MINT-709: CAPSTONE PROJECT

DESIGN AND IMPLEMENT MINT-708 CONFIGURATION AND TROUBLESHOOT EXAM IN EVE-NG AND AUTOMATE MARKING

Supervised By:

Shahnawaz Mir

Program Coordinator, MINT

Submitted By:

Deepansha Gaur

Student ID: 1684148

Contents

INTRODUCTION

Exam paper marking has always been time-consuming and takes a lot of work. By automating the marking of structured-type questions, we hope to solve this problem. A web-based system has been created to gather answers, mark them, and give students feedback. The final course grades that lecturers give out are meant to reflect each student's degree of accomplishment.

Numerous judgments are based on these grades. An adequate amount of accuracy and efficiency is needed in the marking and grading process to ensure the grades are likely to convey accurate information and mislead the decision-maker. Exam script marking is laborious, especially when done manually and with many pupils. It takes a long time and is prone to mistakes. However, using a computer and the appropriate software package makes the process simpler and more accurate. Finding a means to carry out this marking process that is both quick and accurate enough is a complex but intriguing topic. An electronic marking script is created in this work. The program is developed in Python and is run on a Linux-based server. The designed program operated efficiently and delivers the desired outcomes when finished. It enables automatic exam marking and score grading based on student responses that match the right ones that had been added to the system's data collection. A computer-based automated marking system automatically grades tests, assignments, and other sorts of assessments. Algorithms and statistical models evaluate student responses and provide grades or comments.

The approach can aid in lessening instructors' workloads and improving the effectiveness of the grading process. It will enable the teacher to grade the exam using the students' configuration. In contrast to conventional marking, it can be carried out using a local program or a browser-based application sent over the Internet, in our case, using a Linux machine and browser.

The MINT-708 exam is based on the EVE-NG server, a multi-vendor graphical network emulation server that allows us to build network topologies for testing and development quickly. MINT-708 involves configuring multi-vendor network devices like Juniper, Cisco, Alcatel, etc., physically making the connections, and logically configuring the device. Additionally, we can configure IP addresses, routes, routing protocols, etc. All this is done using the EVE-NG server, wherein the users can remotely access the server and configure the devices accordingly. EVE-NG is part of MINT-708 and MINT-720 for the complete lab practice, assignments, and final lab exam.

BACKGROUND OF STUDY

The manual marking system is riddled with issues. A quick glance at the manual marking system reveals instances of partiality, favoritism, and injustice on the part of the staff or the class instructor during the marking process. In addition, outside of the situations, a person may be experiencing physical or emotional trauma, which could prevent them from performing their duties in a safe manner. Additionally, mistakes may occur during the manual computation of results because of "omission" or "commission". Exam scripts going missing while being marked is another issue. Time wastage is one issue among many others.

Additionally, a system for efficiently and effectively keeping electronic exam scripts is created, allowing for online testing, instantaneous results computation, and publication, as well as the storage of exam data in a central database for documentation, planning, and evaluation reasons in the future. The personnel will use the time that would have been used to prepare results and grade exams to conduct their research. The less severe pupils will be motivated to focus on their studies.

It is not only desirable but necessary to fully adopt a computerized approach possible when assessing students' academic progress due to the inaccuracies connected with the current manual method of marking student exams used by most universities. The currently used manual procedures have several drawbacks. They delay the publication of exam-marked results while also making the process lengthy and prone to errors. Even worse, incorrect scores are occasionally calculated for and entered. As a result, the issuance of an incorrect degree class is eventually tied to the cumulative errors generated. Some students would receive unjustly high grades or degrees, while others might suffer unfair victimization, which would frustrate them. Finding a way of marking exam scripts that would be both sufficiently accurate and suitably timely becomes the challenge.

STATEMENT OF PROBLEM

Manual techniques of marking exam scripts provide significant issues. Exam script processing is slowed down by manual marking because it must be done by hand. This has led to academic students delaying their outcomes. To stop this issue, there is a need to increase the usage of electronic marking systems that are precise, error-free, and, if possible, run in real-time online so that students may view their scores whenever they need to. Additionally, it will speed up the process of receiving exam script results.

AUTOMATION OF MARKING

1. BENEFITS

The benefits of an automated marking system include the following:

- Increased efficiency: Automated grading can save instructors time by allowing them to grade numerous submissions swiftly and appropriately while saving on labor.
- Objective grading: Automated grading ensures that each student receives an impartial and uniform grade by removing the risk of human bias.
- Feedback: Students can receive fast feedback from automated grading, helping them to recognize their errors and enhance their performance.
- Data analysis: Automated grading can gather information on student performance, giving teachers the tools to spot problem areas and make informed choices.

Additionally, the implementation of an automated marking system requires careful planning and consideration of several important factors:

- Accuracy: It's essential to ensure that the system accurately grades submissions based on predefined criteria. This may require fine-tuning the algorithms and models used by the system.
- Flexibility: The system should be flexible enough to accommodate different types of assignments, tests, and exams and adapt to changing requirements.
- Security: The system should ensure the confidentiality and security of student submissions, especially in cases involving sensitive information.
- User-friendliness: The system should be user-friendly, allowing instructors and students to use it easily. A user-friendly interface and clear instructions can help increase adoption and reduce confusion.

Integration with existing systems: The system should integrate well with existing educational systems, such as learning management systems or student information systems, to reduce administrative overhead and provide a seamless experience for users.

In conclusion, a Python-based automated marking system can assist educational institutions, but it's crucial to thoroughly analyze the requirements and constraints before adopting it. Institutions can make sure the system is precise, adaptable, secure, user-friendly, and well-integrated with current systems by carefully planning the implementation.

2. WHAT PROBLEM DOES IT SOLVE?

Automated marking systems solve several problems that exist in traditional manual grading methods, including:

- Time-consuming: Manual grading can be labor- and time-intensive, mainly when there are many entries.
- Automated grading can facilitate grading and free up the time of the instructor.
- Inconsistency and human error: Manual grading is more likely to contain mistakes and inconsistencies when numerous instructors are involved in the grading process.
- Automated grading ensures that each student receives an accurate and uniform grade by removing the chance of human error.
- Lack of immediate feedback: When grades are manually graded, students frequently have to wait for them, which can hinder their capacity to learn from their errors and perform better. Students can receive immediate feedback via automated grading.
- Limited data analysis: Collecting information on student performance through manual grading can be challenging. Automated grading can generate a plethora of information about student performance, enabling teachers to pinpoint the areas in which their pupils need additional help and to make informed choices.
- Bias: Whether deliberate or not, bias in human beings can affect manual grading.

Automated grading ensures that each student receives an objective and fair grade by removing the risk of human bias.

Automated marking systems address issues with manual grading that are time-consuming, inconsistent, and error-prone, as well as issues with restricted data analysis, human bias, and the absence of fast feedback.

3. LIMITATIONS

It's crucial to remember that automated grading systems may have flaws, such as difficulties evaluating open-ended questions or identifying handwritten responses. Therefore, before building the system, it is crucial to carefully assess the use case and ensure it is appropriate for the task. Several limitations and restrictions include:

- Difficulty grading open-ended questions: Open-ended questions that call for human interpretation and judgment, like essays or creative writing projects, may be challenging for automated marking systems to grade.
- Limitations in recognizing handwritten answers: Handwritten replies may be challenging for automated marking systems to identify, mainly if the handwriting needs to be legible or irregular.
- Dependence on predefined criteria: The quality of automated marking systems depends on the standards that are used to evaluate students. The caliber and clarity of the requirements and the algorithms utilized to evaluate them affect how accurately the grades are assigned.
- Potential for false positive or negative results: The quality of automated marking systems depends on the standards used to evaluate students. The caliber and clarity of the criteria and the algorithms utilized to evaluate them affect how accurately the grades are assigned.
- Dependence on technology: Automated marking systems rely on technology and are susceptible to technical issues like network outages or software failures.
- Limited ability to deal with complex situations: Automated marking systems rely on technology and are susceptible to technical issues like network outages or software failures.
- Resistance from instructors and students: Some instructors and students may need more clarification regarding the fairness and accuracy of automated marking systems. It can be challenging to get through this reluctance and increase adoption.

In conclusion, even though automated marking systems have many advantages, they also have drawbacks. It's crucial to carefully analyze these restrictions and select a suitable option that satisfies the requirements of the educational setting and the students.

EVE-NG SERVER

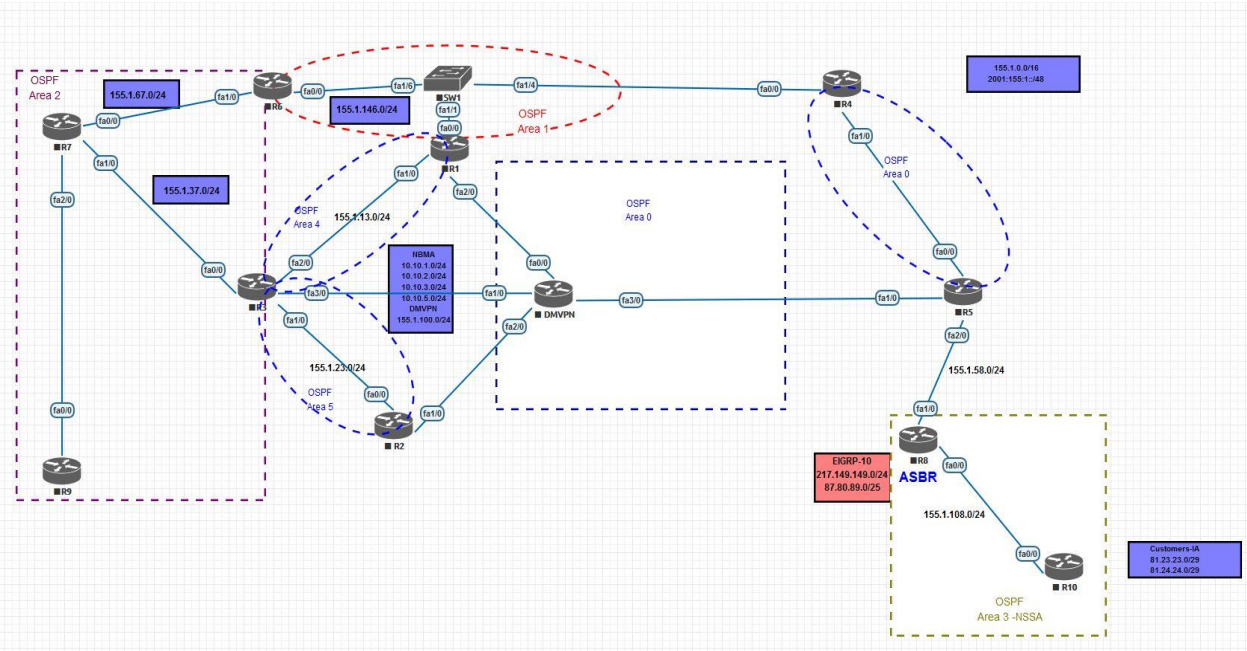
EVE-NG (Emulated Virtualization Engine Next Generation) is an open-source network emulator software used for network simulation and testing. It provides a virtual environment that allows network engineers and administrators to build, configure, and test network topologies without the need for physical hardware. EVE-NG supports a wide range of network devices, including routers, switches, firewalls, and security appliances, and it supports various network protocols such as OSPF, BGP, and ISIS.

EVE-NG is designed to be scalable, allowing users to build large-scale network simulations with multiple devices. It also provides a web-based user interface that makes managing and monitoring virtual networks easy. Additionally, EVE-NG supports integration with other software, such as GNS3, allowing users to create complex network scenarios and simulate real-world network environments. [1]

In conclusion, EVE-NG is a powerful and versatile network emulator software that offers many benefits for network engineers and administrators, including the ability to build, configure, and test network topologies in a virtual environment, without the need for physical hardware.

For MINT-708, EVE-NG is part of the labs and final exam. Since it is a multi-vendor server, it allows us to setup labs, including Cisco, Juniper, and Alcatel devices.

Below is an example of the same:



PYTHON

Python is a high-level, interpreted programming language widely used for web development, scientific computing, data analysis, artificial intelligence, and more. It was first released in 1991 and is currently on its 3rd major version, Python 3, first released in 2008. Python 3 has improved support for Unicode, simplified syntax, and memory management compared to Python.

Benefits of using Python:

- Easy to learn and use: Python has a simple syntax that makes it easy to learn and use, especially for beginners.
- Extensive community and libraries: Python has a massive community of developers who have created various libraries and packages that can be used for multiple purposes.
- Versatile: Python can be used for various applications, including web development, scientific computing, data analysis, artificial intelligence, and more.
- Cross-platform compatibility: Python can run on various platforms, such as Windows, MacOS, and Linux, making it a highly portable language.
- Dynamic typing: In Python, you don't need to declare variable types explicitly, which makes it easier to write code quickly.
- Interpreter: Python is an interpreted language, meaning the code can be executed line by line, making it easier to debug and test your code.

Developers, data scientists, and researchers frequently utilize Python because it is a well-liked, adaptable, and potent programming language. Some everyday use cases for automation using Python include:

- Web scraping: Python can be used to automate the process of gathering data from websites and web applications, allowing you to extract large amounts of data quickly and efficiently.
- System administration: Python scripts can be used to automate tasks such as system monitoring, log analysis, and software deployment, reducing the amount of manual effort required to manage complex systems.
- Data processing: Python is well-suited for tasks such as data cleaning, transformation, and analysis and can be used to automate these tasks to speed up workflows and improve data quality.
- Testing and QA: Python scripts can be used to automate testing and quality assurance tasks, such as unit testing, integration testing, and performance testing, helping to improve software quality and reduce time to market.

An automated marking system for assessing and grading student work, such as assignments or exams, can be made using Python. The system can employ algorithms to evaluate student responses and award a grade based on predetermined standards. Here are the fundamental actions to develop such a system:

- Read the answers: Read the answers from a file or database and store them in a file
- Preprocess the answers: Preprocess the answers to remove any irrelevant information, such as punctuation, whitespaces, and special characters, and convert them to a standard format for easier comparison.
- Compare answers: Compare the student answers to the correct answers and assign a score based on the similarity between the two. Different algorithms can be used for this, such as exact match, find etc.
- Store the results: Store the results in a file or database for future reference and analysis.
- Evaluate and grade: Evaluate the results and assign a grade to each student based on the score they received. A grading scale can be defined beforehand to determine the grade for each score range.

This is merely a brief introduction to creating an automated marking system in Python. The specifics will depend on the specifications and system complexity, but Python's adaptability and libraries make it a good choice for the job.

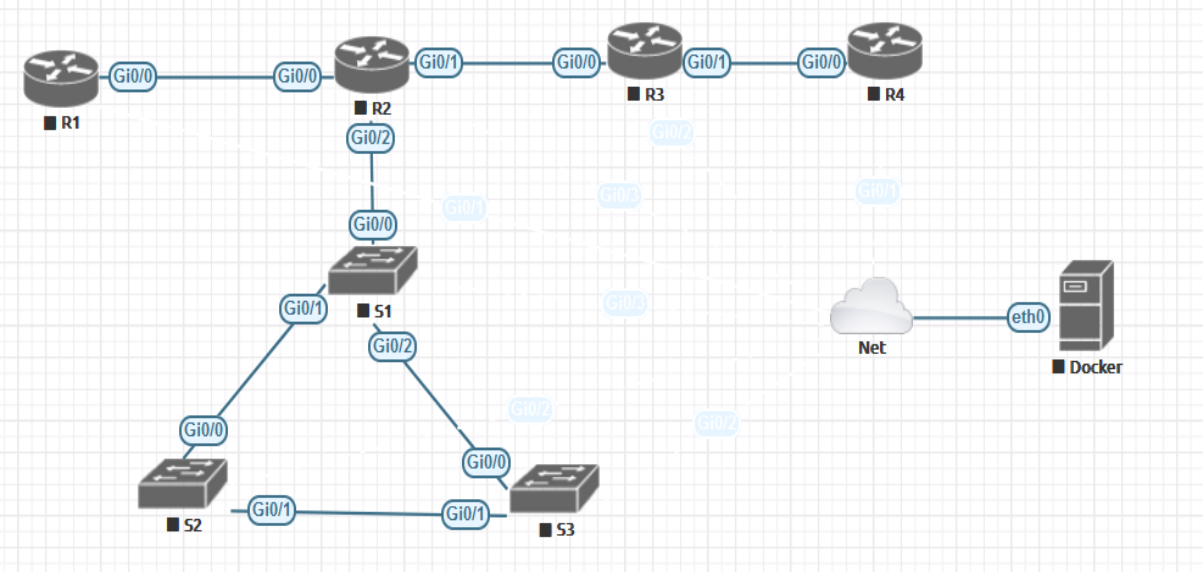
For our script, we're using Python3. Python 3 is the latest version of the Python programming language. It was released in 2008 and is the successor to Python 2.

DESIGN

The exam will consist of a series of tasks that the candidate must complete within a specified time frame. The tasks will include configuring various network devices, such as routers and switches. The exam will consist of two parts:

- Troubleshoot Exam: The candidate is presented with a simulated network environment with various configuration issues that must be resolved. The candidate must identify and fix the problems within a specified time frame.
- Configuration Exam: A networking configuration exam evaluates a candidate's ability to configure various network devices, such as routers and switches. The exam typically consists of a series of tasks that the candidate must complete within a specified time frame. The duties may include configuring network interfaces, setting up routing protocols, configuring NAT/PAT access control lists (ACLs), etc. The exam is conducted in a simulated environment using a network emulation tool, EVE-NG.

The instructor sets these exams. Four sets of exams have different questions. The script can be modified according to each exam. Below is one such example:



Management Cloud0 Interface

Switches, routers, servers, and other network equipment are managed and monitored on a separate management network. The management network is used for operations like device configuration, monitoring, troubleshooting, and software updates. It is often segregated from the leading production network to ensure security and avoid disturbances from management traffic. This minimizes the risk of security breaches or assaults on the management network and ensures that management traffic does not obstruct production traffic. A dedicated network segment or VLAN can be used to build up the management network.

Some benefits of having a management network include the following:

- Improved security: Separating management traffic from production traffic reduces the risk of security breaches and attacks.
- Enhanced visibility: The management network provides greater visibility into the performance and status of network devices, allowing administrators to identify and resolve issues more quickly.
- Centralized management: A dedicated management network allows administrators to manage network devices from a central location, reducing the need for physical access to individual devices.
- Easier troubleshooting: The management network allows administrators to remotely troubleshoot network issues, reducing the need for on-site visits and minimizing downtime.

In summary, a management network is a separate network for managing and monitoring network devices. It provides improved security, visibility, centralized management, and easier troubleshooting.

The Cloud0 network for labs is another name for the EVE management interface. Our EVE's first NIC is bridged with the Cloud0 interface. To gain administration access to nodes running inside EVE from a host machine outside of EVE, Cloud0 is frequently used in EVE labs.

A specific IP address range or subnet enables network administrators to configure and control network devices. All the devices in the exam will be connected to a Linux-based server, and the devices will be managed using a management network which in our case is **11.0.0.0/24**.

Default Configuration on the devices

The default configuration refers to the preconfigured settings and options with a system or application out of the box. These settings are usually set by the manufacturer or developer and are intended to provide a starting point for administrators to configure the system to their specific needs. One of the benefits of having a default configuration is that it can help administrators get started quickly with using a new system or application. Additionally, the default configuration often includes best practices and recommended settings that can help ensure that the system is secure and stable from the outset. However, it's important to note that the default configuration may not be suitable for all use cases and may need to be modified to meet specific requirements.

In our scenario, the devices will have the following default configurations:

1. Management Interface

   A management interface is a software component that allows users to interact with a system or application to perform administrative tasks. Management interfaces provide a graphical user interface (GUI) or command-line interface (CLI) that allows users to configure, monitor, and manage various aspects of the system or application. Some common features of management interfaces include:

   - User authentication and access control
   - Configuration of system settings and options
   - Monitoring of system status and performance
   - Logging and reporting of system events.
   - Remote access and management capabilities

   The management interface will be preconfigured with an IP address from the 11.0.0.0/24 range. The candidates will be instructed not to make any changes to the management interface. Example of one such configuration:

```
interface GigabitEthernet0/0
 description MGMT
 ip address 11.0.0.1 255.255.255.0
 duplex auto
 speed auto
 media-type rj45
```

1. Telnet

Telnet is a network protocol used to provide remote access to networking devices such as routers, switches, and firewalls. In Cisco devices, Telnet can be used to remotely access the device's command-line interface (CLI) and perform configuration and management tasks. To enable Telnet on a Cisco device, follow these steps:

- Enter privileged EXEC mode by entering the enable command and providing the privileged EXEC password if prompted.
- Enter global configuration mode by entering the configure terminal command.
- Enter the line configuration mode for the VTY lines you want to enable Telnet on.
- Enter the transport input telnet command to enable Telnet access on the VTY line.
- (Optional) Set a Telnet password by entering the password command followed by the desired password.
- (Optional) Set a login message by entering the login message command followed by the desired message.
- Exit line configuration mode by entering the exit command.
- Save the configuration by entering the write memory command.

Once Telnet is enabled, we can remotely connect to the device's CLI by using a Telnet client and specifying the device's IP address or hostname and the VTY line number you want to connect to. For example, to connect a router with IP 11.0.0.1, you can use the following command from a command prompt or terminal: **telnet 11.0.0.1**

You will then be prompted for the Telnet password (if set), and then you can access the device's CLI. Example of one such configuration:

```
line vty 0 4
 password adminpwd12
 login local
 transport input telnet
```

AUTOMATION SCRIPT

The script has been divided into four parts:

1. Pull “show running-config” output from all the devices and store it in a file.
2. Remove the special characters from the output and divide the output into distinct file for every device.
3. Compare the output with the correct configuration.
4. Execute all the above parts and display results.

PART 1: “show running-config” output

WORKING:

We will input the list of management IPs using a list. In Python, a list is a collection of items that are ordered and changeable. Lists are denoted by square brackets [] and each element in a list is separated by a comma. Example:

HOST = ["11.0.0.1", "11.0.0.2", "11.0.0.3", "11.0.0.4", "11.0.0.5", "11.0.0.6", "11.0.0.7", "11.0.0.8", "11.0.0.9", "11.0.0.10"]

To retrieve the “show running-config” output from the devices, we’re using the **telnetlib**

TELNETLIB

‘telnetlib’ is a built-in module in Python that provides a simple way to establish a Telnet connection with a remote host and interact with it. It implements the Telnet protocol and allows you to send and receive data over a Telnet session. To use telnetlib, you first create a Telnet object by specifying the hostname or IP address of the remote host, and optionally the port number, timeout value, and other parameters. Once you have a Telnet object, you can use its methods to interact with the remote host.

With telnetlib, you can establish a connection to a remote server and interact with it programmatically. Some of the common use cases for telnetlib include:

- Automating remote server administration tasks
- Testing network connectivity and diagnosing network issues
- Building custom Telnet-based applications

```
import telnetlib

# create a Telnet object
tn = telnetlib.Telnet("localhost", 23)

# send a command to the Telnet server
tn.write(b"ls\n")

# read the output from the Telnet server
output = tn.read_all()

# close the Telnet connection
tn.close()
```

In this example, the tn variable is used to create a Telnet object that connects to a local Telnet server on port 23. The write() method is then used to send a command to the Telnet server, and the read_all() method is used to read the output from the server. Finally, the close() method is used to close the Telnet connection.

In our case, we use the below methods:

1. *tn.open() :* The open() method is used to establish a Telnet connection to a remote host using a Telnet object in the telnetlib module in Python.
2. *tn.read_until() :* The read_until() method in the telnetlib module is used to read data from the Telnet connection until a specific byte sequence is found or until a timeout occurs.
3. *tn.write() :* The write() method in the telnetlib module is used to send data over a Telnet connection. The write() method takes a byte string as an argument and sends it to the Telnet server.
4. *tn.read_all() :* The read_all() method in the telnetlib module is used to read all the data available from the Telnet connection. It blocks until the Telnet connection is closed or until a timeout occurs.
5. *tn.close() :* The close() method in the telnetlib module is used to close the Telnet connection that was established with the Telnet object. Once the connection is closed, no further data can be sent or received.

Once the output is retrieved, it is stored in a file.

PART 2: File Creation

WORKING:

We create different files for every device config which will later be compared with the correct config. For this, we'll use a bash script.

1. *Tr*

   To remove special characters from a string in a Bash script, you can use the tr command with the -d option to delete the characters you want to remove.

2. *Csplit*

   csplit is a command-line utility in Unix and Linux systems that is used to split a file into multiple smaller files based on a specified pattern. It is particularly useful for breaking up large log files or other types of text files into smaller, more manageable pieces.

PART 3: Output Comparison

The instructor will have a correct configuration which will be used to compare with the candidate's configuration.

The script will open each router's configuration file and will check if the tasks have been done correctly. In Python, you can work with files using the built-in open() function. The open() function returns a file object, which you can use to read from or write to the file.

1. *Opening a file: To open a file, use the open() function and specify the file name and mode (read, write, or append). For example, to open a file for reading:*
   f = open('filename.txt', 'r')

2. *Closing a file: When you're done working with a file, it's important to close it using the close() method of the file object. For example:*
   f.close()

WORKING:

All the tasks will be assigned a default value of 0. When the config line is matched, the counter is increased by 1. For the comparison, we're using the below modules:

1. *Itertools (islice)*

itertools is a Python module that provides a set of tools for working with iterators, which are objects that can be iterated over (i.e., processed one at a time) to produce a sequence of values.

islice(iterable, start, stop, step=1): returns an iterator that generates a slice of the elements in iterable, starting from start, up to but not including stop, and skipping by step at each iteration.

The functions and others provided by itertools can be very useful for working with large or complex sequences of data in a memory-efficient way, or for performing complex data processing operations that involve iterating over multiple sequences simultaneously. [2]

2. *IPAddress*

Python's IPaddress module offers classes for interacting with networks, subnets, and IP addresses. It enables operations like subnetting, supernetting, and network merging, as well as the manipulation and validation of IPv4 and IPv6 addresses.

The `ipaddress` module provides a number of useful methods for working with IP addresses and networks. For example, it allows for the parsing of IP addresses and subnets from strings using the `ip_address()` and `ip_network()` functions. It also provides methods for checking if an IP address or subnet is private, if two IP addresses or subnets overlap, and for calculating the number of hosts in a subnet. [3]

Below is an example:

```
if line.startswith('interface GigabitEthernet0/3'):
    output1 = list (islice(R1,n))
    m=output1[0].split(' ')
    if ipaddress.ip_address(m[3]) in
    ipaddress.ip_network('192.168.12.0/24'):
        ipaddr=ipaddr+1
```

PART 4: Display results

Once the script completes running, it will create an HTML file that will display the results of the students.

To display results in an HTML file, you can use the HTML module in Python to generate HTML code and write it to a file.

```
with open ("file.html","w") as file:
    file.write("<html")
    file.write("<head></head>")
    file.write("<body><p>IP Addressing: " + str(ipaddr) + "</p></body>")
    file.write("<body><p>RIP: " + str(rip) + "</p></body>")
    file.write("<body><p>NAT: " + str(nat) + "</p></body>")
    file.write("<body><p>ISIS: " + str(isis) + "</p></body>")
    file.write("<body><p>OSPF: " + str(ospf) + "</p></body>")
    file.write("<body><p>BGP: " + str(bgp) + "</p></body>")
    file.write("<body><p>STP: " + str(stp) + "</p></body>")
    file.write("<body><p>TOTAL: " + str(total) + "</p></body>")

    file.write("</html>")
```

In this example, the with open() statement opens a file named configuration_exam_results.html for writing. Then, the HTML code is generated and written to the file using the write() method. After running the code, a new file named results.html is created on the Desktop. When the candidates open this file in a web browser, they should see the results. The results will show the total marks and the marks received in each task. Example:

Name and Student ID: Student1_123456

Task 1: 17

Task 2: 6

Task 3 3

Task 4: 5

Task 5: 8

Task 6: 2

Task 7: 16

TOTAL: 57

We can setup a cronjob of the above script to run the corresponding order. Once the exam time finishes, the HTML file will be automatically created, and results can be displayed.

CONCLUSION

In conclusion, our project has successfully developed an automated grading system using Python. The system takes in student submissions and compares them to a set of expected answers, using a combination of string manipulation and other Python tools to analyze and evaluate the submissions. Overall, our automated grading system offers several advantages over traditional manual grading methods. It is faster and more efficient, reducing the workload on instructors and enabling faster turnaround times for grading. It is also more consistent and objective, reducing the potential for bias or variability in grading.

While our system is currently limited to grading text-based submissions, it could be expanded to include support for grading other types of media, such as images or audio files. It could also be integrated with learning management systems or other educational platforms to streamline the grading process even further.

In conclusion, our automated grading system offers a powerful and flexible solution for grading student submissions and can potentially revolutionize the way that grading is done in the education industry.

References

[1] "EVE-NG Cookbook," [Online]. Available: https://www.eve-ng.net/index.php/documentation/community-cookbook/.

[2] "Itertools Module," [Online]. Available: https://docs.python.org/3/library/itertools.html.

[3] "IP Address Module," [Online]. Available: https://docs.python.org/3/library/ipaddress.html.