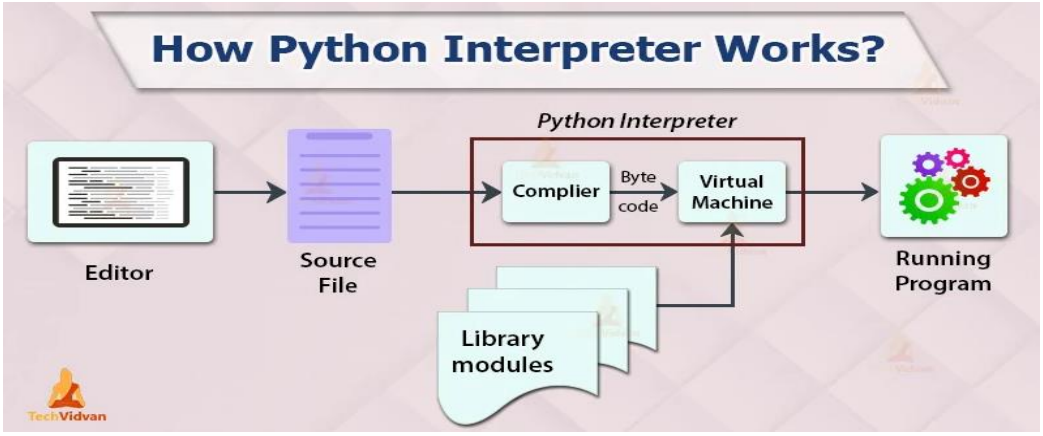


Fig. 1: How Python Interpreter Works

Python interpreter behaves as a virtual machine and execute as a stock machine it executes a number of stock operations to perform a particular task.

In-Text Question

What is the main difference between interpretation and compilation of Python code?

Answer

The main difference is that interpreted Python code is executed line by line by an interpreter without the need for an intermediate compilation step, while compiled Python code is translated into machine instructions before execution, making the execution faster.

3.2 PyCharm IDE installation

PyCharm is a cross-platform IDE that provides consistent experience on the Windows, macOS, and Linux operating systems. There are two versions of PyCharm available: Professional and Community. With fewer features, the Community edition is a free, open-source project. A superb collection of tools and features are offered by the Professional edition, which is available for purchase. Python versions that PyCharm supports are as follows:

1. Version 2.7 of Python 2
2. Python 3: 3.12 and higher versions 3.6 and up

The installation process for PyCharm is stated in the steps below:

- Download the installer .exe. from https://www.jetbrains.com/pycharm/download/?section=windows
- There is a separate installer for ARM64 processors.
- Run the installer and follow the wizard steps.
- Mind the following options in the installation wizard
  - 64-bit launcher: Adds a launching icon to the Desktop.
  - Open Folder as Project: Adds an option to the folder context menu that will allow opening the selected directory as a PyCharm project.
  - .py: Establishes an association with Python files to open them in PyCharm.
  - Add launchers dir to the PATH: Allows running this PyCharm instance from the Console without specifying the path to it.
- To run PyCharm, find it in the Windows Start menu or use the desktop shortcut. You can also run the launcher batch script or executable in the installation directory under bin.

When you run PyCharm for the first time, you can take several steps to complete the installation, customize your instance, and start working with the IDE.

3.3 Setup Python Environment in PyCharm IDE

Once the PyCharm IDE is properly installed, one can use the desktop shortcut or the PyCharm application from the Windows Start menu to launch PyCharm. The executable or batch script for the launcher can also be run from the installation directory's bin folder. After that, you'll get a Welcome screen as shown in Figure 2, which is where you can begin working with the IDE and adjusting its settings. When you close any open projects, you will also get this screen.

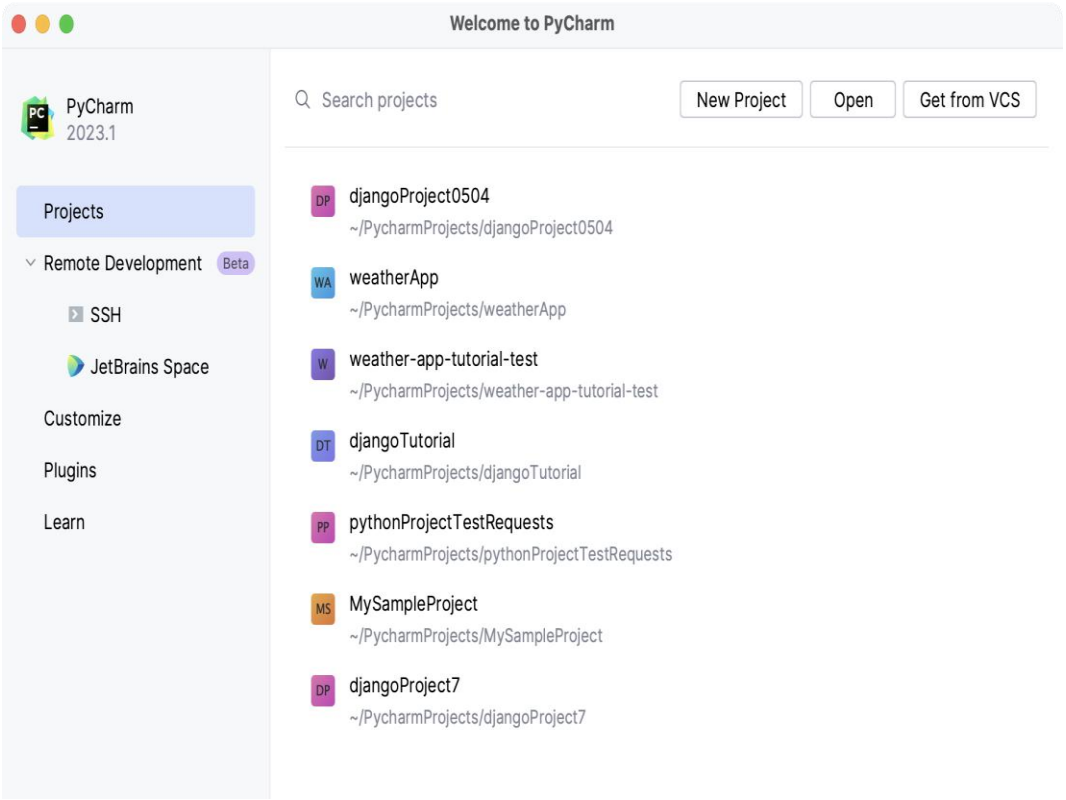


Fig. 2: PyCharm Welcome Screen
(Source: https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html)

From the Welcome to PyCharm dialog, you can do the following:

- Create a new project
- Open an existing project

3.3.1 Create a New Project

1. Click New Project if the Welcome page is displayed to you as shown in Figure 2. Choose File | New Project from the main menu if you already have a project open.
2. While PyCharm allows you to build a variety of project types, for the purposes of this guide, we will construct a basic Pure Python project as shown in Figure 3. An empty project will be created using this template.

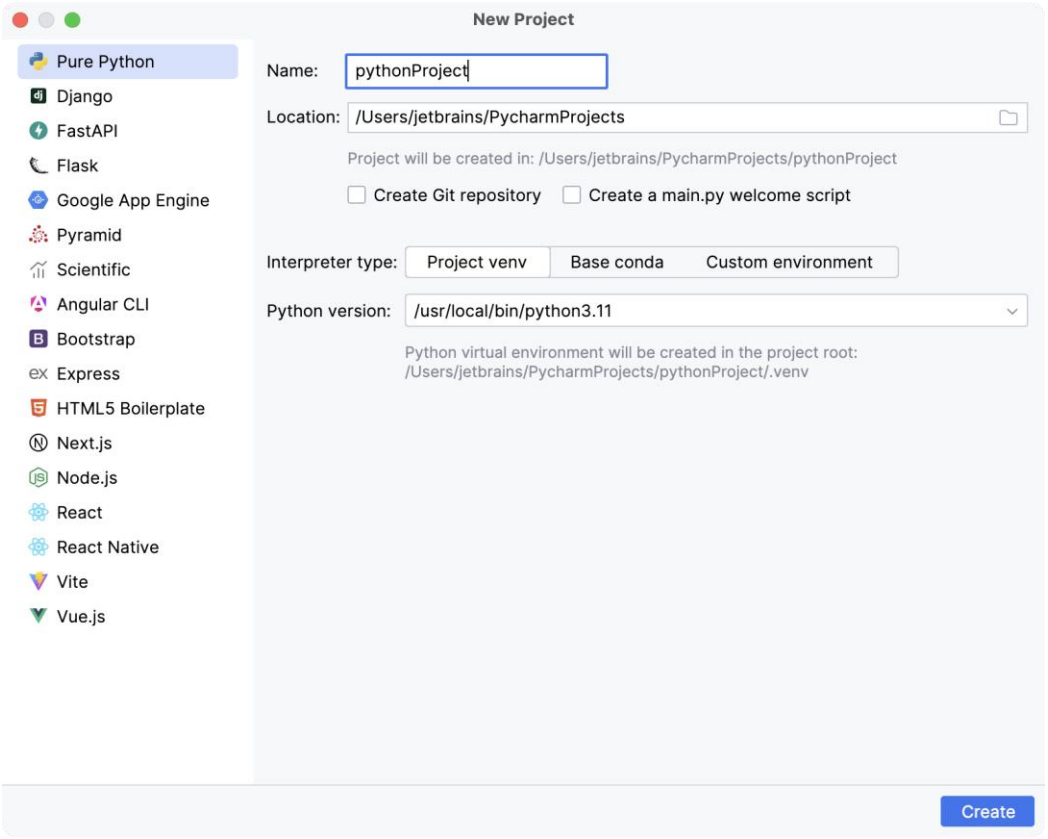


Fig. 3: Creating a new Project
(Source: https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html)

3. Select the project's location. Choose the project directory by clicking the Browse button in the Location column.
4. The recommended practice for Python is to set up a specific environment for every project. For the most part, you will not need to change anything because Project environment's default settings will suffice.
5. Click Create when you are ready.

3.3.2 Creating a python File

1. Choose the project root as shown in Figure 4 (usually the root node in the project tree) in the Project tool window. Then, right-click on it and choose File | New.

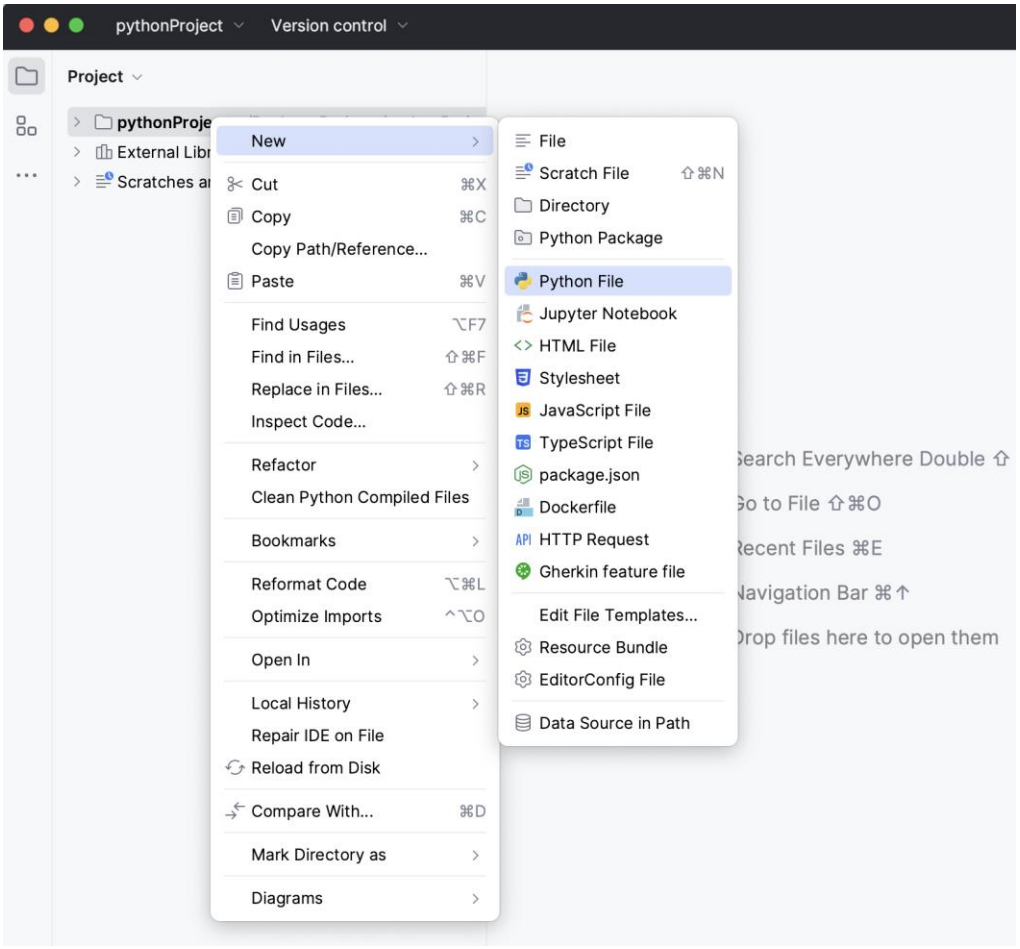


Fig. 4: Creating a Python file
(Source: https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html)

2. Select the option Python File from the context menu as shown in Figure 5, and then type the new filename.

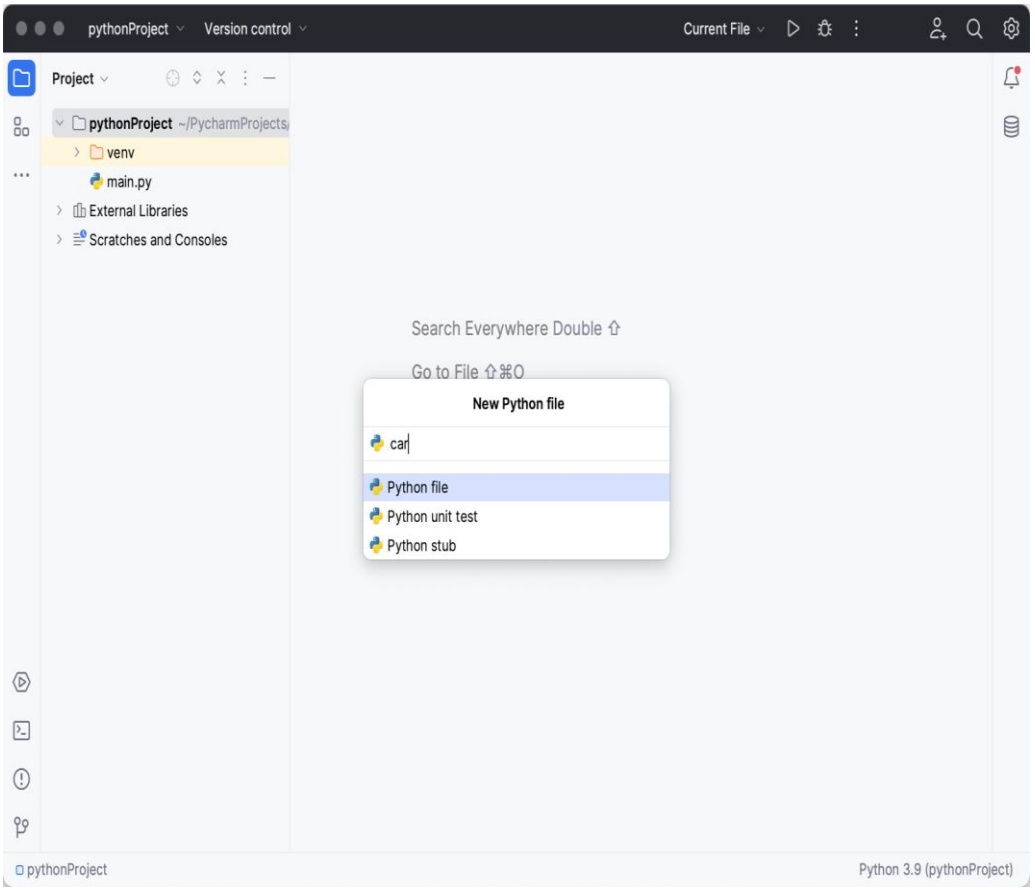


Fig. 5: Naming a Python File
(Source: https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html)

3. PyCharm creates a new Python file and opens it for editing as shown in Figure 6.

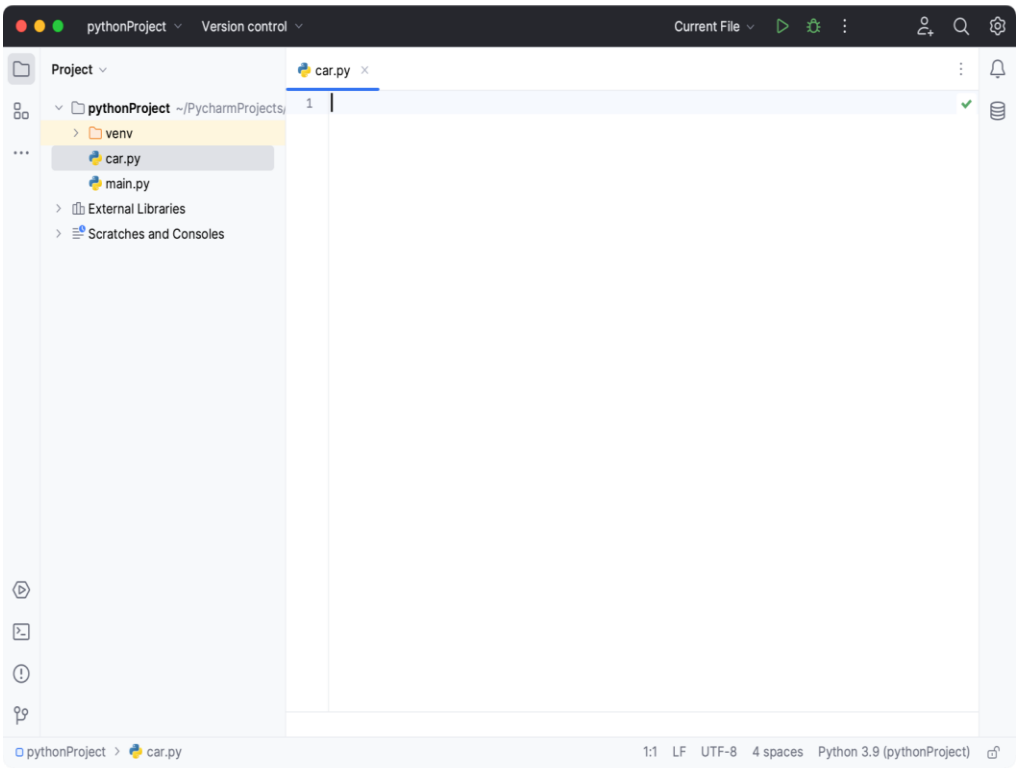


Fig. 6: Sample PyCharm Python File
(Source: https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html)

4.0 Self-Assessment Exercise(S)

1. Differentiate between and interpreted and a compile language:

Answer:
Interpreted languages are executed line after line by an interpreter without the need for an intermediate compilation Step while compiled languages translate the entire code into machine instruction before execution making execution faster.

2. What is an interpreter?

Answer:

An interpreter is a software program that executes an instruction written in high level language directly without converting it into machine code.

Topic for Discussion: Exploring Basic Python Programming Constructs for Security

5.0 Conclusion

In conclusion, understanding the process of installing Python and setting up the environment is crucial for anyone embarking on programming journeys. Installing Python involves selecting the appropriate version and platform-specific installer, ensuring compatibility with existing software and dependencies. Setting up the environment involves configuring paths, installing necessary packages, and possibly integrating with development tools like IDEs or text editors. These initial steps lay the foundation for efficient development by providing a stable and tailored environment where programmers can write, test, and debug code effectively. As Python's popularity grows across various fields, mastering the setup process ensures readiness to explore its vast capabilities in areas such as web development, data analysis, and cybersecurity, empowering developers to harness Python's full potential.

6.0 Summary

In this unit you have learnt that python is an interpreted language, you also learnt about PyCharm IDE, its installation steps and how PyCharm IDE can be used to create projects and Python files.

6.0 References/Further Reading

https://www.jetbrains.com/pycharm/download/?section=windows

https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html

Python for Cyber security: Using Python for Cyber Offense and Defense (2002) 1st Edition by Howard E. Poston III

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.

Unit 3 Basic Python Syntax

Unit Structure

1.0 Introduction

In this unit, I will take you through an introduction to the basic syntax of Python programming language. You will learn about variable and data types, control flow and functions.Basic Python syntax forms the backbone of coding in Python, characterized by its clarity and simplicity. Python uses indentation to indicate code blocks, replacing traditional curly braces or keywords found in other programming languages. This indentation-based structure enhances readability and helps maintain consistent code formatting. Statements in Python are typically terminated by newlines, and comments are prefixed with the hash (#) symbol. Understanding these fundamental syntax rules is crucial for writing clean and understandable Python code, ensuring that beginners and seasoned developers alike can create efficient and reliable programs.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- Define variables and explain their purpose
- Identify and utilize different data types such as integers, floats, strings, Booleans, lists, tuples, and dictionaries.
- Perform basic operations and manipulations on variables and data types.
- Demonstrate comprehension of conditional statements (if, elif, else) and their syntax.
- Define functions and explain their significance in programming.

3.0 Main Content

3.1 Variables and Data Types

Data values are kept in containers called variables. A variable is memory location that is given a name and can be referenced during programming. Variable names can be short (e.g., a and b) or long (e.g., gpa, horsename, total_volume). The guidelines for naming variables in Python include:

1. An underscore character or a letter must appear at the beginning of a variable name.
2. You cannot begin a variable name with a number.
3. Only alpha-numeric characters and underscores (A-z, 0–9, and _) are permitted in a variable name.
4. Car, car, and CAR are three distinct variables, and the variable names are case-sensitive.
5. No Python keyword can appear in the name of a variable.

There is no command in Python for declaring variables. The instant you give a variable a value for the first time, it is formed. An example of variable declaration is presented in code listing 1.

Code Listing 1: Variable Declaration

```
1 x = 5
2 y = "John"
3 print(x)
4 print(y)
```

In code listing 1, two variable are declared namely x and y. x is assigned a value 5 any y is assigned a string value “John”

Code Listing 1: Output

```
5
John
```

Variables in Python can change types after they have been set and are not required to be defined with a certain type. An example of is presented in code listing 2.

Code Listing 2: Variable Change Type

```
1 x = 4# x is of type int
2 x = "Sally"# x is now of type str
```

3 **print**(x)

Casting is an effective way to specify the data type of a variable in Python. An example of the use of casting is shown in code listing 3

Code Listing 3: Casting
1 x = str(**3**) # x will be '3' 2 y = int(**3**) # y will be 3 3 z = float(**3**) # z will be 3.0

Python Data Types

The concept of data type is crucial in programming. Different kinds of data can be stored in variables, and various types can perform different functions. The following data types are pre-installed in Python:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview
None Type:	NoneType

The data type of an object can be obtained using the type() function as shown Code Listing 4.

code Listing 4: The use of type() function
1 2 a =**5** **print**(type(a))

code Listing 4: Output
<class 'int'>

The output indicates that the variable “a” is of type int.

3.2 Control Flow (if statements, loops)

Python control flow constructs, including if statements and loops, are essential tools for managing a program's execution flow. They enable programmers decide what to do and how to do it again depending on certain parameters. An outline of various control flow structures is provided below:

If Statements:
In Python, if statements have a simple syntax as shown below:

```
1 if condition:
2 # statement or statements to execute if condition is True
3 elifalternative_condition:
4 # statement or statements to execute if alternative_condition is True
5 else:
6 # statement or statements to execute if none of the above conditions
  are True
```

Keep in mind that Python uses indentation to construct code blocks inside of if statements, which improves readability. Now let us consider a simple problem that require the use of if statements.

Problem:
Write a Python program to check if a given number is positive, negative, or zero.

Solution: The python code is presented in Code Listing 5

Code Listing 5: Check if a given number is positive, negative, or zero.

```
 1 defcheck_number(number):
 2 if number >0:
 3 return"Positive"
 4 elif number <0:
 5 return"Negative"
 6 else:
 7 return"Zero"
 8
 9 # Taking input from the user
10 num = float(input("Enter a number: "))
11
12 # Calling the function and printing the result
13 print("The number is", check_number(num))
```

A function called check_number() is constructed that accepts a number as input and outputs a string that indicates if the number is zero, positive, or negative. To check the number's value, conditional statements (if, elif, and else) is utilized. The input() function is used to ask the user to enter a number. To accommodate both integer and decimal inputs, a float() function is used to transform the input to a float. We have learnt that in casting. With the input number, the check_number() method is called and output the outcome.

Loops:
Loops are important structures in programming because they allow the execution of a block of code repeatedly. In Python, there are mainly two types of loops: for loops and while loops.

For loops
For loop in Python iterates through a sequence (such a list, tuple, string, or range) and runs the code block for each entry. The syntax of a for loop in Python is given below:

```
1 for item in sequence:
2 # Code block to be executed
```

An example presented in Code Listing 6

Code Listing 6: Example of FOR Loop

```
1 fruits = ["apple", "banana", "cherry"]
2 for fruit in fruits:
3 print(fruit)
```

Code Listing 6 creates a list of fruits, uses a for loop to print the fruits items in the list as shown in the output below:

Code Listing 6: Output

```
apple
banana
cherry
```

While loops
While loop in Python continually runs a block of code as long as a condition is true. The syntax of a while loop in Python is given below:

```
1 while condition:
2 # Code block to be executed
```

An example presented in Code Listing 7

Code Listing 7: Example of while Loop

```
1 count =0
2 while count <5:
3 print(count)
4     count +=1
```

Code Listing 7 creates variable called count, initialize count to 0. The

while condition checks the variable count and print its value as long as the condition evaluates to true. The output is shown below:

Code Listing 7: Output
0 1 2 3 4

In-Text Question
Why are control flow constructs like if statements and loops important in Python programming?

Answer
Control flow constructs like if statements and loops are important because they enable programmers to manage a program's execution flow by making decisions based on certain conditions and repeating actions as needed.

3.3 Functions

A function in Python is a piece of reusable code that carries out a particular activity. By using functions, you may divide your code into more manageable, smaller chunks that are simpler to read, write, and amend. The major advantage of using function in programming is reusability meaning a generalized function can be used a number of times. Functions are also easy to debug. In python a function need to be defined and called. The syntax for defining a function is guided by the following point:

- Every function should begin with a **def** keyword
- Every function may or may not have arguments or parameters which should be in parenthesis
- Every function should returned to its calling function either empty or value

Below is function definition syntax in python

```
1 deffunction_name(parameters):
2 # Function body
3 # Statements
4 return value  # Optional
```

A function need to be called which may be written in a parent program and the syntax for a Python call is guided by the following point:

- A calling function should have a function name
- if there are argument of parameters, they must be the same as in the function definition

Example
Write a function in Python to read two integers from the user compute the sum of the integers and print the result.

The solution to this example is given in Code Listing 8.

Code Listing 8: A program to add two numbers

```
1  # A function to add two numbers
2  # User defined function
3  defsum(a, b):
4  sum = a + b
5  returnsum
6  # Actual Program
7  a = int(input('Enter 1st number: '))
8  b = int(input('Enter 2nd number: '))
9  res = sum(a,b)
10 print ("The sum is =", res)
```

The output of the program in Code Listing 8 is shown below:

Code Listing 8: Output

```
Enter 1st number:
2
Enter 2nd number:
4
The sum is = 6
```

4.0 Self-Assessment Exercise(S)

1. What are some best practices for effectively using functions in Python to maximize code reusability and maintainability?

ANSWER:

Some best practices for effectively using functions in Python to maximize code reusability and maintainability include modularizing code into smaller functions that perform specific tasks, using meaningful names and clear documentation (docstrings) to enhance readability and understanding, and parameterizing functions to make them adaptable for different inputs and contexts. These practices not only improve code

organisation but also facilitate easier debugging, testing, and future modifications.

2. What is the advantage of using functions to write programs?

The major advantage of using function in programming is reusability meaning a generalized function can be used a number of times. Functions are also easy to debug.

3. Write a function in Python to read two integers from the use, compute the product of the integers and print the result.

A program to find the product of two numbers

```
# A function to find the product of two numbers
# User defined function
Defproduct (a, b):
    product = a * b
return product
# Actual Program
a = int(input('Enter 1st number: '))
b = int(input('Enter 2nd number: '))
res = product(a,b)
print ("The product is =", res)
```

Topic for Discussion: Secure Coding Practices and Defensive Programming in Python

5.0 Conclusion

In conclusion, understanding basic Python syntax forms the cornerstone of programming proficiency. Understanding constructs like variables, data types, control flow statements (such as loops and conditionals), and functions is fundamental for writing clear, efficient code. Python's syntax emphasizes readability and simplicity, making it accessible for beginners while offering powerful features for advanced applications. Mastery of these foundational elements enables developers to build more complex algorithms, handle data effectively, and implement solutions across diverse domains, from web development to scientific computing and cybersecurity. As programmers deepen their understanding of Python syntax, they unlock the language's versatility and streamline their ability to solve real-world problems with elegance and clarity.

6.0 Summary

In this unit, you were taught the introduction to the basic syntax of Python programming language which includes variable and data types, control flow and functions.

7.0 References/Further Reading

Python for Cybersecurity: Using Python for Cyber Offense and Defense (2002) 1st Edition by Howard E. Poston III

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newness.

Unit 4 Overview of Strings in Python

Unit Structure

1.0 Introduction

Having learnt some basics of Python programming language, I will introduce you to Strings in Python. In this Unit you will be learning about basic string operations, how to create strings concatenation and formatting of strings. In Python, strings are fundamental data types used to store and manipulate text. They are defined within single quotes (' ') or double quotes (" "), allowing flexibility in handling both simple characters and more complex text with spaces, punctuation, and special characters. Python provides various methods to manipulate strings, such as concatenation (joining strings together), slicing (extracting specific parts of a string), and formatting (inserting variables into strings). Strings are immutable in Python, meaning once created, they cannot be changed. Understanding how to work with strings effectively is essential for tasks ranging from basic text processing to more advanced applications in data handling and algorithm development.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- define strings in Python
- create strings using Python
- access characters in string
- perform string concatenation in Python.

3.0 Main Content

1.1 What are strings?

A String is referred to as a group or sequence of characters. In python a string can be a single word or a group of words that form a sentence. Strings are created in python using either a single quote, double or triple quote. In python strings are immutable meaning the elements of a string cannot be changed once created. To learn more on strings, you can visit (https://www.naukri.com/code360/library/string-manipulation-in-python)

In-Text Question

How are strings created in Python and what does it mean that they are immutable?

Answer

Strings in Python are created using single quotes, double quotes, or triple quotes. They are immutable, meaning that once a string is created, its elements cannot be changed.

1.2 Importance of String Manipulation in Programming

Programming requires extensive string manipulation for a number of reasons which includes:

1. Data processing: Tasks involving data processing frequently use strings. Processing and extracting meaningful information from data requires manipulating strings, whether the data is being parsed from files, network streams, or user inputs.
2. Text Processing: Word processors, online browsers, messaging apps, and other programs deal mostly with textual data. Efficient activities like text analysis, formatting, replacement, and search are made possible via string manipulation.
3. User Interaction: String manipulation is essential in user interfaces to provide information to users, verify inputs, and give feedback. Handling user commands and creating dynamic user interface elements both require the ability to manipulate strings, which is essential for user interaction.
4. Data Representation: Dates, integers, and structured formats such as XML and JSON can all be represented using strings. It is possible to convert between different data types by manipulating strings.

1.3 Basic String Operations

Strings are immutable character sequences in Python, and the language offers many built-in methods and operators for handling common string operations. Several fundamental Python string operations include concatenation, indexing, length, formatting etc.

1.4 Creating strings

Strings are created in python using either a single quote, double or triple quote. The syntax for creating a string using the three quotes is given below:

1. Using single quote

```
1 string= 'Welcome to Python Programming.'
```

2. Using double quote

```
1 string= "Welcome to Python Programming."
```

3. Using triple quote

```
1 string= '''Welcome to Python Programming.'''
```

In Python, all of these methods will produce string objects. Depending on your taste and the particular needs of your string—particularly if you need to include quotations or newlines inside the string itself—you can choose between single, double, or triple quotes.

Strings can also be read using an input function method at run time from the keyboard as shown below

```
1 user_input = input("What is your name: ")
2 print("Your name is:", user_input)
```

This code will ask the user what is your name when it runs. The user_input variable will hold whatever the user types. The entered string will then be printed by the application.

1.5 Accessing Characters in a String

Characters of a string can be accessed in python using the index method. Note that the index of the string starts from 0 to N −1 where N is the length of the string. Each element of the string can be accessed using its index as shown in Code Listing 9:

Code Listing 9: Index Method
1 string ="Welcome to Python" 2 **print**(string[**0**]) # Output: P 3 **print**(string[-**1**]) # Output: n

1.6 String Concatenation

String concatenation is the ability to combine 2 different strings using the plus (+) operator. An example is shown in Code Listing 10:

Code Listing 10: String Concatenation
1. str1 ="Hello" 2. str2 ="Welcome to Python Programming" 3. concatenated_str = str1 +", "+ str2 4. **print**(concatenated_str) # Output: Hello, Welcome to Python Programming

4.0 Self-Assessment Exercise(s)

1. What are some other ways you could concatenate `str1` and `str2` in Python, and how might they differ in terms of readability or performance?

ANSWER:
In Python, besides using the + operator for string concatenation as shown in the code snippet, you can also use formatted strings (f-strings) or the .join() method for concatenating str1 and str2. Here's how they compare:

1. **Formatted Strings (f-strings):**

python
Copy code
concatenated_str = f"{str1}, {str2}"
Advantages: F-strings are concise and offer improved readability by embedding variables directly into the string. **Performance:** They are generally faster than traditional concatenation methods due to optimisations in Python's string handling.

2. **.join() Method:**

python
Copy code
concatenated_str = ", ".join([str1, str2])

Advantages:.join() is effective for concatenating multiple strings efficiently, especially when dealing with iterable data. **Performance:** It can be more efficient than using + for concatenating many strings because it minimizes the creation of intermediate string objects.

Comparison:

- **Readability:** F-strings are highly readable and concise, especially for combining a few strings. .join() is clear but requires converting strings into a list or tuple.
- **Performance:** F-strings and .join() are generally faster and more memory-efficient than using +, especially with larger strings or many concatenations.

Choosing between these methods often depends on the specific use case, readability preferences, and performance considerations in your Python code.

2. Write a python program that prompt the user for his first name and prints it.

ANSWER:

```
1 # Prompting the user for their first name
2 first_name = input("Please enter your first name: ")
3
4 # Printing the entered first name
5 print("Your first name is:", first_name)
```

Topic for Discussion: Malware Analysis and Detection Techniques with Python

5.0 Conclusion

Understanding strings in Python is crucial as they are fundamental for handling text data and manipulating characters within programs. Python's string operations, such as concatenation, slicing, and formatting, provide powerful tools for data manipulation and transformation. Strings are immutable in Python, meaning they cannot be changed once created, which affects how operations are performed on them. With Python's support for Unicode, handling different character encodings and internationalization becomes straightforward. Additionally, Python offers a rich set of built-in string methods for common tasks like searching, replacing, and formatting strings, enhancing productivity and code readability. Mastery of string handling in Python is essential for developing robust applications across various domains, from simple text processing to complex data analysis and web development tasks.

6.0 Summary

In this unit, you have been introduced to strings in Python, the importance of string manipulation, basic string operations, how to create strings, concatenation and formatting of strings.

7.0 References/Further Reading

Python for Cybersecurity: Using Python for Cyber Offense and Defense (2002) 1st Edition by Howard E. Poston III

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newness.

https://www.naukri.com/code360/library/string-manipulation-in-python

Module 2 Working with Networks and Sockets

Module Introduction

In this module, you will learn Introduction to Networking Concepts which involves Understanding TCP/IP, OSI Model Overview. You will also learn Socket Programming with Python starting with the basics and moving to creating a simple client-server application and Handling connections and data transfer.

The module is organised into two units. These are as follows:
Unit 1 Introduction to Networking Concepts
Unit 2 Socket Programming with Python

UNIT1 INTRODUCTION TO NETWORKING CONCEPTS

Unit Structure

1.0 Introduction

In this Unit, you are going to learn the introduction to networking concepts, which involve Understanding TCP/IP and Model Overview. Networking concepts in Python involve using libraries and modules to facilitate communication between devices over networks. Python provides robust libraries such as `socket` for low-level networking tasks like creating network connections, sending and receiving data over TCP/IP or UDP protocols. Higher-level libraries like `requests` simplify HTTP requests, making it easier to interact with web services and APIs. Understanding networking in Python includes grasping concepts like IP addressing, port numbers, sockets, protocols (such as HTTP, FTP, SMTP), and handling data transmission securely. Python's versatility in

networking makes it suitable for developing applications ranging from simple client-server interactions to complex distributed systems.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- define communication network
- explain OSI Model Overview
- explain TCP/IP

3.0 Main Content

3.1 What is a Network?

What is a network? Normally at home you have a bunch of devices such as computers, printers tablet etc., you connect these devices by means of a network when devices are connected, they are able to share data this could be sending jobs to a printer, sending an email or even streaming in videos. Network can also be used for sharing internet connection. In other for devices to communicate, they need to be connected somehow. One way is to connect cables to these devices and connect them to another device called a switch. However, there are protocols and models for building networks which may either be the Open System Interconnection (OSI) model or the Transmission Control Protocol/Internet Protocol TCP?IP model.

3.2 OSI Model Overview

The OSI model is a theoretical stock of seven (7) layers that can be used as a reference to understand how network operate. The model was introduced to standardized networks in a way that allowed a multi-vendor system. Prior to this, there only exists one vendor for network because devices from one vendor could not communicate with others.
Note that the OSI model is not practically in use, we use TCP/IP model which have similar concept but the layers are a little different. Although OSI model is not in use but it is used as a reference point when troubleshooting a network. Figure 7 represent the layers of the OSI model.

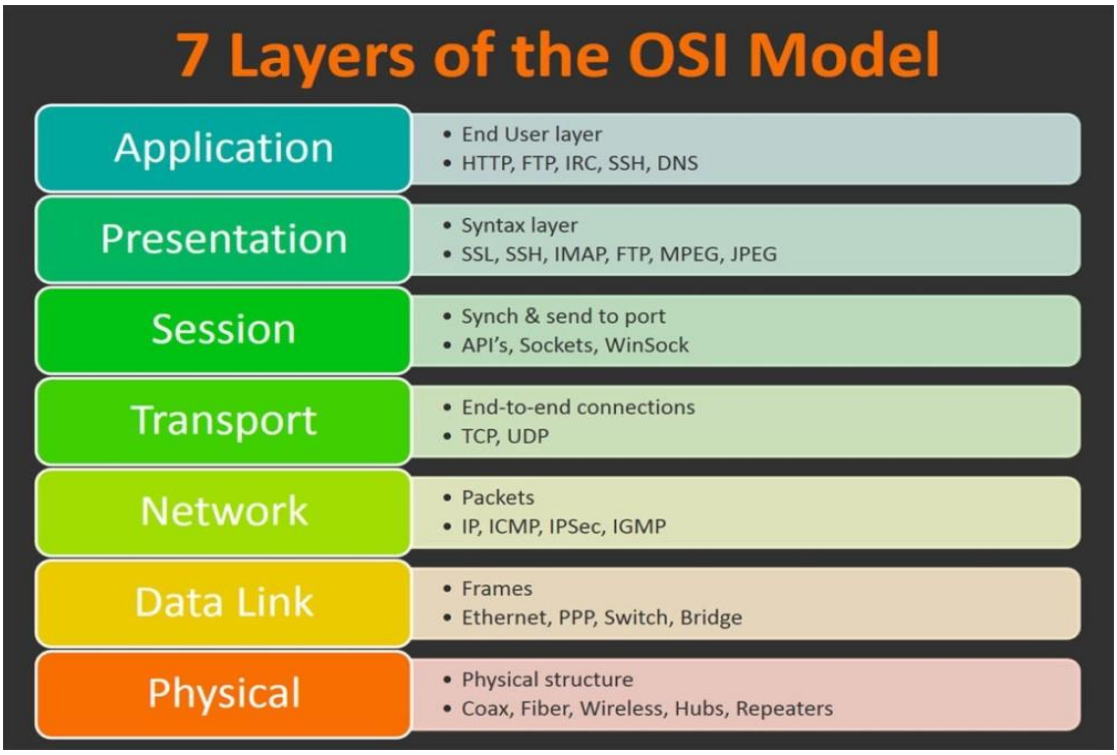


Figure 7: 7 Layers of the OSI Model (Source:https://www.linkedin.com/pulse/osi-model-explained-how-easily-remember-its-7-layers-rahul-jasrotia)

1. **Physical layer**: This layer is responsible for data movement across physical hardware such as cables, network interface cards and or hubs.
2. **Data Link Layer:** This layer is where physical addresses are added to data. Switches are also located at this layer as well.
3. **Network Layer**: This is a layer that handles IP addresses and routine. Source and destination IP addresses are added as this layer.
4. **Transport layer**: The transport layer adds the transport protocols such as the TCP the UDP and port numbers.
5. **Session Layer**: This layer is responsible for adding and terminating connection between devices.
6. **Presentation Layer**: This layer for formats data in a way that receiving applications can understand it. This layer is also used to encrypt and decrypt data.
7. **Application layer**: This is the layer where application and users communicate, applications specific protocols such as SMTP, FTP and Telnet are utilized in this layer.

3.3 Understanding TCP/IP

TCP/IP model is a model was built to standardize computer networking. The TCP/IP model is the model that is practically used in building computer networks the TCPIP model is made up of five (5) layers a shown in Figure 8.

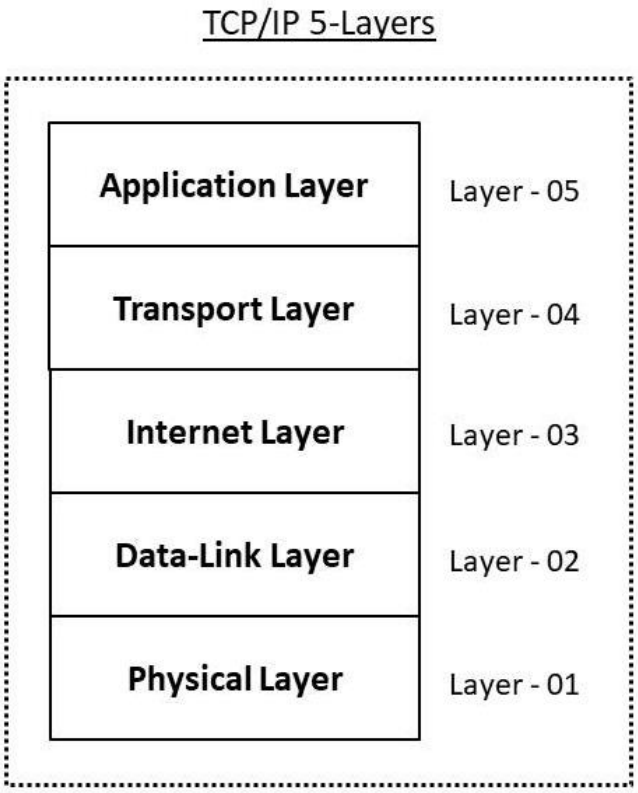


Figure 8: TCP/IP 5-Layers (Source:https://afteracademy.com/blog/what-is-the-tcp-ip-model-and-how-it-works/)

1. **Physical layer**: This layer is responsible for data movement across physical hardware such as cables, network interface cards and or hubs.
2. **Data Link Layer:** This layer is where physical addresses are added to data. Switches are also located at this layer as well.
3. **Internet Layer**: The Network layer of the OSI model and the Internet layer of the TCP/IP model are similar in many ways. It works with data that is organised into packets or datagrams. This layer primarily adds the IP (Internet Protocol) address to data packets to accomplish logical addressing. Either Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6) can be

used for IP addressing. Using IP addresses, the Internet layer also handles data packet routing.

4. **Transport layer**: The transport layer adds the transport protocols such as the TCP the UDP and port numbers.
5. **Application layer**: The upper three levels (Application, Physical, and Session Layer) of the OSI model are corresponding to the Application layer in the TCP/IP model. It addresses the transmission of the entire data message. An interface between the application programs and the network services is provided by the application layer. It primarily offers end users network services so they may work together. For instance, transferring files, surfing the web, etc. All higher-level protocols, including DHCP, FMTP, SNMP, SMTP, FTP, HTTP, HTTPS, and Telnet, are used by this layer.

4.0 Self-Assessment Exercise(s)

1. Which layer is responsible for data movement across physical devices?

Answer:
The physical layer

2. Which layers in the OSI model corresponds to the application layer of the TCP/IP model?

Answer:
Application, Physical, and Session Layer

Topic for Discussion: Understanding Network Security Fundamentals with Python

5.0 Conclusion

Networking concepts in Python for cybersecurity provide a foundational understanding of how Python can be leveraged to interact with network protocols, devices, and services. Python's socket programming capabilities enable developers to create networked applications that can communicate over the internet or local networks using TCP/IP or UDP protocols. Understanding concepts like sockets, IP addressing, port numbers, and protocols (such as HTTP, HTTPS, and DNS) is essential for implementing secure communication channels and performing

network reconnaissance tasks. Python's libraries and frameworks, such as `socket`, `requests`, and `scapy`, facilitate network programming tasks, making it versatile for tasks ranging from network scanning to creating secure communication channels for data transmission. Mastery of networking concepts in Python equips cybersecurity professionals with the skills to analyze network traffic, detect vulnerabilities, and secure network infrastructures effectively.

In this unit, you have learnt that, despite minor differences in their layers and structure, the TCP/IP (Transmission Control Protocol/Internet Protocol) and OSI (Open Systems Interconnection) models both function as frameworks for understanding and implementing computer networks.

6.0 Summary

Models like OSI and TCP/IP are essential to the standardization and interoperability of computer networks. The seven-layered OSI provides a theoretical framework for comprehending network functions. The practical model that is most frequently used in network implementation is TCP/IP, which has five layers. The upper layers of both models concentrate on applications, while the bottom layers handle physical concerns. Understanding these models facilitates troubleshooting and the development of effective networks;

7.0 References/Further Reading

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newness.

https://www.linkedin.com/pulse/osi-model-explained-how-easily-remember-its-7-layers-rahul-jasrotia
https://afteracademy.com/blog/what-is-the-tcp-ip-model-and-how-it-works/

UNIT 2 SOCKET PROGRAMMING WITH PYTHON

Unit Structure

1.0 Introduction

In this unit, you will learn about socket programming in Python. You will be equipped with basic of socket, how to create a simple client-server application and learn how to handle connection and data transfer. Socket programming in Python involves using the `socket` module to create network connections between devices over a network. It enables communication between two endpoints (client and server) by establishing a connection, sending data, and receiving responses. Socket programming is essential for building networked applications that can exchange data in real-time, such as chat applications, file transfer systems, and network protocols implementation. Python's `socket` module provides both low-level access to network functionalities, allowing developers to implement custom protocols, as well as higher-level abstractions for common tasks like TCP/IP or UDP communication. Understanding socket programming in Python involves learning about socket creation, binding to specific addresses and ports, listening for incoming connections, and handling data transmission securely and efficiently.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- Explain what is socket
- create a simple client server application

- handle connection and data transfer

3.0 Main Content

3.1 Socket Basics

Think about what we can do on the Internet. There are a lot of machines merely referred to as nodes. Now all these nodes can be either a server or a client.
Therefore, a client-server network or a client-client network may be established. All these nodes talk to each other basically using peer to peer P2P network. The focus here is on client-server networks. There are always some services on the internet, servers on the internet listen to the clients to send a request for which the server will send back a response to the client. Since we have been introduced to the basics of python and network on some previous units, now it is time to implement the use of python. Socket programming uses two (2) things namely the server and the client. In order to understand socket programming we need to get two concepts Clear:

1. **Port Number**: A machine on the network can provide multiple services, now each of the services run on a particular port and each port has a port number. As an example will be using port number 9999 which is a port number that does not provide any global service.
2. **Type of connection**: We can have a connection oriented or a connectionless protocol. A connection oriented means that a connection needs to be established before data is sent from source to destination while UDP uses no connection that means you only send packets from source to destination. However, the disadvantage of UDP is that you have no guarantee whether the packet has reaches its destination or not.

In-Text Question
What are the two key concepts to understand in socket programming, and how do they differ?

Answer
The two key concepts to understand in socket programming are Port Number and Type of Connection. A Port Number identifies a specific service on a machine, while the Type of Connection can be either connection-oriented (requiring a connection to be established before data transfer) or connectionless (such as UDP, which sends packets without establishing a connection and offers no guarantee of delivery).

3.2 Creating a Simple Client-Server Application

Python's socket module provides an interface to the Berkeley sockets API. This is the module that we will use in this unit. The primary socket API functions and methods in this module are:

.socket()
.bind()
.listen()
.accept()
.connect()
.connect_ex()
.send()
.recv()
.close()

We will now create a simple server as shown in Code Listing 11 that listens on a specific port.

Code Listing 11: Creating a Server

```
1  # An example script to connect to Google using socket
2  # programming in Python
3  importsocket# for socket
4  importsys
5
6  try:
7          s                =                socket.socket(socket.AF_INET,
8  socket.SOCK_STREAM)
9          print ("Socket successfully created")
10 except socket.error as err:
11         print ("socket creation failed with error %s" %(err))
12
13 # default port for socket
14 port = 80
15
16 try:
17         host_ip = socket.gethostbyname('www.google.com')
18 except socket.gaierror:
19
20         # this means could not resolve the host
21         print ("there was an error resolving the host")
22         sys.exit()
23
24 # connecting to the server
25 s.connect((host_ip, port))
26
```

```
print ("the socket has successfully connected to google")
```

Output

```
Socket successfully created
the socket has successfully connected to google
```

To learn more on socket programming using python, visit (https://realpython.com/python-sockets/)

3.3 Handling Connections and Data Transfer

Initially, we created a socket. Google's IP was then resolved, and finally, we were able to connect to Google in the previous section. We now need to figure out how to send data via a socket. The socket library includes a send all function for data transmission. Using this function, you can send data to a server that the socket is connected to, and the server can use it to deliver data to the client. A client that connects to the server and sends a message to it is shown in Code Listing 12.

Code Listing 12: A simple client program

```
# first of all import the socket library
importsocket

# next create a socket object
s = socket.socket()
print ("Socket successfully created")

# reserve a port on your computer in our
# case it is 12345 but it can be anything
port =12345

# Next bind to the port
# we have not typed any ip in the ip field
# instead we have inputted an empty string
# this makes the server listen to requests
# coming from other computers on the network
s.bind(('', port))
print ("socket binded to %s"%(port))

# put the socket into listening mode
s.listen(5)
print ("socket is listening")

# a forever loop until we interrupt it or
# an error occurs
whileTrue:
```

```
22.  # Establish connection with client.
23.  c, addr = s.accept()
24.  print ('Got connection from', addr )

25.  # send a thank you message to the client. encoding to send byte type.
26.  c.send('Thank you for connecting'.encode())

27.  # Close the connection with the client
28.  c.close()

29.  # Breaking once connection closed
30.  break
```

Start the server script first, then launch the client script to make this run. It should be visible that the client is communicating with the server and getting a response.

4.0 Self-Assessment Exercise(s)

1. What is the difference between connection oriented and connectionless communication?

Answer:
A connection oriented means that a connection needs to be established before data is sent from source to destination while connectionless communication does not need to guarantee an established communication.

2. *What is the disadvantage of UDP?*

Answer:
The disadvantage of UDP is that you have no guarantee whether the packet has reaches its destination or not.

Topic for Discussion: Socket Programming and Network Communication Security

5.0 Conclusion

Socket programming with Python is crucial in cybersecurity for establishing network connections and facilitating communication between devices over the internet or local networks. Python's `socket` module provides a straightforward interface for creating client-server applications, enabling data exchange through sockets using TCP or UDP protocols. Understanding socket programming allows cybersecurity professionals to implement secure communication channels, perform network reconnaissance, and develop tools for monitoring network traffic and detecting anomalies. Mastery of socket programming in Python equips practitioners with the skills to build robust network applications, handle network protocols securely, and respond effectively to cybersecurity threats involving network-based attacks.
In this unit, you have learnt that Python socket programming makes it possible for nodes in a network to communicate with one another, making client-server networks easier to set up.

6.0 Summary

Python socket programming facilitates network node-to-node communication, with a primary focus on client-server networks. Important elements are port numbers, which designate certain services operating on a system, and connection type, which differentiates between protocols that are connection-oriented (like TCP) and connectionless (like UDP). The socket module in Python offers all the necessary methods and functions for socket programming, including opening and shutting sockets, binding sockets to ports and addresses, listening for incoming connections, accepting connections, connecting to servers, and sending and receiving data. A client connects to the server and transmits messages, and the server waits on a designated port in a basic client-server program.

7.0 References/Further Reading

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.

https://realpython.com/python-sockets/

MODULE 3 WEB SCRAPING AND RECONNAISSANCE

Module Introduction

In this module, you will learn the concept web of web scrapping and reconnaissance. By the time you complete the module, you will be able to explain the concept of web scrapping and reconnaissance using python.
The module is organised into two units as follows:

Unit 1: Introduction to Web Scraping
Unit 2: Using Python for Reconnaissance

UNIT 1 Introduction to Web Scraping

Unit Structure

1.0` Introduction

In this unit you will learn about web scrapping, web scraping libraries and the practical implementation of web scrapping using Python. Web scraping in Python refers to the automated process of extracting data from websites. It involves fetching web pages, parsing the HTML or XML content, and extracting the desired information for analysis, storage, or further processing. Python offers powerful libraries such as BeautifulSoup and Scrapy that simplify the scraping process by providing tools to navigate through the HTML structure, handle cookies, sessions, and forms, and simulate human interactions with websites. Web scraping is widely used for gathering data from various sources on the internet, including news articles, product information, weather data, and more. However, it's crucial to adhere to ethical guidelines and respect website terms of service when performing web scraping activities.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- explain the concept of web scrapping
- discuss the use of different we scrapping libraries in Python
- perform we scrapping using Python

3.0 Content

3.1 Web Scrapping

Web scraping is the automated method of gathering data from websites. Certain websites expressly prohibit users from using automated tools to scrape their data. Websites may act in this manner for two reasons. The website has good reason to safeguard its information. For example, you cannot request too many results too rapidly in Google Maps.

Sending a website's server a lot of repetitive queries can use up bandwidth, slowing it down for other users and possibly overloading it to the point where the website stops working altogether.

You should always check the acceptable use policy of the website you are targeting to see if utilizing automated tools to access the website violates its conditions of use before using your Python abilities for web scraping. Web scraping against a website's wishes is still very much in the legal unclear area.

In-Text Question
Why should you check a website's acceptable use policy before performing web scraping?

Answer
You should check a website's acceptable use policy before performing web scraping to ensure that using automated tools to access the website does not violate its conditions of use and to avoid legal and ethical issues, as web scraping against a website's wishes is often legally unclear.

3.2 Introduction to Web Scrapping Libraries

There are a number of web scrapping libraries that are provided by python. Some of which includes:

1. **REQUESTS**: Based on urllib3, Requests is a Python web scraping toolkit designed for ease of use. It doesn't need a PoolManager instance to obtain a URL directly. Additionally, you may use the content property on the response object to retrieve the contents of the web page after making a GET call. Developers can communicate with web services and APIs more easily since it streamlines the process of generating HTTP queries and managing responses.
2. **ZenRows**: A Python web scraping toolkit called ZenRows API can handle the most common issue with scraping, which is being blocked. It has several capabilities, such as JavaScript rendering, geo-targeting, rotating and premium proxies, and a headless browser. You will save aggravation, time, and money by using ZenRows.
3. **Selenium**: A popular Python scraping tool for gathering dynamic web material is called Selenium. It simulates button clicks, form completion, and other human-to-human interactions. Because Selenium works with so many different browsers—including Chrome and Firefox—you may select the one that works best for your web scraping project. This adaptability contributes to consistent outcomes in various browser contexts**.**
4. **Beautiful Soup**: Beautiful Soup is a strong Python web scraping toolkit that excels at parsing both HTML and XML documents. One of its most well-liked benefits is its convenience. Based on popular Python parsing programs, Beautiful Soup lets you experiment with various methods. You can use Beautiful Soup to scan a document that has already been parsed and find every piece of data that falls under a specific category or format. It is quite good at detecting encodings.

3.3 Libraries for Web Scraping (e.g. Requests)

We shall use a reqbin.com page to learn how to perform web scrapping using the request library of python.

To send an HTTP GET request using the Python Requests library, you must call the requests.get() method and pass the target URL as the first parameter. This can be achieved using the Python code below. In this Python Requests GET example, we send a GET request to the ReqBin echo URL.

```
1. importrequests
2. r = requests.get('https://reqbin.com/echo/get/json',
```

```
1.      headers={'Accept': 'application/json'})
3.      print(f"Status Code: {r.status_code}, Content: {r.json()}")
```

Output
Status Code: 200, Content: {'success': 'true'}

A response object is returned by the GET method. It may be used to retrieve the HTML data with the content property and the status code (in this case, 200) with the status code property. The variable r contains the response object.

The headers = parameter allows you to pass HTTP headers to Python Requests Library methods. We send custom HTTP headers to the ReqBin echo URL in this Python Requests Headers Example as shown below:

```
1.      importrequests
2.      url ='https://reqbin.com/echo'
3.      headers = {'Accept': '*/*', 'X-User-IP': '1.1.1.1'}
4.      r = requests.get(url, headers=headers)
5.      for key, val in r.headers.items():
6.      print(key, ':', val)
```

Output
Date : Sat, 30 Mar 2024 10:40:21 GMT Content-Type : text/html; charset=utf-8 Transfer-Encoding : chunked Connection : keep-alive Cache-Control : , private, s-maxage=0, proxy-revalidate Display : orig_site_sol Pagespeed : off Response : 200 Vary : Accept-Encoding X-Middleton-Display : orig_site_sol X-Middleton-Response : 200 X-Sol : orig Last-Modified : Sat, 30 Mar 2024 03:02:46 GMT CF-Cache-Status : EXPIRED Report-To : {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=BbhRaW%2FY2m000resSuBHk%2Bqe%2FXgz%2B5dpKQSkyVEXPVNz0Q3vlo2AAVsP4kL2OtEUlck5bACeGLKcw%2FFc6ab9bRK3Pu51Bhmf5vQgDiTGlYMTZvS8iy5v1H9Kix8%3D"}],"group":"cf-

```
nel","max_age":604800}
NEL : {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
referrer-policy : same-origin
x-content-type-options : nosniff
x-frame-options : SAMEORIGIN
x-xss-protection : 1; mode=block
Access-Control-Allow-Origin : https://reqbin.com
Server : cloudflare
CF-RAY : 86c77b268ccb8cc5-EWR
Content-Encoding : gzip
alt-svc : h3=":443"; ma=86400
```

4.0 Self-Assessment Exercise(S)

1. What are the implications of the headers Cache-Control, CF-Cache-Status, and CF-RAY in the context of HTTP caching and server response handling, and how do they contribute to optimising web performance and reliability?

Answer
The Cache-Control header specifies how caching should be managed by browsers and intermediary caches, such as proxies. It dictates whether a response can be cached and under what conditions it can be reused, thus optimising web performance by reducing server load and improving response times for subsequent requests.
CF-Cache-Status is a Cloudflare-specific header that indicates the caching status of a response within Cloudflare's edge servers. It helps in understanding whether a response was served from the cache (HIT), bypassed the cache (MISS), or expired (EXPIRED), which aids in troubleshooting and optimising cache configuration for faster content delivery.
CF-RAY is a unique identifier generated by Cloudflare for each request. It helps in tracking and identifying requests within Cloudflare's network, providing insights into request handling, diagnostics, and troubleshooting, which collectively contribute to enhancing web reliability and performance monitoring.

2. State and explain four libraries that can be used for web scrapping in Python.

Answer
There are a number of web scrapping libraries that are provided by python. Some of which includes:

1. **REQUESTS**: Based on urllib3, Requests is a Python web scraping toolkit designed for ease of use. It doesn't need a PoolManager instance to obtain a URL directly. Additionally, you may use the content property on the response object to retrieve the contents of the web page after making a GET call.Developers can communicate with web services and APIs more easily since it streamlines the process of generating HTTP queries and managing responses.
2. **ZenRows**:A Python web scraping toolkit called ZenRows API can handle the most common issue with scraping, which is being blocked. It has several capabilities, such as JavaScript rendering, geo-targeting, rotating and premium proxies, and a headless browser. You will save aggravation, time, and money by using ZenRows.
3. **Selenium**: A popular Python scraping tool for gathering dynamic web material is called Selenium. It simulates button clicks, form completion, and other human-to-human interactions. Because Selenium works with so many different browsers—including Chrome and Firefox—you may select the one that works best for your web scraping project. This adaptability contributes to consistent outcomes in various browser contexts**.**
4. **Beautiful Soup**: Beautiful Soup is a strong Python web scraping toolkit that excels at parsing both HTML and XML documents. One of its most well-liked benefits is its convenience. Based on popular Python parsing programs, Beautiful Soup lets you experiment with various methods. You can use Beautiful Soup to scan a document that has already been parsed and find every piece of data that falls under a specific category or format. It is quite good at detecting encodings.

Topic for Discussion: Web Scraping and Data Collection for Security Analysis

5.0 Conclusion

Web scraping with Python is a powerful technique in cybersecurity for extracting and analyzing data from websites. It enables practitioners to automate data collection, monitor online content for security threats, and gather intelligence for threat detection and analysis. Python libraries like Beautiful Soup and Scrapy simplify the process of fetching web pages, parsing HTML, and extracting relevant information efficiently. Understanding web scraping empowers cybersecurity professionals to gather threat intelligence, monitor adversary activities, and enhance situational awareness in cyberspace. It is a valuable skill for building

tools that scrape websites responsibly, adhere to legal and ethical guidelines, and contribute to proactive cybersecurity measures.

You have learned from this unit that one effective way to collect data from websites automatically is through web scraping, which is provided by a number of Python packages, including Requests, ZenRows, Selenium, and Beautiful Soup. Different scraping requirements are met by different libraries, which can handle HTTP queries, parse HTML/XML documents, and handle dynamic information. Respecting website policies is essential to avoiding potential server overload and legal problems. Making effective use of these libraries guarantees data extraction that is simplified, increasing efficiency and reducing issues in web scraping applications.

6.0 Summary

Automated data extraction from websites is known as "web scraping," but it should only be done ethically and in compliance with the terms of service of the websites in question. Web scraping is made easier by a number of Python packages, such as Beautiful Soup, ZenRows, Selenium, and Requests. Each has advantages of its own, such the ease of use of Requests and the powerful parsing powers of Beautiful Soup. You may use Requests to send HTTP GET requests to a URL and get content and status codes by gaining access to the response object. Requests may also have custom headers. For example, if you send ReqBin a GET request with custom headers, it will provide response headers like Content-Type and Server details.

7.0 References/Further Reading

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.

UNIT 2: USING PYTHON FOR RECONNAISSANCE

Unit Structure

1.0 Introduction

In this unit, you will learn about using Python for Reconnaissance which involves Parsing HTML data, and extracting useful data from websites. Using Python for reconnaissance in cybersecurity involves leveraging its capabilities to gather information about systems, networks, and applications. Python scripts can automate the discovery and enumeration of targets by querying public databases, performing DNS lookups, scanning ports, extracting metadata from files, and identifying vulnerabilities. This reconnaissance phase is crucial for understanding the target environment, assessing potential attack surfaces, and gathering intelligence that informs subsequent steps in security assessments or penetration testing. Python's versatility and extensive library support make it a preferred choice for developing reconnaissance tools that aid in gathering essential information ethically and efficiently.

.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- parse HTML data from web pages
- extract useful data from websites

3.0  Content

3.1 Reconnaissance

The first step in the penetration Testing process is called reconnaissance, or just recon. Acquiring as much information as possible about the target is the aim of recon. The more information available, the more useful it will be for further stages of penetration testing. Although most inexperienced learners undervalue and overlook this stage, recon is the most crucial part of penetration testing. If you fully comprehend this process, your perspective on the digital world shifts. For everyone, mastering the recon process is an invaluable ability. There are two approaches to reconnaissance: active and passive.

1. Active recon is engaging the target directly in order to obtain information. Because it goes against the pen testing principle of "hiding traces," this is not advised.
2. Passive recon is the process of learning about a target by utilizing the wealth of online information. Since we aren't communicating with the target directly, we don't have to worry about them monitoring or logging our activities.

Reconnaissance can be achieved through Parsing HTML Data or by utilizing the wealth of online information.

In-Text Question
What are the two approaches to reconnaissance in penetration testing, and which one is recommended?

Answer
The two approaches to reconnaissance in penetration testing are active recon and passive recon. Active recon involves engaging the target directly to obtain information, which is not recommended because it goes against the principle of "hiding traces." Passive recon involves gathering information without directly interacting with the target, making it the recommended approach.

3.2 Parsing HTML Data

The first thing to do when parsing HTML is to read the file into python using the request library that we have studied in the previous unit. The code below is an example of how to use the request library to read HTML.

Reading a HTML File

```
importmatplotlib.pyplotasplt
importpandasaspd
importnumpyasnp
importscipyassp
importrequests

x = requests.get('https://w3schools.com')
print(x.text[0:2000])
```

The code fetches the webpage from its URL and stores the result in a response object called ‘x’ which has a text attribute that contained the same HTML code that we see when viewing the source from a browser. The output is given below:

Output

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<title>W3Schools Online Web Tutorials</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="Keywords" content="HTML, Python, CSS, SQL, JavaScript, How to, PHP, Java, C, C++, C#, jQuery, Bootstrap, Colors, W3.CSS, XML, MySQL, Icons, NodeJS, React, Graphics, Angular, R, AI, Git, Data Science, Code Game, Tutorials, Programming, Web Development, Training, Learning, Quiz, Exercises, Courses, Lessons, References, Examples, Learn to code, Source code, Demos, Tips, Website">
<meta name="Description" content="Well organised and easy to understand Web building tutorials with lots of examples of how to use HTML, CSS, JavaScript, SQL, Python, PHP, Bootstrap, Java, XML and more.">
<meta property="og:image" content="https://www.w3schools.com/images/w3schools_logo_436_2.png">
<meta property="og:image:type" content="image/png">
<meta property="og:image:width" content="436">
<meta property="og:image:height" content="228">
<meta property="og:description" content="W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many more.">
<link rel="icon" href="https://www.w3schools.com/favicon.ico">
```

```
<link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
<link rel="icon"type="image/png" sizes="32x32" href="/favicon-32x32.png">
<link rel="icon"type="image/png" sizes="16x16" href="/favicon-16x16.png">
<link rel="manifest" href="/site.webmanifest">
<link rel="mask-icon" href="/safari-pinned-tab.svg" color="#04aa6d">
<meta name="msapplication-TileColor" content="#00a300">
<meta name="theme-color" content="#ffffff">
<link rel="preload" href="/lib/fonts/fontawesome.woff2?14663396"as="font"type="font/woff2" crossorigin>
<link rel="preload" href="/lib/fonts/source-code-pro-v14-latin-regular.woff2"as="font"type="font/woff2" crossorigin>
```

Next we are going to pass the HTML using the beautiful soup library which is a popular python library for webscraping. This code parses the HTML stored in our text 'x' into a special object called soup that soup library understands.

Parsing HTML Code using Beautiful Soup

```
importmatplotlib.pyplotasplt
importpandasaspd
importnumpyasnp
importscipyassp
importrequests
x = requests.get('https://w3schools.com')
print(x.text[0:500])

frombs4import BeautifulSoup

soup = BeautifulSoup(x.text, "html.parser")

for child in soup.descendants:

if child.name:
print(child.name)
```

In other words beautiful soup is reading the HTML and making sense of structure.

Now, one can examine the patterns noticed in the formatting of the HTML document to extract a lot of information.

3.3 Extracting Useful Data from Websites

One can as well decide to extract only information from certain tags after parsing the HTML document. The code below given an example in which only the h2, P, li tags were printed to output.

Extracting Information from h2, P, li tags

```
importmatplotlib.pyplotasplt
importpandasaspd
importnumpyasnp
importscipyassp
importrequests

x = requests.get('https://w3schools.com')
print(x.text[0:1000])

frombs4import BeautifulSoup

soup = BeautifulSoup(x.text, "html.parser")
print(soup.h2)
print(soup.p)
print(soup.li)
```

Output

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<title>W3Schools Online Web Tutorials</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="Keywords" content="HTML, Python, CSS, SQL, JavaScript, How to, PHP, Java, C, C++, C#, jQuery, Bootstrap, Colors, W3.CSS, XML, MySQL, Icons, NodeJS, React, Graphics, Angular, R, AI, Git, Data Science, Code Game, Tutorials, Programming, Web Development, Training, Learning, Quiz, Exercises, Courses, Lessons, References, Examples, Learn to code, Source code, Demos, Tips, Website">
<meta name="Description" content="Well organised and easy to understand Web building tutorials with lots of examples of how to use HTML, CSS, JavaScript, SQL, Python, PHP, Bootstrap, Java, XML and more.">
<meta property="og:image" content="https://www.w3schools.com/images/w3schools_logo_436_2.png">
```

```
<meta property="og:image:type" content="image/png">
<meta property="og:image:width" content="436">
<meta property="og:image:he
<h2 style="color: #fff4a3"><b>Tutorials</b></h2>
<p class="tnb-services-headlines">
        W3Schools offers a wide range of services and products for
beginners and professionals,
<br/>
        helping millions of people everyday to learn and master new
skills.
</p>
<li><img src="/spaces/files/check3.db67d31e.svg" style="margin-
right:15px;"/>Browse W3Schools <strong>without ads</strong></li>
```

4.0 Self Assessment Exercise(S)

1. What are the are two approaches to reconnaissance

Answer
There are two approaches to reconnaissance: active and passive.\

1. Active recon is engaging the target directly in order to obtain information. Because it goes against the pen testing principle of "hiding traces," this is not advised.
2. Passive recon is the process of learning about a target by utilizing the wealth of online information. Since we aren't communicating with the target directly, we don't have to worry about them monitoring or logging our activities.

2. What is the first thing to do when parsing HTML data in Python?

Answer:
The first thing to do when parsing HTML is to read the file into python using the request library**.**

5.0 Conclusion

IN this unit, you have learnt that the first step in penetration testing is reconnaissance which entails learning as much as possible about the target. It is a crucial stage that combines active and passive strategies

and is frequently overlooked. One popular passive technique is to parse HTML data. This may be done by fetching the HTML information using the Requests library, parsing it with BeautifulSoup, and then using it for structured analysis. Finding and removing particular information from HTML tags, such as headers, paragraphs, and list items, is necessary to extract usable data from web pages. This information can offer insightful information for additional research or application. When data extraction is done right, it creates the foundation for targeted activities and well-informed decision-making during penetration testing projects.

6.0 Summary

The first step in penetration testing is reconnaissance, which is essential for learning more about the target. Both active and passive methods are possible; passive reconnaissance uses internet data to gather information without requiring direct interaction. Python's Requests package can be used to retrieve HTML content from a webpage so that it can be parsed later. A well-liked Python web scraping module called BeautifulSoup helps parse HTML and provides structured data for analysis. HTML tags like <h2>, <p>, and <li> can be used to extract specific information that provides insights into the structure and content of webpages.

7.0 References/Further Reading

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.

MODULE 4 LOG FILE PARSING

Module Introduction

In this module, you will build on previous knowledge, and also, you will learn about Log File Parsing.

The module is organised into three units as follows:

Unit 1 Introduction to Log File Parsing
Unit 2 Basics of Log File Structure
Unit 3 Reading Log Files in Python

UNIT 1 INTRODUCTION TO LOG FILE PARSING

Unit Structure

1.0 Introduction

In this unit, we will be learning about log files, why log file is important in cyber security and the advantages of using Python for parsing log files. Log file parsing using Python is essential in cybersecurity for extracting and analyzing valuable information from various log sources, such as system logs, web server logs, and application logs. Python scripts can be designed to parse log files line by line, extracting specific fields like timestamps, IP addresses, URLs, error codes, and user activities. This parsed data can then be further processed for anomaly detection, trend analysis, security incident investigation, or compliance auditing purposes. Python's string manipulation capabilities and regular expressions are particularly useful for efficiently parsing structured and

unstructured log formats, aiding cybersecurity professionals in understanding system behaviors, identifying security incidents, and maintaining system integrity.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- explain what is a log file
- state the importance of parsing a log file
- use python libraries for parsing log files

3.0 Main Content

3.1 What are Log Files?

The main source of data for network observability is log files. An operating system, application, server, or other device's utilization patterns, activities, and operations are documented in a log file, which is a computer-generated data file. Log files provide information on how well and efficiently resources are operating.

3.2 Overview of log files

Most popular operating systems are all specially set up to produce and classify event logs in response to particular kinds of events. Applications performance-related problems can be easily understood, tracked down, and resolved with the help of log management systems, which centralize all log files and collect, organise, and analyze log data. Figure 9 is an example of a log file.

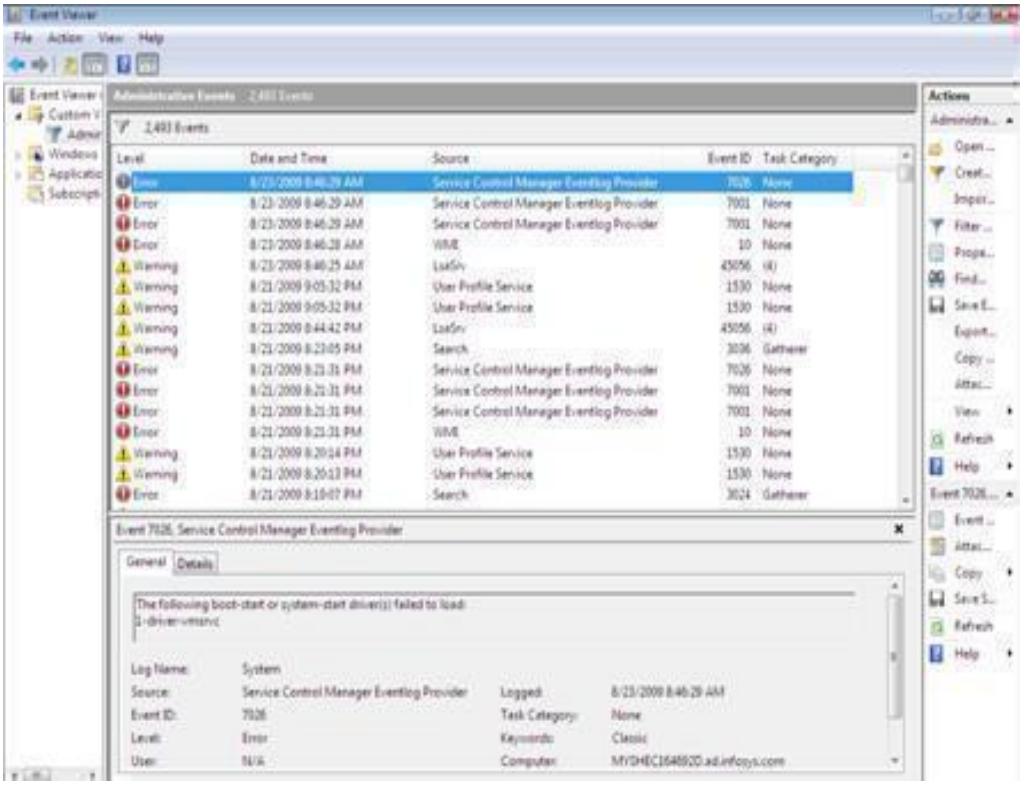


Figure 9: Sample Log File for Windows

Every log includes a list of the events that took place, along with issues, errors, and cautions. The Windows application, security, and system log files can be read with a Windows application called “Event Viewer,” which is accessed through the Control Panel.

3.3 Importance of Parsing Log Files

For a log management system to retrieve useful information, the files must first be parsed. The process of parsing logs allows your log management system to read, index, and store data from either unstructured or structured log files. In this manner, you may quickly filter, examine, and work with the key-value data. With log parsing, data from your logs is extracted and arranged into fields. It is nearly impossible to browse through and visualize your log data if your logs contain poorly organised fields.

3.4 Why Python for Log File Parsing?

Python is commonly used for log file parsing for two reasons:

1. Python is renowned for being easy to read and simple to use. Both inexperienced and seasoned programmers can use it because of its clear and simple syntax. This user-friendliness extends to log file processing as well, enabling developers to quickly construct scripts that extract pertinent data from logs.

2. Ample Library Resources: Python boasts a robust library and module ecosystem that makes log file parsing easier. Re (regular expressions), datetime, csv, json, and other libraries offer strong capabilities for working with a variety of log file types and extracting pertinent information.

In-Text Question
Why is Python commonly used for log file parsing?

Answer
Python is commonly used for log file parsing because of its easy-to-read and simple syntax, which makes it accessible to both inexperienced and seasoned programmers, and its robust library ecosystem, including modules like re, datetime, csv, and json, which offer strong capabilities for working with various log file types and extracting pertinent information.

3.5 Introduction to Python Libraries for Log Parsing

Python offers several libraries and modules specifically designed for parsing log files efficiently. This is an example on using the re (regular expressions) module to parse logs in Python.

Import Necessary Modules
1 **import** re 2 **from** datetime **import** datetime

This imports the datetime module to parse timestamps and the re module to work with regular expressions.

Define Log Parsing Function

def parse_log(log_file_path):
with open(log_file_path, 'r') **as** file:
for line **in** file:
# Define your regular expression pattern to match log entries 	
pattern =r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}) - (\w+): (.*)'
 	
# Match log entries with the pattern 	
match = re.match(pattern, line)
if match:
# Extract timestamp, log level, and message from the matched groups 	
timestamp_str, log_level, message = match.groups()
 	
# Convert timestamp string to datetime object 	

```
        timestamp = datetime.strptime(timestamp_str, '%Y-%m-%d %H:%M:%S')

# Process the extracted information (e.g., print or store it)
print(f'Timestamp: {timestamp}, Level: {log_level}, Message: {message}')
```

This Create the parse_log function, which accepts as input the path to a log file. Open the log file inside, then go over each line one by one. To match log entries and retrieve pertinent data, including timestamp, log level, and message, a regular expression is used. To make manipulation easier, the timestamp string is converted to a datetime object.

Call the Parsing Function

```
1 log_file_path = 'path/to/your/log/file.log'
2 parse_log(log_file_path)
```

This specifies the path to your log file and calls the parse_log function with the path as an argument.

4.0 Self-Assessment Exercise(S)

1. Why Python for Log File Parsing?

Answer:

Python is commonly used for log file parsing for two reasons:

- Python is renowned for being easy to read and simple to use. Both inexperienced and seasoned programmers can use it because of its clear and simple syntax. This user-friendliness extends to log file processing as well, enabling developers to quickly construct scripts that extract pertinent data from logs.
- Ample Library Resources: Python boasts a robust library and module ecosystem that makes log file parsing easier. Re (regular expressions), datetime, csv, json, and other libraries offer strong capabilities for working with a variety of log file types and extracting pertinent information.

2. What is the Importance of Parsing Log Files

Answer:
For a log management system to retrieve useful information, the files must first be parsed. The process of parsing logs allows your log management system to read, index, and store data from either unstructured or structured log files. In this manner, you may quickly filter, examine, and work with the key-value data. With log parsing, data from your logs is extracted and arranged into fields. It is nearly impossible to browse through and visualize your log data if your logs contain poorly organised fields.

Topics for Discussion: Log File Analysis and Parsing for Security Insights

5.0 Conclusion

To sum up, log files are essential data sources for network observability since they record system activity and resource use. To extract useful insights, these logs must be parsed efficiently. Python is a popular option for log file parsing because of its readability and robust library ecosystem, which allow developers to swiftly extract and handle pertinent data. Using Python modules such as datetime and regular expressions (re) makes parsing easier and makes it easier to extract messages, log levels, and timestamps from log entries.

6.0 Summary

Log files are essential for keeping an eye on system operations and resource usage in network observability. Python is a superior language for efficiently processing log files because of its ease of use and wide library support. Through the use of Python packages like datetime and regular expressions, developers may quickly extract pertinent data from log entries, including timestamps, log levels, and messages.

7.0 References/Further Reading

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming.*

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.

UNIT 2 BASICS OF LOG FILE STRUCTURE

Unit Structure

1.0 Introduction

In this unit, you will learn about log file formats such as CSV, JSON and plaintext. You will also learn about structured vs. unstructured log files and some key components of a log file such as timestamp, log level, message, etc. Understanding the basics of log file structure is crucial for effective log file parsing and analysis in cybersecurity. Log files typically consist of chronological records that capture events, actions, or transactions occurring within a system or application. Each log entry is often structured with standardized fields such as timestamps, severity levels, event IDs, user IDs, IP addresses, and descriptions of events. The structure may vary depending on the logging framework or application generating the logs, ranging from simple text-based formats to more complex structured data formats like JSON or XML. Knowledge of these structures enables cybersecurity analysts to parse, interpret, and extract meaningful insights from log data, helping to identify security incidents, troubleshoot system issues, and ensure compliance with regulatory requirements.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- explain the components of a log file
- discuss structured Vs unstructured log files
- identify some key components of a log file such as timestamp,

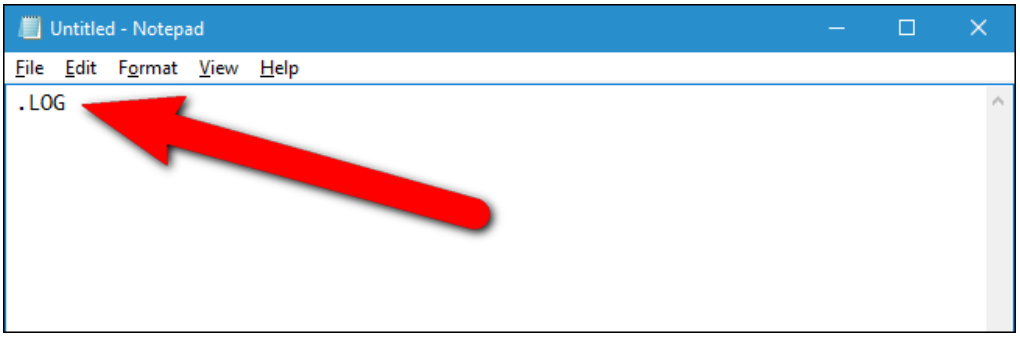
log level, message, etc

3.0 Content

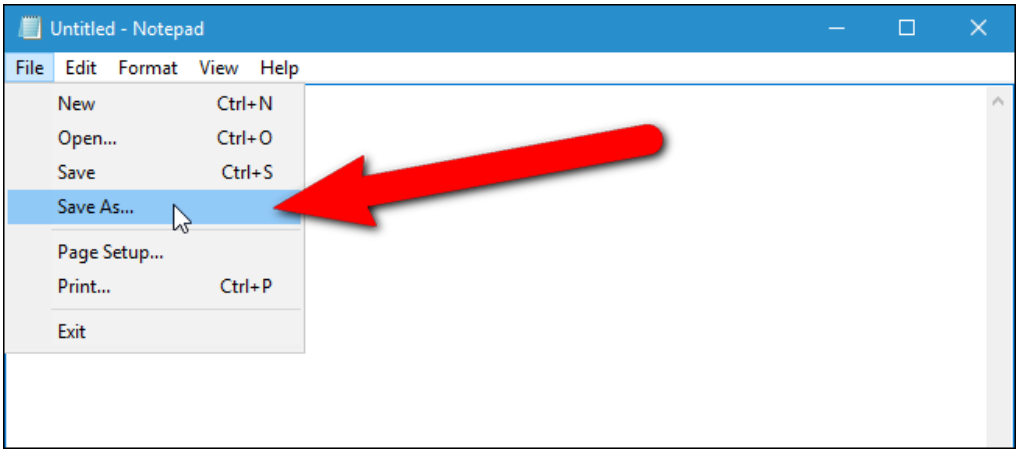
3.1 Creating a Simple Log File using Notepad

A log file keeps a record of events details, date and time. This is generally used by antivirus software, operating systems and other applications to track some unusual activity or errors.A simple log file can be created without any compiler. The steps include:

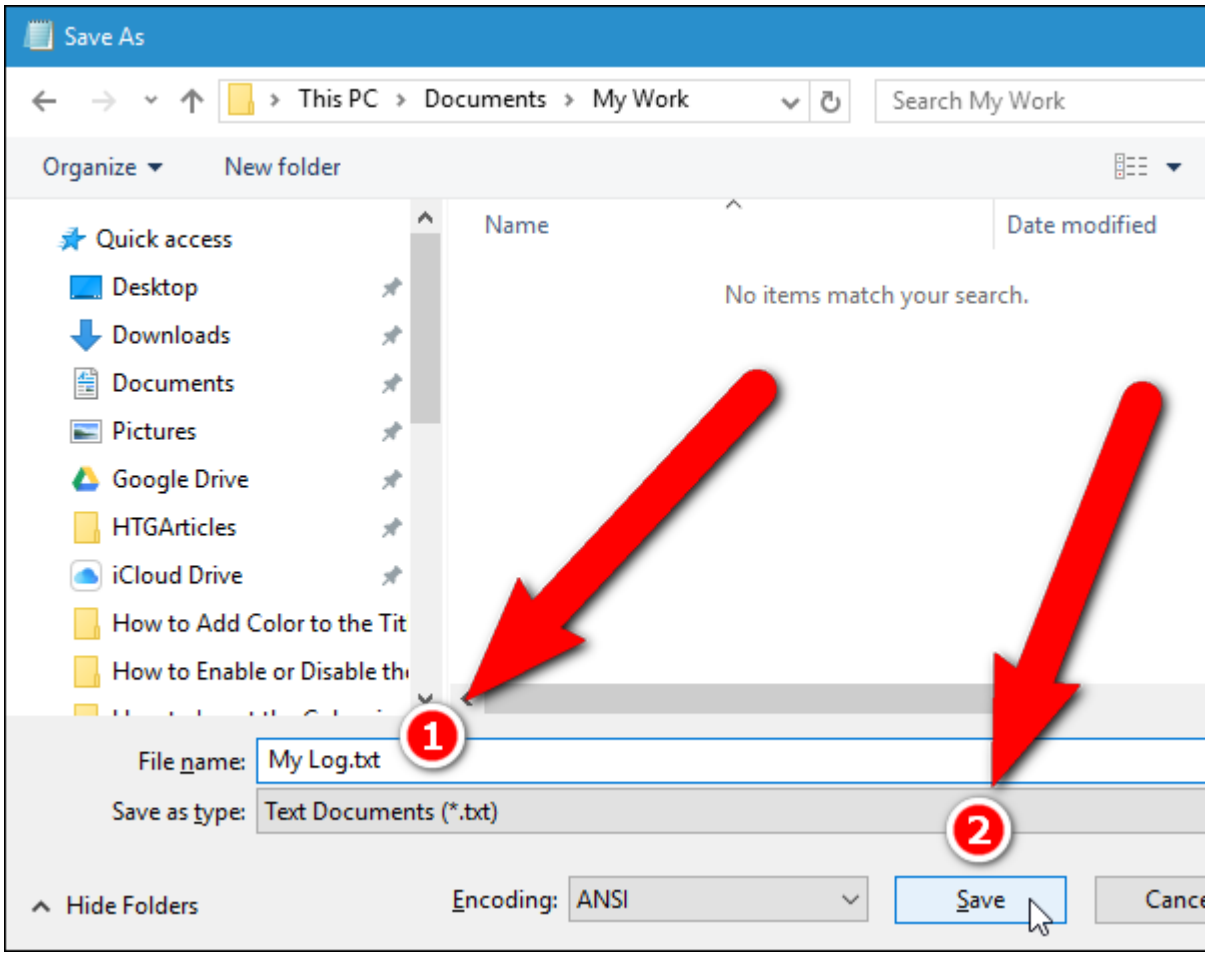
- Click on windows search bar
- Type notepad
- Click on the notepad app
- Type .LOG in the first line and press enter



- Click on the file menu and from the drop down list click the save as option



- Select destination on your computer to save the log file, type the name of your log file, Click on save button



- Close the notepad app

Now whenever you open your log file, it will be noticed that the date and time has been automatically added to your log file as shown in Figure 10.

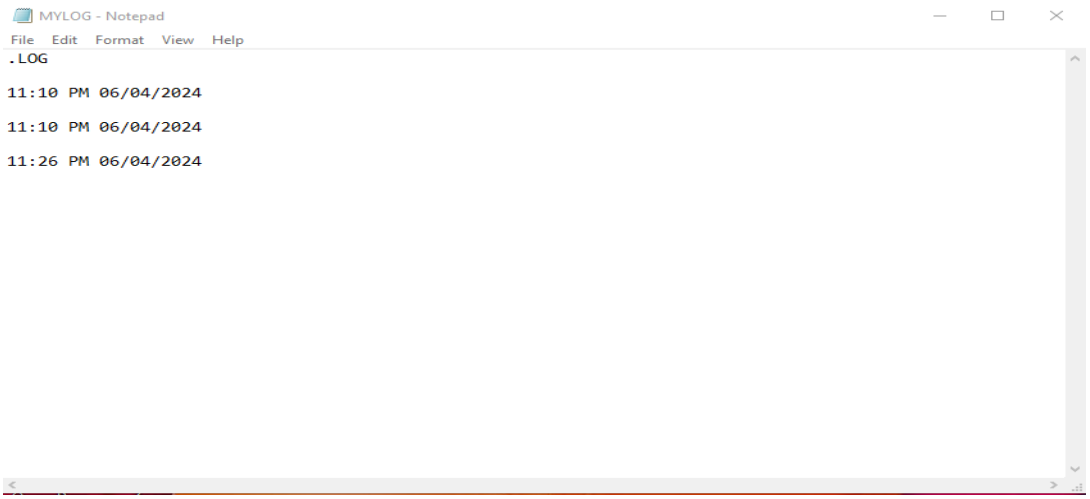


Figure 10: Simple Log File Created using Notepad

5.1 Common log file formats (e.g., CSV, JSON, plain text)

A log file's format is determined by the particular system or program that created it. JSON, CSV, and plain text files are a few prevalent log file types.

JSON Log File

One of the most widely used log formats is JavaScript Object Notation (JSON). JSON logs have several key-value pairs and are semi-structured. Logs can nest data into multiple layers using JSON, all the while maintaining a human-readable language. Data types including string, number, boolean, null/empty, object, and array can also be maintained with JSON. Figure 11 is an instance of a JSON log file:

```
{
  "timestamp": "2022-07-29T02:03:45.2932",
  "message": "User Jane.Doe has logged in",
  "log": {
    "level": "info",
    "file": "auth.c",
    "line": 66,
  },
  "user": {
    "name": "jane.doe",
    "id": 235
  },
  "event": {
    "success": true
  }
}
```


Fig. 11: A sample JSON Log File (Source: https://www.crowdstrike.com/cybersecurity-101/observability/log-file-formats/)

CSV Log File

A CSV summary log file contains one line of summary data for each specific import, export. For every database table handled during the delete procedure, one line of summary data is generated. The CSV LogFileDelimiter character is used to separate the summary data in each line of the file. Figure 12 is a snapshot of a sample CSV file.

	A	B	C	D
1	student_id	timestamp		
2	2014642038	02:01:31		
3	2014201242	02:01:45		
4	2014213192	02:02:11		
5				
6				
7				
8				
9				

Fig. 12: A snapshot of a sample CSV file
(Source: Yusof et.al., 2018)

3.3 Understanding structured vs. unstructured log files

There are three types of log files: semi-structured, unstructured, and structured. Both humans and machines can interpret structured log formats because they follow a logical, consistent pattern. Fields can occasionally be separated by a hyphen, space, or comma (as in CSV files). An equal (=) sign can also be used to link them (name=Jane, city=Paris, etc.).The majority of log management systems are equipped with pre-configured parsers that make it simple to ingest structured log formats. An example of an organised log file may be found Figure 13:

```
[{
  "Env" : "Prod",
  "ServerName" : "LAPTOP123",
  "AppName" : "Console1.vmhost.exe",
  "AppLoc" : "C:\Test\stackify-api-dotnet\dst\ConsoleApplication1\bin\Debug\Console1.vmhost.exe",
  "Logger" : "StackifyLib.net",
  "Msgs" : [{
    "Msg" : "Incoming metrics data",
    "data" : "{"clientid":12345}",
    "EpochMs" : 1445345672470,
    "Level" : "INFO",
    "id" : "0c12301b-e4ge-11e6-8933-897567896a4"
  }]
}]
```


Fig. 13: A Sample of a Structured Log File (Source:https://www.crowdstrike.com/cybersecurity-101/observability/log-file-formats/)

Unstructured log formats are straightforward for people to read even though they do not adhere to any certain pattern. As a result, splitting up the events and extracting key-value pairs during parsing becomes challenging. An unstructured log will require special parsing if the log management system does not have a built-in parser, which frequently adds to the engineer's workload.

Semi-structured logs are simple for people to read, but they also contain a schema or pattern that allows robots to interpret them as well. They do have a pattern, but their field and event separators are more intricate than those of a comma or an equal sign. Semi-structured logs can be ingested by log management systems, however splitting events and extracting key-value pairs typically calls for a parser. Usually, code or regular expressions are used for this.

3.4 Log File Components

An event that happened at a specific time is recorded in a log file, which may also contain contextual metadata. A system's log files include a

history of all events that have occurred therein, including transactions, errors and intrusions.

3.5 Identifying key components in log files (timestamp, log level, message, etc.)

Log files are organised records of messages or events produced by devices, systems, or software programs. They usually have a number of essential elements that aid in deciphering and interpreting the logged data. The following are a few of the most typical elements of log files:

Timestamp: This shows the exact moment a message or event was recorded. It aids in determining the events' chronological sequence and in the diagnosis of timing-related problems.

Log Level: This indicates how serious or significant the event that was logged was. INFO, WARNING, ERROR, FATAL, DEBUG, and others are common log levels. With DEBUG being the least severe and FATAL being the most severe, each level denotes the severity of the incident.

Message: This is the precise text that makes up the event or message that is being logged. It gives information about what transpired, including background. Depending on the system or application creating the logs as well as the particular event being captured, messages can differ significantly.

4.0 Self-Assessment Exercise(S)

1. State and explain three (3)key components in log files

Answer:
Timestamp: This shows the exact moment a message or event was recorded. It aids in determining the events' chronological sequence and in the diagnosis of timing-related problems.

Log Level: This indicates how serious or significant the event that was logged was. INFO, WARNING, ERROR, FATAL, DEBUG, and others are common log levels. With DEBUG being the least severe and FATAL being the most severe, each level denotes the severity of the incident.

Message: This is the precise text that makes up the event or message that is being logged. It gives information about what transpired,

including background. Depending on the system or application creating the logs as well as the particular event being captured, messages can differ significantly.

.

2. Differentiate between Unstructured and Semi-structured logs.

Answer:
Unstructured log formats are straightforward for people to read even though they do not adhere to any certain pattern. As a result, splitting up the events and extracting key-value pairs during parsing becomes challenging. An unstructured log will require special parsing if the log management system does not have a built-in parser, which frequently adds to the engineer's workload.

Semi-structured logs are simple for people to read, but they also contain a schema or pattern that allows robots to interpret them as well. They do have a pattern, but their field and event separators are more intricate than those of a comma or an equal sign. Semi-structured logs can be ingested by log management systems, however splitting events and extracting key-value pairs typically calls for a parser. Usually, code or regular expressions are used for this.

5.0 Conclusion

In conclusion, log files are essential records of messages and events that provide important information about how systems, applications, and networks behave and function. Common log file formats that serve a variety of purposes include JSON, CSV, and plain text. Each has its own structure and features. It is necessary to comprehend the differences between semi-structured, unstructured, and structured log formats in order to perform efficient log management and analysis. Timestamps, log levels, and messages are three essential log file components.

6.0 Summary

Log files are vital records that record messages and events for systems and applications, facilitating monitoring and troubleshooting. They are available in multiple formats, each with unique structures and applications, including JSON, CSV, and plain text.

7.0 References/Further Reading

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.

UNIT 3 READING LOG FILES IN PYTHON

Unit Structure

1.0 Introduction

In this unit, you will learn about opening and reading log files using Python's built-in open() function, iterating through lines in a log file and reading Structured Log Files. Reading log files in Python involves opening a file using Python's built-in functions or libraries like `open()` and `read()`. Log files, typically structured text files, contain chronological records of events or actions within a system. Python provides various methods to read and parse these files, such as iterating line by line to extract relevant information like timestamps, event IDs, and messages. This process is essential in cybersecurity for analyzing system behavior, detecting anomalies, and identifying security incidents. Using Python's file handling capabilities, cybersecurity professionals can efficiently process and interpret log data to enhance system security and maintain operational integrity.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, you will be able to:

- open and read different types of log files
- iterate through lines in a log file
- read structured log files

3.0 Main Content

3.1 Opening and Reading Log Files using Python's Built-in open() Function

Create a new Python file in your code editor and give it the name files.py.
We must first find a method to link a file on disk to a Python variable in order to open the file in Python. This variable is known as a file handle, and the procedure is known as opening a file. First, we tell Python the location of the file. The file path, which in this case is path = '/Users/Halima/Desktop/test.txt', is the location of your file. To store this path information, create a variable as shown below

```
Opening a File
1 path = '/Users/Halima/Desktop/test.txt'
```

Now, you can open our days.txt file with Python's open() function. The file path must be supplied as the first argument to the open() function. Many more parameters are also accepted by the function. The optional mode parameter, though, is the most crucial. The mode in which the file is opened is indicated by this optional string. What you want to do with the file will determine which mode you select. Some of the possible modes are as follows:

- 'r' : use for reading from a file
- 'w' : use for writing to a file
- 'a' : use for appending to a file
- 'r+' : use for reading and writing to the same file

We shall utilize the 'r' mode in the the code below to enable us to read from the file. The test.txt file is opened using the open() method, and the file handle that is obtained is assigned to the variable test_file.

```
1 test_file = open(path, 'r')
```

After opening the file, the following step will guide you through reading its contents.

3.2 Reading a Log File

Now that our file has been opened, we may work with it (read from it) by using the variable to which it was allocated. Let us examine the use if a read() function in python in more detail now.

- Making use of read

The full contents of the file are returned as a single string by read(). As an illustration is shown in the code below

Using the read()
1 test_file.read()

Output
'This is for testing'

The output is the entire content of the test.txt file.

3.3 Writing to a Log File

Python also provides that flexibility for writing to a file. Let us examine the use if a write() function in python in more detail using the code below.

Write() fuction
1 path = '/Users/Halima/Desktop/test.txt' 2 new_path = '/Users/Halima/Desktop/test1.txt' 3 title = 'Wrting to a File**n**' 4 **with**open(path, 'r') **as** test_file, open(new_path, 'w') **as** test1: 5 days = test_file.read() 6 test1.write(title) 7 test1.write (days) 8 **print**(title) 9 **print**(days)

4.0 Self-Assessment Exercise(s)

1. What mode should you use if you want to read from a file using Python's open() function?

A. 'w'

B. 'a'

C. 'r+'

D. 'r'
Answer: D

2. What does the read() method do when applied to a file handle in Python?

A. Reads a single line from the file

B. Reads the entire contents of the file as a single string

C. Writes data to the file

D. Appends data to the end of the file

Answer: B. Reads the entire contents of the file as a single string

3. Given the code snippet below, what will the variable days contain?

```
with open(path, 'r') as test_file:
        days = test_file.read()
```

A. The path to the file

B. The file handle

C. The contents of the file as a single string

D. The first line of the file

Answer: C. The contents of the file as a single string

5.0 Conclusion

We have covered the necessary topics for managing log files using Python in this unit. We began by using the open() function to open and read log files. We next learned how to connect files to Python variables and investigated other file access modes, such writing and reading.

6.0 Summary

The file handling features of Python for working with log files are introduced in this unit. It covers using the open() function and different modes to open, read, and write to log files. It shows how to use it practically through examples, making it possible to manipulate log data in Python programs effectively.

7.0 References/Further Readings

https://www.digitalocean.com/community/tutorials/how-to-handle-plain-text-files-in-python-3

Matthes, E. (2019). Python Crash Course: A Hands-On. *Project-Based Introduction to Programming*.

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. no starch press.

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell*. " O'Reilly Media, Inc.

Seitz, J., & Arnold, T. (2021). *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press.

O'Connor, T. J. (2012). *Violent Python: a cookbook for hackers, forensic analysts, penetration testers and security engineers*. Newnes.