

Algorithm for file updates in Python

❖ Project description

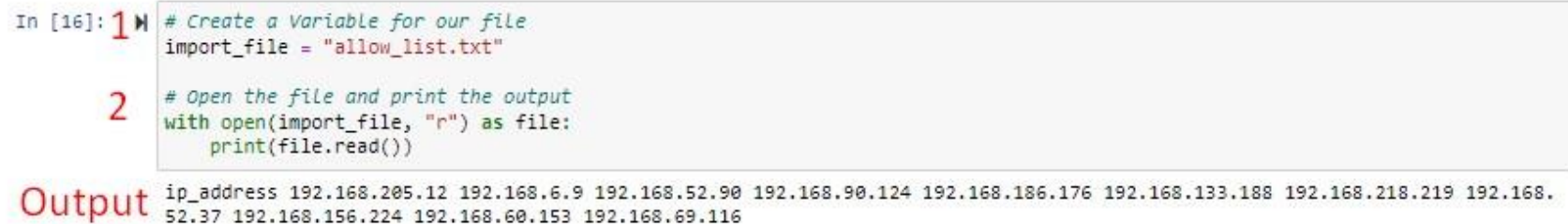
For this project I have been tasked with removing IP addresses that are required to be removed from the file **allow_list.txt**. I created an algorithm that allows us to update our **remove_list** variable with any IPs we'd like to block access for and then write this information back into the **allow_list.txt** file.

❖ 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.

Screenshot showing code to open the file that contains the allow list:



```
In [16]: 1 # Create a Variable for our file
import_file = "allow_list.txt"

2 # Open the file and print the output
with open(import_file, "r") as file:
    print(file.read())
```

Output ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219 192.168.52.37 192.168.156.224 192.168.60.153 192.168.69.116

1: As noted in comment 1, I create a variable called **import_file** that I'll set to our filename **allow_list.txt**.

2: As noted in comment 2, I then want to open the **allow_list.txt** file and then print the contents. I first utilize the **with** and **open** commands, passing in our **import_file** variable as the first parameter and “r” as the second parameter.

- The **with** statement and **.open()** function allow us to open the file. I use our **import_file** variable as the first parameter of the open function stating this is the file we want to open. I use “r” as the open functions second parameter, which means I want to read the file.

- Inside the **with** statement, I use the **as** keyword to assign another variable called **file**. The **file** variable will store the output of our **.open()** function.

Output: Shows the output of the code we’ve entered so far.

❖ 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.

```
In [4]: ▶ # Create a Variable for our file
import_file = "allow_list.txt"

# Open the file and read the contents
with open(import_file, "r") as file:

    1 # Use the .read() method to read the file and store in a variable called ip_addresses
    ip_addresses = file.read()
```

1: I create a comment stating we are using the **.read()** method to read the file and store its contents as a string in a variable called **ip_addresses**.

- Since I used the “r” parameter in our **open()** function, I can now call upon the **.read()** function.
- The **.read()** function will store the contents of our **file** variable as a string.
- I assign the string output to our variable **ip_addresses**.

In conclusion, so far the code reads the contents of the **allow_list.txt** file into a string.

❖ 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.

```
In [6]: ▶ # Create a Variable for our file
import_file = "allow_list.txt"

# Open the file and read the contents
with open(import_file, "r") as file:

    # Use the .read() method to read the file and store in a variable called ip_addresses
    ip_addresses = file.read()

1 # Use the .split() method to put format the string into a list
  ip_addresses = ip_addresses.split()

2 #Print our variable ip_addresses to confirm the string has been converted to a list
  print(ip_addresses)
```

Output ['ip_address', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']

1: I have our comment stating we are using the **.split()** method to convert the string into a list.

- I update the **ip_addresses** variable to equal **ip_address.split()**. Since I haven't passed an argument into the **.split()** method, the code will break up the string every time there is a **space**.

2: I print out the **ip_addresses** variable to confirm our string has successfully been converted to a list every time there is some whitespace.

Output: The output shows our **ip_addresses** variable has successfully been converted from a string into a list, broken up at every whitespace.

❖ 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.

```
In [9]: # Create a Variable for our file  
import_file = "allow_list.txt"  
  
# Open the file and read the contents  
with open(import_file, "r") as file:  
  
    # Use the .read() method to read the file and store in a variable called ip_addresses  
    ip_addresses = file.read()  
  
# Use the .split() method to put format the string into a List  
ip_addresses = ip_addresses.split()  
  
1 # Assign a variable called remove_list with ip addresses to remove  
remove_list = ["192.168.205.12", "192.168.60.153"]  
  
2 # Create a for Loop to to iterate through the remove List and use element as the Loop variable  
for element in remove_list:
```

1: I create a variable called **remove_list** with two IP addresses to remove as an example.

2: I start a **for** loop and a loop variable called **element** using the **in** command. This command will loop through the **remove_list** variable and assign each item in the list to the loop variable **element**.

❖ 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.

```
In [1]: # Create a Variable for our file
import_file = "allow_list.txt"

# Open the file and read the contents
with open(import_file, "r") as file:

    # Use the .read() method to read the file and store in a variable called ip_addresses
    ip_addresses = file.read()

# Use the .split() method to put format the string into a list
ip_addresses = ip_addresses.split()

# Assign a variable called remove_list with ip addresses to remove
remove_list = ["192.168.205.12", "192.168.60.153"]

# Create a for Loop to to iterate through the remove List and use element as the Loop variable
for element in remove_list:

1   #Create conditional statement to see if element is in ip_addresses list
    if element in ip_addresses:

2       ## if above condition is true, use the .remove() method to remove the element from ip_addresses
        ip_addresses.remove(element)

3   # Print ip_addresses variable to confirm ip addresses in remove_list were successfully removed
    print(ip_addresses)
```

Output ['ip_address', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.69.116']

1: I start our conditional statement saying **if** our variable **element** is in **ip_addresses**. This will look too see if an **element** (each individual value from **element** variable) is in our variable **ip_addresses**. It will look for a match of the strings from **element** to a string from **ip_addresses** and if it equates to **True**, it will move on to line 2 of our code.

2: If the variable **element** is also located in the variable **ip_addresses**, then we want to remove the **element** from **ip_addresses**.

3: I print the output of **ip_addresses** to confirm the two IP addresses we wanted to remove have successfully been removed from the list.

Output: As you can see in the output, the IP addresses "**192.168.205.12**", "**192.168.60.153**" are no longer included in the **ip_addresses** list.

❖ 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.

```

In [2]: # Create a Variable for our file
import_file = "allow_list.txt"

# Open the file and read the contents
with open(import_file, "r") as file:

    # Use the .read() method to read the file and store in a variable called ip_addresses
    ip_addresses = file.read()

# Use the .split() method to put format the string into a List
ip_addresses = ip_addresses.split()

# Assign a variable called remove_list with ip addresses to remove
remove_list = ["192.168.205.12", "192.168.60.153"]

# Create a for Loop to to iterate through the remove List and use element as the Loop variable
for element in remove_list:

    #Create conditional statement to see if element is in ip_addresses List
    if element in ip_addresses:

        ## if above condition is true, use the .remove() method to remove the element from ip_addresses
        ip_addresses.remove(element)

1 #Convert ip_addresses back to a string
ip_addresses = "\n".join(ip_addresses)

2 #Rewrite to the original file and replace the contents
with open(ip_addresses, "w") as file:
    file.write(ip_addresses)

3 #print out output of the new file
print(ip_addresses)

```

Output

```

ip_address
192.168.6.9
192.168.52.90
192.168.90.124
192.168.186.176
192.168.133.188
192.168.218.219
192.168.52.37
192.168.156.224
192.168.69.116

```

1: I want to convert **ip_addresses** back into a string with each IP address on a new line. I set the **ip_addresses** variable to **"\n".join(ip_addresses)**. The **"\n"** portion of the code when used with the **.join()** method will put all the items back into a string with each item on its own line.

2: I use the **with** statement and **open()** function to set our variable **import_file** as parameter 1 and “**w**” as parameter two. This will allow us to call the **write()** function in the body of the with statement. I then call the **write()** function and pass our variable **ip_addresses**. This will replace all the existing contents of the file with our new string data.

3: I then print the output of the new file to confirm our work was successful.

Output: I can confirm our work was successful as each item is now on a new line and our file has successfully been written too with the new information.

❖ Summary

I created an algorithm using Python that removes IPs that I identified in my **remove_list** variable. The file we use is **allow_list.txt** and it contains IP addresses. I first open the file and convert it to a string so the data can be read by the code. I then set this data to my variable **ip_addresses**. I then create a **for** loop to cycle through each item in the **remove_list** and create a **for** loop variable called **element**. Our for loop looks for a match between IPs in **remove_list** and **ip_addresses**. If there is a match, I have written an **if** statement that will remove the matching IPs using the **.remove()** method. I then use the **.join()** method to convert the data back into a string and write the contents back to **allow_list.txt**. For the **.join()** method, I also use “**\n**” to put each item on a new line. The final result now shows all IPs, minus the ones that were in the **remove_list** variable.