

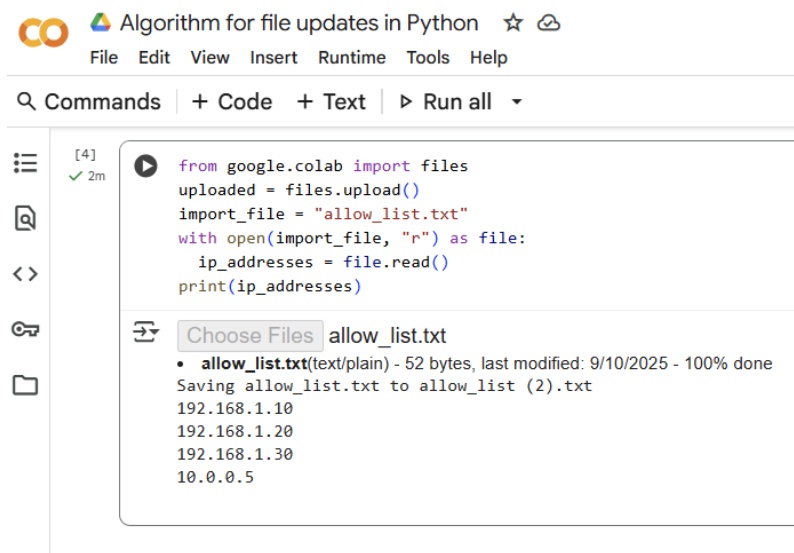
# Algorithm for file updates in Python

## Project description

As a security professional in a health care organization, I needed a way to efficiently update a file that lists which employees by, IP address, can access restricted content. The goal was to develop a Python algorithm that checks the allow list for IP address found on a separate remove list and removes them if necessary. Automating this task helps keep the access list up-to-date and secure, making sure only authorized employees are allowed access.

## Open the file that contains the allow list

To access the allow list, I used Python's `with` statement and the `open ( )` function. The `with` statement is important because it handles closing the file automatically, even if an error occurs. The `open ( )` function takes two arguments: the filename and the mode. Here, I used `"r"` to open the file in read mode.



The screenshot shows a Google Colab notebook interface. At the top, the title is "Algorithm for file updates in Python". Below the title bar, there are tabs for "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The "Commands" tab is active, showing a search bar and buttons for "+ Code", "+ Text", and "Run all". The notebook content area shows a code cell with the following Python code:

```
[4] ✓ 2m
from google.colab import files
uploaded = files.upload()
import_file = "allow_list.txt"
with open(import_file, "r") as file:
    ip_addresses = file.read()
    print(ip_addresses)
```

Below the code cell, there is a file upload section titled "Choose Files" with a button labeled "allow\_list.txt". Below this, there is a list of files:

- **allow\_list.txt**(text/plain) - 52 bytes, last modified: 9/10/2025 - 100% done

Below the list, there is a message: "Saving allow\_list.txt to allow\_list (2).txt". Below this, there is a list of IP addresses:

```
192.168.1.10
192.168.1.20
192.168.1.30
10.0.0.5
```

## Read the file contents

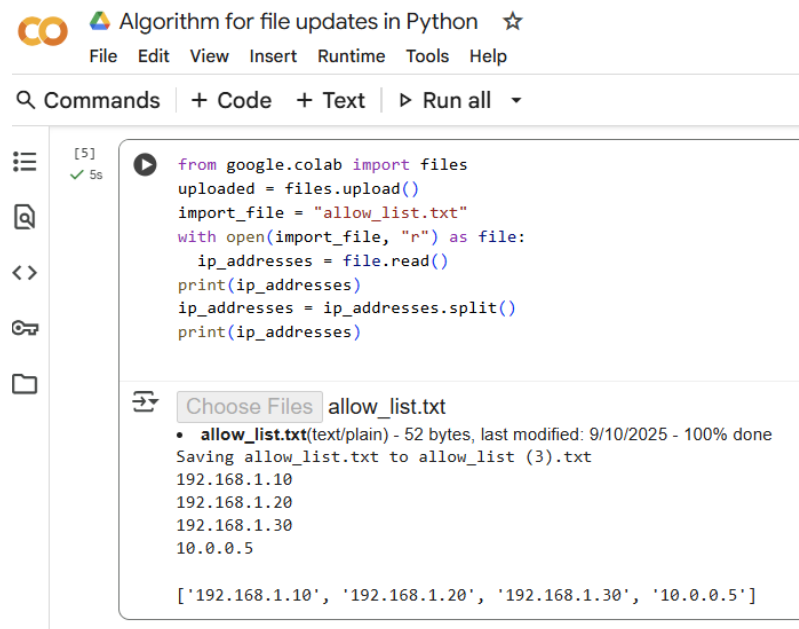
To read the content of the file, I used the `.read ( )` method on the `file` object, which returns the file's contents as a single string.

```
ip_addresses = file.read()
```

## Convert the string into a list

Since the IP addresses are stored as a single string, I used the `.split()` method to turn that string into a list. This allows me to iterate over each IP address individually.

```
ip_addresses = ip_addresses.split()
```



The screenshot shows a Google Colab notebook interface. At the top, there's a header with the Colab logo, the text "Algorithm for file updates in Python", and a star icon. Below this is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A search bar contains "Commands", and there are buttons for "+ Code", "+ Text", and "Run all". The main area shows a code cell with the following Python code:

```
from google.colab import files
uploaded = files.upload()
import_file = "allow_list.txt"
with open(import_file, "r") as file:
    ip_addresses = file.read()
print(ip_addresses)
ip_addresses = ip_addresses.split()
print(ip_addresses)
```

Below the code cell, there's a file upload section titled "Choose Files" with a button "allow\_list.txt". It shows a list of files: "allow\_list.txt(text/plain) - 52 bytes, last modified: 9/10/2025 - 100% done". Below this, it says "Saving allow\_list.txt to allow\_list (3).txt". The file contents are displayed as a list of IP addresses: "192.168.1.10", "192.168.1.20", "192.168.1.30", and "10.0.0.5". At the bottom, the output of the code is shown as a list: `['192.168.1.10', '192.168.1.20', '192.168.1.30', '10.0.0.5']`.

## Iterate through the remove list

I then looped through each element in the `remove_list` using a `for` loop. This makes it easy to check each IP address that needs to be removed.

```
for element in remove_list:
    # check if element is in ip_addresses
```

```
[7] ✓ 5s
from google.colab import files
uploaded = files.upload()
import_file = "allow_list.txt"
with open(import_file, "r") as file:
    ip_addresses = file.read()
print(ip_addresses)
ip_addresses = ip_addresses.split()
print(ip_addresses)
remove_list = ["192.168.1.20", "10.0.0.5"]

for element in remove_list:
    if element in ip_addresses:
        ip_addresses.remove(element)

print(ip_addresses)
remove_list = ["192.168.1.20", "10.0.0.5"]

for element in remove_list:
    if element in ip_addresses:
        ip_addresses.remove(element)

print(ip_addresses) # should show ['192.168.1.10', '192.168.1.30']
```

Choose Files allow\_list.txt

- **allow\_list.txt**(text/plain) - 52 bytes, last modified: 9/10/2025 - 100% done

Saving allow\_list.txt to allow\_list (5).txt

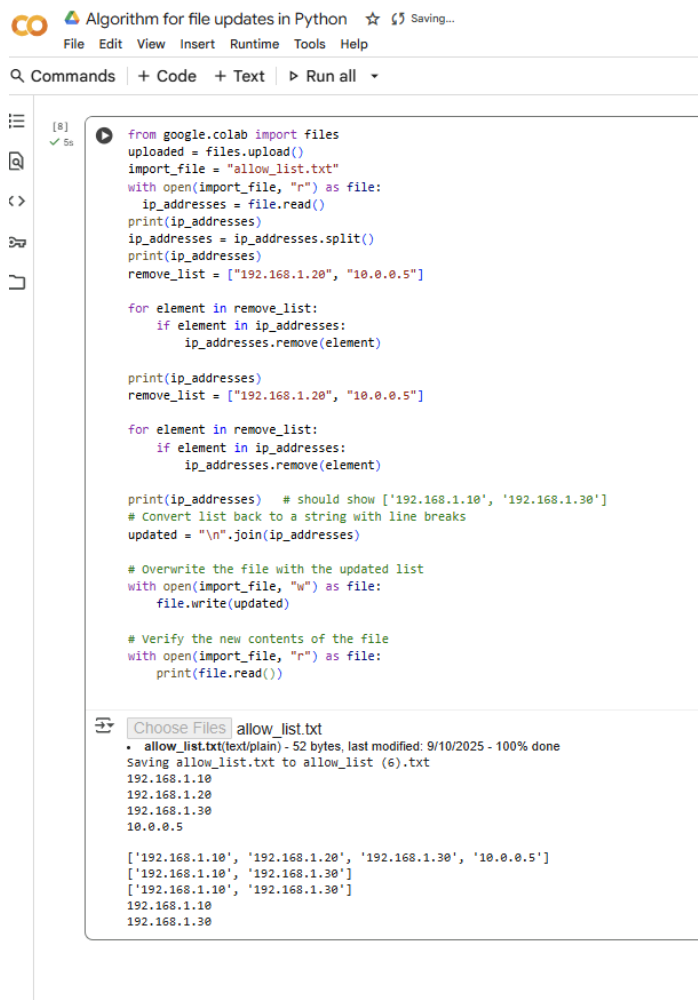
```
192.168.1.10
192.168.1.20
192.168.1.30
10.0.0.5

['192.168.1.10', '192.168.1.20', '192.168.1.30', '10.0.0.5']
['192.168.1.10', '192.168.1.30']
['192.168.1.10', '192.168.1.30']
```

## Remove IP addresses that are on the remove list

Inside the loop, I used a conditional statement to see if the current IP address (`element`) exists in the allow list (`ip_addresses`). If it does not, I removed it with the `.remove( )` method. This method works here because we know the allow list has no duplicates.

```
if element in ip_addresses:
    ip_addresses.remove(element)
```



The screenshot shows a Google Colab notebook interface. At the top, there's a header with the Colab logo, the text "Algorithm for file updates in Python", a star icon, and "Saving...". Below this is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A "Commands" bar is also present with options like "+ Code", "+ Text", and "Run all". The main area contains a code cell with the following Python code:

```
[8] ✓ 5s
from google.colab import files
uploaded = files.upload()
import_file = "allow_list.txt"
with open(import_file, "r") as file:
    ip_addresses = file.read()
    print(ip_addresses)
ip_addresses = ip_addresses.split()
print(ip_addresses)
remove_list = ["192.168.1.20", "10.0.0.5"]

for element in remove_list:
    if element in ip_addresses:
        ip_addresses.remove(element)

print(ip_addresses)
remove_list = ["192.168.1.20", "10.0.0.5"]

for element in remove_list:
    if element in ip_addresses:
        ip_addresses.remove(element)

print(ip_addresses) # should show ['192.168.1.10', '192.168.1.30']
# Convert list back to a string with line breaks
updated = "\n".join(ip_addresses)

# Overwrite the file with the updated list
with open(import_file, "w") as file:
    file.write(updated)

# Verify the new contents of the file
with open(import_file, "r") as file:
    print(file.read())
```

Below the code cell, there's a "Choose Files" section for "allow\_list.txt". It shows a list of files: "allow\_list.txt(text/plain) - 52 bytes, last modified: 9/10/2025 - 100% done". Below this, it says "Saving allow\_list.txt to allow\_list (6).txt". The file contents are displayed as follows:

```
192.168.1.10
192.168.1.20
192.168.1.30
10.0.0.5

['192.168.1.10', '192.168.1.20', '192.168.1.30', '10.0.0.5']
['192.168.1.10', '192.168.1.30']
['192.168.1.10', '192.168.1.30']
192.168.1.10
192.168.1.30
```

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

After updating the list, I used the `.join()` method to convert the list back into a string. I separated each IP address by a newline character (`"\n"`), so each address appears on a new line in the file. Then, I opened the file again in write mode (`"w"`) and wrote the updated string back to the file using the `.write()` method.

```
ip_addresses = "\n".join(ip_addresses)
with open(import_file, "w") as file:
    file.write(ip_addresses)
```

```

from google.colab import files
uploaded = files.upload()
import_file = "allow_list.txt"
with open(import_file, "r") as file:
    ip_addresses = file.read()
    print(ip_addresses)
    ip_addresses = ip_addresses.split()
    print(ip_addresses)
    remove_list = ["192.168.1.20", "10.0.0.5"]

    for element in remove_list:
        if element in ip_addresses:
            ip_addresses.remove(element)

    print(ip_addresses)
    remove_list = ["192.168.1.20", "10.0.0.5"]

    for element in remove_list:
        if element in ip_addresses:
            ip_addresses.remove(element)

    print(ip_addresses)  # should show ["192.168.1.10", "192.168.1.30"]
    # Convert list back to a string with line breaks
    updated = "\n".join(ip_addresses)

    # Overwrite the file with the updated list
    with open(import_file, "w") as file:
        file.write(updated)

    # Verify the new contents of the file
    with open(import_file, "r") as file:
        print(file.read())
    def update_file(import_file, remove_list):
        # Step 1: Read the file
        with open(import_file, "r") as file:
            ip_addresses = file.read()

        # Step 2: Split into list
        ip_addresses = ip_addresses.split()

        # Step 3: Remove matches
        for element in remove_list:
            if element in ip_addresses:
                ip_addresses.remove(element)

        # Step 4: Write updated list back
        updated = "\n".join(ip_addresses)
        with open(import_file, "w") as file:
            file.write(updated)

        # Verify updated contents
        with open(import_file, "r") as file:
            print(file.read())

    # Test it
    update_file("allow_list.txt", ["192.168.1.20", "10.0.0.5"])

```

Choose Files allow\_list.txt

```

• allow_list.txt (500B) - 52 bytes, last modified 9/10/2025 - 100% done
Saving allow_list.txt to allow_list (7).txt
192.168.1.10
192.168.1.30
["192.168.1.10", "192.168.1.30"]
["192.168.1.10", "192.168.1.30"]
["192.168.1.10", "192.168.1.30"]
192.168.1.10
192.168.1.30
192.168.1.10
192.168.1.30

```

## Summary

This algorithm streamlines the process of updating an allow list for restricted content by:

- Opening the file containing IP addresses.
- Reading its contents and converting them into a list.
- Iterating through a separate remove list and removing any matching IP addresses from the allow list.
- Writing the updated list back to the file, ensuring the allow list always reflects current access permissions.

Using Python for this process makes the task efficient, repeatable, and less prone to human error. The logic can easily be adapted for any similar file update or access-control task in a cybersecurity setting.