Top of Form

Bottom of Form

**Portfolio Activity: Update a file through a Python algorithm**



**Activity Overview**



In this activity, you will create a new portfolio document to demonstrate your experience using Python to develop algorithms that involve opening files and parsing their contents. You can add this document to your cybersecurity portfolio, which you can share with prospective employers or recruiters. To review the importance of building a professional portfolio and options for creating your portfolio, read [Create a cybersecurity portfolio](https://www.coursera.org/learn/automate-cybersecurity-tasks-with-python/resources/Xku4r).

To create your portfolio document, you will review a scenario and follow a series of steps. This scenario is connected to the [Create another algorithm](https://www.coursera.org/learn/automate-cybersecurity-tasks-with-python/ungradedLab/aeLOb/activity-create-another-algorithm) lab that you have just completed. You will explain the code you developed in that lab, and this will help you prepare for future job interviews and other steps in the hiring process.

Be sure to complete this activity before moving on. The next course item will provide you with a completed exemplar to compare to your own work.

**Scenario**



Review the following scenario. Then complete the step-by-step instructions.

You are a security professional working at a health care company. As part of your job, you're required to regularly update a file that identifies the employees who can access restricted content. The contents of the file are based on who is working with personal patient records. Employees are restricted access based on their IP address. There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list.

Your task is to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, you should remove those IP addresses from the file containing the allow list.

**Note:** This scenario involves developing the same algorithm that is developed in Tasks 2-7 of the [Create another algorithm](https://www.coursera.org/learn/automate-cybersecurity-tasks-with-python/ungradedLab/aeLOb/activity-create-another-algorithm) lab. (You do not need to reference Task 1 and Tasks 8-10 of the lab to complete this portfolio activity.) You should revisit the lab to get screenshots to include in your portfolio document.

**Step-By-Step Instructions**



Follow the instructions to complete each step of the activity. Then, answer the 9 questions at the end of the activity before going to the next course item to compare your work to a completed exemplar.

**Step 1: Access the template**

# Algorithm for file updates in Python

## Project description

[Describe the scenario in your own words.]

## Open the file that contains the allow list

[Add content here.]

## Read the file contents

[Add content here.]

## Convert the string into a list

[Add content here.]

## Iterate through the remove list

[Add content here.]

## Remove IP addresses that are on the remove list

[Add content here.]

## Update the file with the revised list of IP addresses

[Add content here.]

## Summary

Refer to summary section.

Step 2: Access supporting materials

The following supporting material will help you complete this activity. The document **Instructions for including Python code** provides instructions and best practices for including samples of Python code in your portfolio activity. Keep it open as you proceed to the next steps.

To use the supporting material for this course item, click the link and select *Use Template*.

# Instructions for including Python code

It’s strongly recommended that you return to the[**Create another algorithm**](https://www.coursera.org/learn/automate-cybersecurity-tasks-with-python/ungradedLab/aeLOb/activity-create-another-algorithm)lab to include screenshots of your Python code. If this isn’t possible, you can also type your commands in this document.

When including samples of Python code, consider the following:

* There are multiple applications that can be used to take screenshots. If you’re unsure how to take a screenshot on your device, you can perform an online search such as “how to take a screenshot in [your operating system].” Replace “[your operating system]” with the operating system your computer uses.
* Do not include screenshots of the directions that are found in markdown cells. Only include screenshots of the code cells.
* If you type your code, highlight it in gray and use a monospaced font, such as in this example: ip\_addresses = ip\_addresses.split()

Step 3: Open the file that contains the allow list

The file that you want to open is called **"allow\_list.txt"**. Assign a string containing this file name to the **import\_file** variable. Then, use a **with** statement to open it. Use the variable **file** to store the file while you work with it inside the **with** statement.

Describe the Python syntax, functions, and keywords you need to accomplish this in the **Open the file that contains the allow list** section of the **Algorithm for file updates in Python** template. In the **Task 2** section of **Create another algorithm** lab, take a screenshot of this portion of your code. Or, type this code directly into the template.

Step 4: Read the file contents

Next, use the **.read()** method to convert the contents of the allow list file into a string so that you can read them. Store this string in a variable called **ip\_addresses**.

Describe the Python syntax, functions, and keywords you need to accomplish this in the **Read the file contents** section of the **Algorithm for file updates in Python** template. In the **Task 3** section of the **Create another algorithm** lab, take a screenshot of this portion of your code. Or, type this code directly into the template.

Step 5: Convert the string into a list

In order to remove individual IP addresses from the allow list, the IP addresses need to be in a list format. Therefore, use the **.split()** method to convert the **ip\_addresses** string into a list.

Describe the Python syntax, functions, and keywords you need to accomplish this in the **Convert the string into a list** section of the **Algorithm for file updates in Python** template. In the **Task 4** section of the **Create another algorithm** lab, take a screenshot of this portion of your code. Or, type this code directly into the template.

Step 6: Iterate through the remove list

A second list called **remove\_list** contains all of the IP addresses that should be removed from the **ip\_addresses** list. Set up the header of a **for** loop that will iterate through the **remove\_list**. Use **element** as the loop variable.

Describe the Python syntax, functions, and keywords you need to accomplish this in the **Iterate through the remove list** section of the **Algorithm for file updates in Python** template. In the **Task 5** section of the **Create another algorithm** lab, take a screenshot of this portion of your code. Or, type this code directly into the template.

Step 7: Remove IP addresses that are on the remove list

In the body of your iterative statement, add code that will remove all the IP addresses from the allow list that are also on the remove list. First, create a conditional that evaluates if the loop variable **element** is part of the **ip\_addresses** list. Then, within that conditional, apply the **.remove()** method to the **ip\_addresses** list and remove the IP addresses identified in the loop variable **element**.

Describe the Python syntax, functions, and keywords you need to accomplish this in the **Remove IP addresses that are on the remove list** section of the **Algorithm for file updates in Python** template. In the **Task 6** section of the **Create another algorithm** lab, take a screenshot of this portion of your code. Or, type this code directly into the template.

In addition, include a sentence that explains that applying the **.remove()** method in this way is possible because there are no duplicates in the **ip\_addresses** list.

Step 8: Update the file with the revised list of IP addresses

Now that you have removed these IP addresses from the **ip\_address** variable, you can complete the algorithm by updating the file with this revised list. To do this, you must first convert the **ip\_addresses** list back into a string using the **.join()** method. Apply **.join()** to the string **"\n"** in order to separate the elements in the file by placing them on a new line.

Then, use another **with** statement and the **.write()** method to write over the file assigned to the **import\_file** variable.

Describe the Python syntax, functions, and keywords you need to accomplish this in the **Update the file with the revised list of IP addresses** section of the **Algorithm for file updates in Python** template. In the **Task 7** section of the **Create another algorithm** lab, take a screenshot of this portion of your code. Or, type this code directly into the template.

Step 9: Finalize your document

To finalize the document and make its purpose clear to potential employers, be sure to complete the **Project description** and **Summary** sections of the **Algorithm for file updates in Python** template.

In the Project description section, give a general overview of the scenario and what you accomplished in Python. Write three to five sentences.

In the Summary section, provide a short summary of the algorithm by highlighting its main components. Write four to six sentences.

**Pro Tip: Save a copy of your work**

Finally, be sure to save a copy of your completed activity. You can use it for your professional portfolio to demonstrate your knowledge and/or experience to potential employers.

**What to Include in Your Response**



Be sure to address the following in your completed activity:

* Screenshots of your Python code or typed versions of the code
* Explanations of the syntax, functions, and keywords in the code
* A project description at the beginning
* A summary at the end
* Details on using a **with** statement and the **open()** function in your algorithm
* Details on using the **.read()** and **.write()** methods in your algorithm
* Details on using the **.split()** method in your algorithm
* Details on using a **for** loop in your algorithm
* Details on using the **.remove()** method in your algorithm

**Project Description:**

As a cybersecurity professional at a healthcare company, my task was to create a Python algorithm for regularly updating a file that manages access to restricted content based on IP addresses. The file, named "allow\_list.txt," contains IP addresses permitted to access the restricted subnetwork. The algorithm involves checking the allow list against a remove list to identify and remove any IP addresses that should no longer have access due to changes in employee roles or permissions.

**Algorithm for File Updates in Python Template:**

Step 1: Access the template:

Refer to the template

# Algorithm for file updates in Python

## Project description

The project description outlines that the individual is a security professional working at a healthcare company. The task involves regularly updating a file that manages access to restricted content based on IP addresses. It mentions the existence of an allow list and a remove list, both crucial in determining which employees can access restricted subnetworks. The project description serves as an introduction, giving the reader an understanding of the context and purpose of the algorithm before delving into the specific steps and code implementation.

Top of Form

Bottom of Form

## Open the file that contains the allow list

Import\_file = “allow\_list.txt”

With open(import\_file, “r”)as file:

# file variable stores the opened file for further processing

## Read the file contents

Ip\_addresses = file.read()

. syntax/functions/keywords: ‘read()’

## Convert the string into a list

ip\_addresses = ip\_addresses.split()

.Syntax/function/keyword: ‘split()’

## Iterate through the remove list

Remove\_list = [“192.168.97.225”, “192.168.158.170”, “192.168.201.40”]

A list of IP addresses to be removed for element in remove\_list:

[“192.168.97.225”, “192.168.158.170”, “192.168.201.40”]

. Syntax/function/keyword: ‘for’

## Remove IP addresses that are on the remove list

For element in remove\_list:

If element in ip\_addresses:

Ip\_addresses.remove(element)

. Syntax/functions/keyboards: ‘if’, ‘remove()’

## Update the file with the revised list of IP addresses

Updated\_content = “\n”.join(ip\_addresses)

With open(import\_file, “w”) as file:

File.write(updated\_content)

. Syntax/functions/keywords: ‘join()’, ‘write()’

## Summary

[The algorithm opens the "allow\_list.txt" file, reads its contents, converts the string into a list of IP addresses, iterates through the remove list to identify and remove relevant IP addresses, and finally updates the file with the revised list. The process ensures that employees are granted or denied access based on changes in the allow and remove lists.

This Python script offers a secure and efficient way to manage access to sensitive patient records within a healthcare company, demonstrating my proficiency in file handling and algorithm development.]

Step 2: Access supporting material

# Instructions for including Python code

It’s strongly recommended that you return to the[**Create another algorithm**](https://www.coursera.org/learn/automate-cybersecurity-tasks-with-python/ungradedLab/aeLOb/activity-create-another-algorithm)lab to include screenshots of your Python code. If this isn’t possible, you can also type your commands in this document.

When including samples of Python code, consider the following:

* There are multiple applications that can be used to take screenshots. If you’re unsure how to take a screenshot on your device, you can perform an online search such as “how to take a screenshot in [your operating system].” Replace “[your operating system]” with the operating system your computer uses.
* Do not include screenshots of the directions that are found in markdown cells. Only include screenshots of the code cells.
* If you type your code, highlight it in gray and use a monospaced font, such as in this example: ip\_addresses = ip\_addresses.split()

Step 3: Open the file that contains the allow list

Import\_file = “allow\_list.txt”

With open(import\_file, “r”) as file:

[“192.168.97.225”, “192.168.158.170”, “192.168.201.40”]

Import\_file = “allow\_list.txt”

With open(import\_file, “r”)as file:

# file variable stores the opened file for further processing

Step 4: Read the file contents

Ip\_addresses = file.read()

. syntax/functions/keywords: ‘read()’

Step 5: Convert the string into a list

ip\_addresses = ip\_addresses.split()

.Syntax/function/keyword: ‘split()’

Step 6: Iterate through the remove list

Remove\_list = [“192.168.97.225”, “192.168.158.170”, “192.168.201.40”]

A list of IP addresses to be removed for element in remove\_list:

[“192.168.97.225”, “192.168.158.170”, “192.168.201.40”]

. Syntax/function/keyword: ‘for’

Step 7: Remove IP addresses that are on the remove list

For element in remove\_list:

If element in ip\_addresses:

Ip\_addresses.remove(element)

. Syntax/functions/keyboards: ‘if’, ‘remove()’

Step 8: Update the file with revised list of IP addresses

Updated\_content = “\n”.join(ip\_addresses)

With open(import\_file, “w”) as file:

File.write(updated\_content)

. Syntax/functions/keywords: ‘join()’, ‘write()’

Step 9: Finalize your document

Summary:

The algorithm opens the "allow\_list.txt" file, reads its contents, converts the string into a list of IP addresses, iterates through the remove list to identify and remove relevant IP addresses, and finally updates the file with the revised list. The process ensures that employees are granted or denied access based on changes in the allow and remove lists.

This Python script offers a secure and efficient way to manage access to sensitive patient records within a healthcare company, demonstrating my proficiency in file handling and algorithm development.

Top of Form

Bottom of Form