# Algorithm for file updates in Python

## Project description

Work with a text file containing IP addresses that are allowed to access specific restricted content within a potential organization.

Parsing a file allows security analysts to read and update the contents. Python helps analysts develop algorithms to automate the process of parsing files and keeping them up-to-date.

Develop an algorithm that parses this text file of IP addresses and updates the file by removing addresses that no longer have access to the restricted content.

## Open the file that contains the allow list

start by opening the text file using the import\_file variable, the with keyword, and the open() function with the "r" parameter.

Write the first line of the with statement. Running this code will produce an error because it will only contain the first line of the with statement.



The open() function in Python allows you to open a file.

As the first parameter, it takes in the name of the file (or a variable containing the name of the file).

As the second parameter, it takes in a string that indicates how the file should be handled.

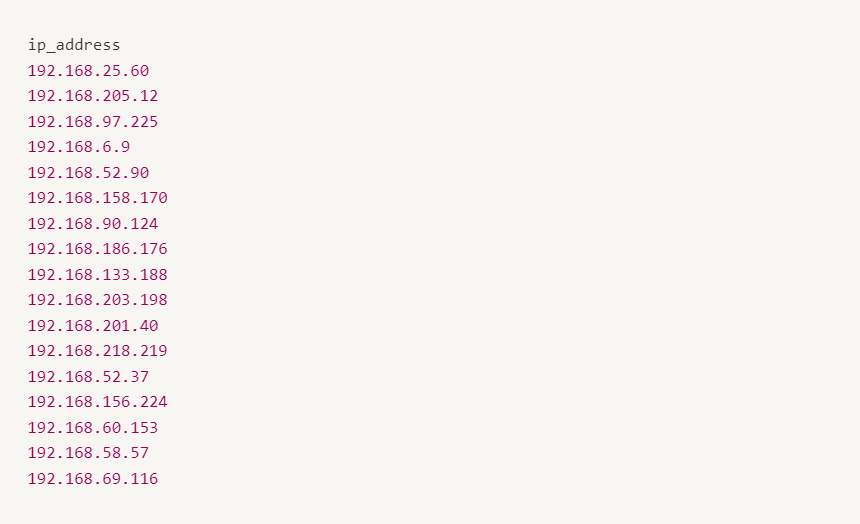
Passin the letter "r" as the second parameter will make the file available to be read.

## Read the file contents

Use the .read() method to read the imported file and store it in a variable named ip\_addresses.

Afterwards, display ip\_addresses to examine the data in its current format.





The .read() method in Python allows you to read in a file.

Call file.read() to read the imported file.

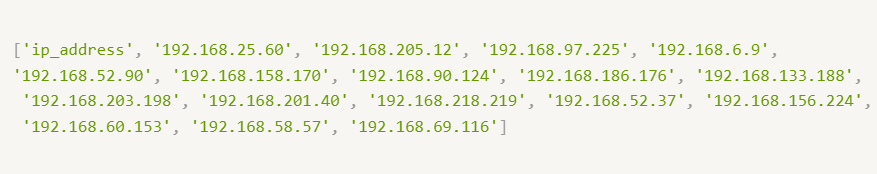
To display the contents of a variable, pass it as an argument to the print() function.

## Convert the string into a list

After reading the file, reassign the ip\_addresses variable so its data type is updated from a string to a list. Using the .split() method to achieve this.

Adding this step will allow it to iterate through each of the IP addresses in the allow list instead of navigating a large string that contains all the addresses merged together.

Afterwards, display the ip\_addresses variable to verify that the update took place.



The .split() method in Python allows you to convert a string to a list. This method can take in a parameter that specifies which character to split on. If a parameter is not passed in, the method will split on whitespace by default.

The default behavior of .split() works well. Each IP address is on a new line in the allow\_list.txt file. When .split() is used, it will separate the IP addresses and output them as a list.

To display the contents of a variable, pass it as an argument to the print() function.

## Iterate through the remove list

Write code that removes the elements of remove\_list from the ip\_addresses list. This will require both an iterative statement and a conditional statement.

First, build the iterative statement. Name the loop variable element, loop through ip\_addresses, and display each element.



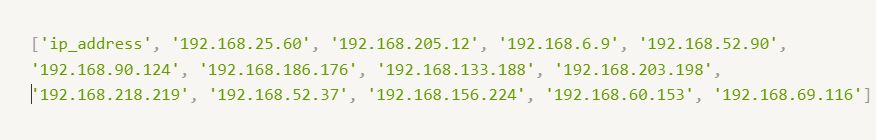
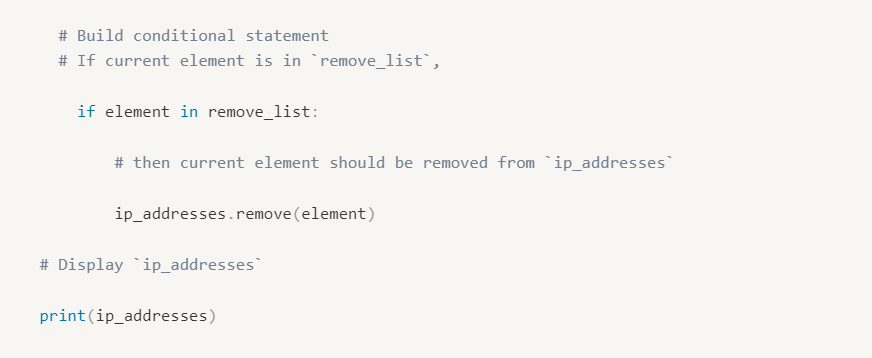
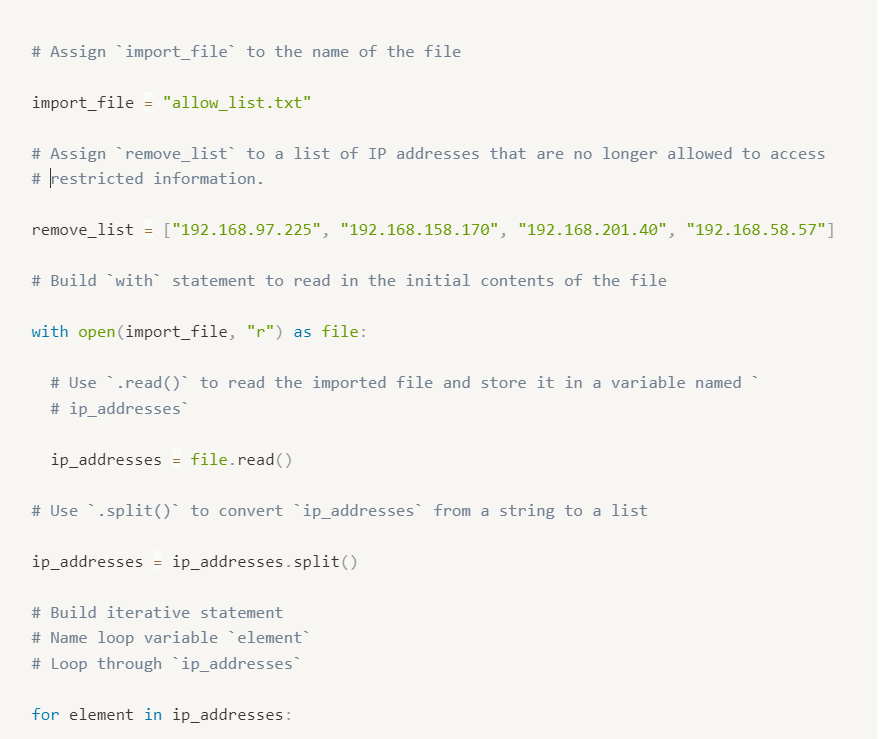
Build a for loop to iterate through ip\_addresses. Start with the for keyword. Use element as the loop variable and use in as the loop condition.

To display the contents of a variable, pass it as an argument to the print() function.

## Remove IP addresses that are on the remove list

Build a conditional statement to remove the elements of remove\_list from the ip\_addresses list. The conditional statement should be placed inside the iterative statement that loops through ip\_addresses. In every iteration, if the current element in the ip\_addresses list is in the remove\_list, the remove() method should be used to remove that element.

Afterwards, display the updated ip\_addresses list to verify that the elements of remove\_list are no longer in the ip\_addresses.



When building the conditional statement, use the in operator to check if element is in remove\_list.

To remove element from ip\_addresses, call the .remove() method on ip\_addresses, and pass in element.

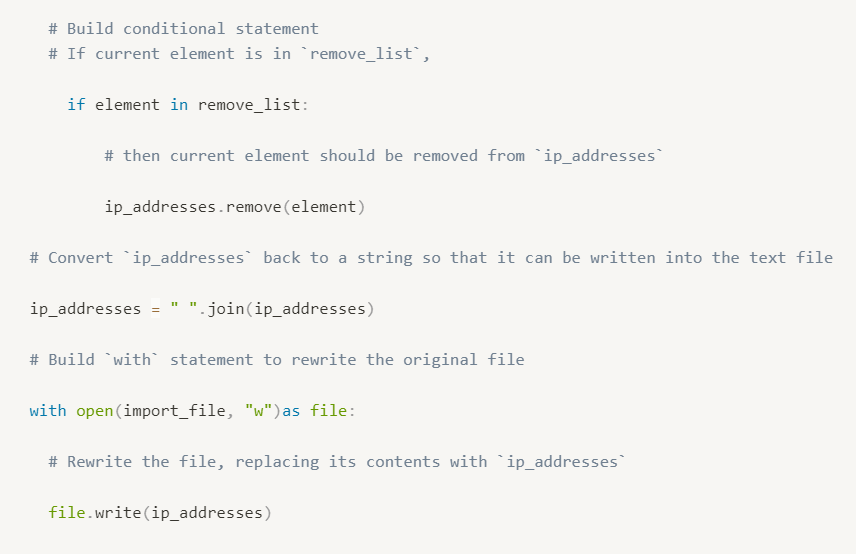
To remove element from ip\_addresses, call ip\_addresses.remove() and pass in element.

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

The next step is to update the original file that was used to create the ip\_addresses list. A line of code containing the .join() method has been added to the code so that the file can be updated. This is necessary because ip\_addresses must be in string format when used inside the with statement to rewrite the file.

The .join() method takes in an iterable (such as a list) and concatenates every element of it into a string. The .join() method is applied to a string consisting of the character that will be used to separate every element in the iterable once it's converted into a string. In the code below, the method is applied to the string " ", which contains just a space character. The argument of the .join() method is the iterable you want to convert, and in this case, that's ip\_addresses. As a result, it converts ip\_addresses from a list back into a string with a space between each element and the next.

After this line with the .join() method, build the with statement that rewrites the original file. Use the "w" parameter when calling the open() function to delete the contents in the original file and replace it with what you want to write.



Complete the first line of the `with` statement, call the `open()` function and pass in the name of the file as the first parameter and the letter `"w"` as the second parameter.

The `"w"` parameter specifies that you're opening the file for the purpose of writing to it.

Inside the `with` statement, call the `.write()` method to replace the contents of the file with the data stored in `ip\_addresses`.

Inside the `with` statement, call `file.write()` and pass in `ip\_addresses.

## Summary

The provided Python code defines a function **update\_file** that updates the contents of a text file containing IP addresses. The function reads the initial contents of the file, converts them into a list of IP addresses, removes the IP addresses specified in the **remove\_list**, converts the list back to a string, and rewrites the file with the updated contents. The code demonstrates the use of **with** statements and the **open()** function to handle file operations safely and efficiently. It also utilizes the **.read()** and **.write()** methods to read from and write to files, respectively. Additionally, the **.split()** method is used to convert the string of IP addresses into a list, and a **for** loop combined with the **.remove()** method is used to remove specific IP addresses from the list.

* + The with statement allows you to efficiently handle files.
  + The open() function allows you to import or open a file. It takes in the name of the file as the first parameter and a string that indicates the purpose of opening the file as the second parameter.
    - Specify "r" as the second parameter if you're opening the file for reading purposes.
    - Specify "w" as the second parameter if you're opening the file for writing purposes.
  + The .read() method allows you to read in a file.
  + The .write() method allows you to append or write to a file.
* You can use a for loop to iterate over a list.
* You can use an if statement to check if a given value is in a list and execute a specific action if so.
* You can use the .split() method to convert a string to a list.
* You can use Python to compare the contents of a text file against elements of a list.
* Algorithms can be incorporated into functions. When defining a function, you must specify the parameters it takes in and the actions it should execute.