LAB-013-Exemplar_Create another algorithm

September 20, 2024

1 Modelo: Crear otro algoritmo

Introducción

Una parte importante de la ciberseguridad es controlar el acceso al contenido restringido. En este laboratorio, trabajarás con un archivo de texto que contiene direcciones IP a las que se les permite acceder a contenido restringido específico de tu organización.

El análisis de un archivo permite a los analistas de seguridad leer y actualizar el contenido. Python ayuda a los analistas a desarrollar algoritmos para automatizar el proceso de analizar archivos y mantenerlos actualizados.

Desarrollarás un algoritmo que analice este archivo de texto de direcciones IP y lo actualice eliminando las direcciones que ya no tienen acceso al contenido restringido.

Consejos para completar este laboratorio

Mientras recorres este laboratorio, ten en cuenta los siguientes consejos:

— `### TU CÓDIGO AQUÍ ###` indica dónde debes escribir el código. Asegúrate de reemplazarlo con tu propio código antes de ejecutar la celda de código. — Siéntete libre de abrir las pistas para obtener información adicional a medida que trabajas en cada tarea. — Para ingresar tu respuesta a una pregunta, haz doble clic en la celda de markdown para editar. Asegúrate de reemplazar "[Haz doble clic para ingresar aquí tus respuestas.]" con tu propia respuesta. — Puedes guardar tu trabajo manualmente con un clic en Archivo y luego en Guardar en la barra de menú en la parte superior del cuaderno. — Puedes descargar tu trabajo localmente con un clic en Archivo y luego en Descargar, luego puedes especificar el formato de archivo que prefieras en la barra de menú en la parte superior del cuaderno.

1.1 Situación hipotética

En este laboratorio, trabajarás como analista de seguridad y serás responsable de desarrollar un algoritmo que analice un archivo que contiene direcciones IP a las que se les permite acceder a contenido restringido y elimine las direcciones que ya no tienen acceso.

Tarea 1 Tu objetivo final es desarrollar un algoritmo que analice una serie de direcciones IP que pueden acceder a información restringida y elimine aquellas que ya no pueden hacerlo. Python puede automatizar este proceso.

Se te proporciona un archivo de texto llamado `"allow_list.txt"` que contiene una serie de direcciones IP a las que se les permite acceder a información restringida.

Nota: En la actividad, el valor de la cadena import_file es `"allow_list.txt"`, pero en el modelo es `"data/allow_list.txt"`. Esto es intencional. En el modelo, se movió una copia única de `"allow_list.txt"` a una carpeta nueva para que los cambios realizados en el archivo de la actividad no afecten al archivo utilizado en el modelo.

Hay direcciones IP que ya no deberían tener acceso a esta información y es necesario eliminarlas del archivo de texto. Se te proporciona una variable llamada `remove_list` que contiene la lista de direcciones IP que es necesario eliminar.

Muestra ambas variables para explorar su contenido y ejecuta la celda. Asegúrate de reemplazar cada `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[1]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"

# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
 ↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
 ↪58.57"]

# Muestra `import_file`

print(import_file)

# Muestra `remove_list`

print(remove_list)
```

```
data/allow_list.txt
['192.168.97.225', '192.168.158.170', '192.168.201.40', '192.168.58.57']
```

Pista 1

Para mostrar el contenido de una variable, pásala como argumento a la función `print()`.

Pregunta 1 ¿Qué observas de la salida anterior?

La primera línea de la salida muestra el nombre del archivo de texto. La segunda línea de la salida muestra la lista de direcciones IP de `remove_list`.

1.2 Tarea 2

En esta tarea, comienza por abrir el archivo de texto con la variable `import_file`, la palabra clave `with` y la función `open()` con el parámetro `"r"`. Asegúrate de reemplazar `### TU CÓDIGO AQUÍ ###` con tu propio código. Por ahora, escribirás la primera línea de la sentencia `with`. La ejecución de este código producirá un error, ya que solo contendrá la primera línea de la sentencia `with`; completarás esta sentencia `with` en la tarea después de esto.

[2]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"

# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
 ↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
 ↪58.57"]

# La primera línea de la sentencia `with`

with open(import_file, "r") as file:
```

```
       [0;36m  File [0;32m"<ipython-input-2-570da020cff3>"[0;36m, line [0;
↪32m11[0m
    [0;31m    with open(import_file, "r") as file:[0m
    [0m                                            ^[0m
    [0;31mSyntaxError[0m[0;31m:[0m unexpected EOF while parsing
```

Pista 1

La función `open()` en Python te permite abrir un archivo.

Como primer parámetro, toma el nombre del archivo (o una variable que contiene el nombre del archivo). Como segundo parámetro, toma una cadena que indica cómo se debe gestionar el archivo.

Pasa la letra `"r"` como segundo parámetro cuando quieras leer el archivo.

Tarea 3 Ahora usa el método `.read()` para leer el archivo importado y almacenarlo en una variable llamada `ip_addresses`.

Luego, muestra `ip_addresses` para examinar los datos en su formato actual.

Asegúrate de reemplazar cada `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[4]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"

# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
 ↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
 ↪58.57"]
```

```
# Crea la sentencia `with` para leer el contenido inicial del archivo

with open(import_file, "r") as file:

  # Usa `.read()` para leer el archivo importado y almacenarlo en una variable␣
 ↪llamada `ip_addresses`

    ip_addresses = file.read()

# Muestra `ip_addresses`

print(ip_addresses)
```

```
ip_address 192.168.25.60 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124
192.168.186.176 192.168.133.188 192.168.203.198 192.168.218.219 192.168.52.37
192.168.156.224 192.168.60.153 192.168.69.116
```

Pista 1

En Python, el método `.read()` te permite leer un archivo.

Pista 2

Llama a `file.read()` para leer el archivo importado.

Pista 3

Para mostrar el contenido de una variable, pásala como argumento a la función `print()`.

Pregunta 2 ¿Alguna de las direcciones IP de la lista de permisos también está en `remove_list`?

Hay cuatro direcciones IP en la lista de permisos que también están en `remove_list`.

1.3 Tarea 4

Después de leer el archivo, vuelve a asignar la variable `ip_addresses` para actualizar el tipo de datos de una cadena a una lista. Para hacerlo, usa el método `.split()`. Si añades este paso, después podrás eliminar direcciones IP de la lista de permisos.

Luego, muestra la variable `ip_addresses` para verificar que se actualizó.

Asegúrate de reemplazar cada `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[5]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"
```

```
# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
↪58.57"]

# Crea la sentencia `with` para leer el contenido inicial del archivo

with open(import_file, "r") as file:

  # Usa `.read()` para leer el archivo importado y almacenarlo en una variable␣
↪llamada `ip_addresses`

    ip_addresses = file.read()

# Usa `.split()` para convertir `ip_addresses` de una cadena a una lista

ip_addresses = ip_addresses.split()

# Muestra `ip_addresses`

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9',
'192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188',
'192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224',
'192.168.60.153', '192.168.69.116']
```

Pista 1

— En Python, el método `.split()` te permite convertir una cadena en una lista. Este método puede tomar un parámetro que especifique en qué carácter realizar la división. Si no se pasa un parámetro, de forma predeterminada, el método se dividirá en espacios en blanco. Ten en cuenta que el espacio en blanco incluye cualquier espacio entre el texto de la misma línea y el espacio entre una línea y la siguiente.

En esta tarea, el comportamiento predeterminado de `.split()` funciona bien. En el archivo `allow_list.txt`, cada dirección IP está en una nueva línea. En otras palabras, en el archivo de texto, hay un espacio en blanco entre las direcciones IP. Al usar `.split()`, las direcciones IP quedan separadas en forma de lista.

Pista 2

Para mostrar el contenido de una variable, pásala como argumento a la función `print()`.

Tarea 5 Ahora, escribirás un código que elimine los elementos de `remove_list` de la lista `ip_addresses`. Necesitarás usar una sentencia iterativa.

Primero, crea la sentencia iterativa. Nombra la variable de bucle `element` que recorre en bucle `remove_list` y muestra cada elemento. Asegúrate de reemplazar cada `### TU CÓDICO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[6]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"

# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
↪58.57"]

# Crea la sentencia `with` para leer el contenido inicial del archivo

with open(import_file, "r") as file:

  # Usa `.read()` para leer el archivo importado y almacenarlo en una variable␣
↪llamada `ip_addresses`

    ip_addresses = file.read()

# Usa `.split()` para convertir `ip_addresses` de una cadena a una lista

ip_addresses = ip_addresses.split()

# Crea una sentencia iterativa
# Nombra la variable de bucle `element`
# Recorre en bucle `remove_list`

for element in remove_list:

  # Muestra `element` en cada iteración

    print(element)
```

```
192.168.97.225
192.168.158.170
192.168.201.40
192.168.58.57
```

Pista 1

Crea un bucle `for` que se recorra en iteración `remove_list`. Asegúrate de comenzar con la palabra clave `for`. Usa `element` como variable de bucle e `in` como condición de bucle.

Pista 2

Para mostrar el contenido de una variable, pásala como argumento a la función `print()`.

1.4 Tarea 6

Ahora, en el cuerpo de la sentencia iterativa, aplica el método `.remove()` a la lista `ip_addresses` y elimina las direcciones IP identificadas en la variable de bucle `element`.

Después de que la sentencia iterativa elimina los elementos, muestra la lista `ip_addresses` actualizada para comprobar que los elementos de `remove_list` ya no están en `ip_addresses`. Asegúrate de reemplazar cada `### TU CÓDICO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

No olvides las dos cualidades de las listas que hacen posible la aplicación del método `.remove()`. Primero, todos los elementos en `remove_list` también están en la lista `ip_addresses`. Si no fuera así, aplicar el método `.remove()` a estos elementos generaría un error. En segundo lugar, no hay valores duplicados en la lista `ip_addresses`. El método `.remove()` solo elimina la primera aparición de un elemento, por lo que si hubiera valores duplicados, no se eliminarían.

[ ]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"

# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
 ↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
 ↪58.57"]

# Crea la sentencia `with` para leer el contenido inicial del archivo

with open(import_file, "r") as file:

  # Usa `.read()` para leer el archivo importado y almacenarlo en una variable␣
 ↪llamada `ip_addresses`

    ip_addresses = file.read()

# Usa `.split()` para convertir `ip_addresses` de una cadena a una lista

ip_addresses = ip_addresses.split()

# Crea una sentencia iterativa
# Nombra la variable de bucle `element`
# Recorre en bucle `remove_list`

for element in remove_list:

  # usa el método `.remove()` para eliminar
  # elementos de `ip_addresses`
```

```
    ip_addresses.remove(element)

# Muestra `ip_addresses`

print(ip_addresses)
```

Pista 1

Para eliminar `element` de `ip_addresses`, llama al método `.remove()` en `ip_addresses` y pásale `element` como argumento.

Pista 2

Para eliminar `element` de `ip_addresses`, llama a `ip_addresses.remove()` y pásale `element` como argumento.

1.5 Tarea 7

El paso siguiente es actualizar el archivo original que se usó para crear la lista `ip_addresses`. Se agregó una línea de código que contiene el método `.join()` para que el archivo se pueda actualizar. Esto es necesario porque `ip_addresses` debe estar en formato de cadena cuando se usa dentro de la sentencia `with` para reescribir el archivo.

El método `.join()` toma un elemento iterable (como una lista) y concatena cada uno de sus elementos en una cadena. Se aplica el método `.join()` a una cadena que consiste en el carácter que se usará para separar cada elemento en el iterable una vez que se convierta en una cadena. En el siguiente código, se aplica el método a la cadena `" "`, que contiene solo un carácter de espacio. El argumento del método `.join()` es el iterable que deseas convertir que, en este caso, es `ip_addresses`. Como resultado, vuelve a convertir la lista `ip_addresses` en una cadena con un espacio entre cada elemento.

Después de la línea con el método `.join()`, crea la sentencia `with` que reescribirá el archivo original. Usa el parámetro `"w"` cuando necesites llamar a la función `open()` para eliminar el contenido del archivo original y reemplazarlo con lo que quieras escribir. Asegúrate de reemplazar cada `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda. Esta celda de código no generará una salida.

[ ]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"

# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
 ↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
 ↪58.57"]

# Crea la sentencia `with` para leer el contenido inicial del archivo
```

```
with open(import_file, "r") as file:

  # Usa `.read()` para leer el archivo importado y almacenarlo en una variable␣
 ↪llamada `ip_addresses`

    ip_addresses = file.read()

# Usa `.split()` para convertir `ip_addresses` de una cadena a una lista

ip_addresses = ip_addresses.split()

# Crea una sentencia iterativa
# Nombra la variable de bucle `element`
# Recorre en bucle `remove_list`

for element in remove_list:

  # usa el método `.remove()` para eliminar
  # elementos de `ip_addresses`

    ip_addresses.remove(element)

# Vuelve a convertir `ip_addresses` en una cadena para que se pueda escribir en␣
 ↪un archivo de texto

ip_addresses = " ".join(ip_addresses)

# Crea una sentencia `with` para reescribir el archivo original

with open(import_file, "w") as file:

  # Para reescribir el archivo, reemplaza su contenido con `ip_addresses`

    file.write(ip_addresses)
```

Pista 1

Para completar la primera línea de la sentencia `with`, llama a la función `open()` y pasa el nombre del archivo como primer parámetro y la letra `"w"` como segundo parámetro.

El parámetro `"w"` especifica que estás abriendo el archivo con el propósito de escribir en este.

Pista 2

En la sentencia `with`, llama al método `.write()` para reemplazar el contenido del archivo con los datos almacenados en `ip_addresses`.

Pista 3

Dentro de la sentencia `with`, llama a `file.write()` y pásale `ip_addresses` como argumento.

1.6 Tarea 8

En esta tarea, comprobarás que el archivo original se reescribió con la lista correcta.

Escribe otra sentencia `with`, esta vez para leer el archivo actualizado. Comienza abriendo el archivo. Luego, lee el archivo y guarda el contenido en la variable `text`.

Muestra la variable `text` para examinar el resultado.

Asegúrate de reemplazar cada `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[3]:
```
# Asigna a `import_file` el nombre del archivo

import_file = "data/allow_list.txt"

# Asigna a `remove_list` una lista de direcciones IP a las que ya no se les␣
 ↪permite acceder a información restringida.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
 ↪58.57"]

# Crea la sentencia `with` para leer el contenido inicial del archivo

with open(import_file, "r") as file:

  # Usa `.read()` para leer el archivo importado y almacenarlo en una variable␣
 ↪llamada `ip_addresses`

    ip_addresses = file.read()

# Usa `.split()` para convertir `ip_addresses` de una cadena a una lista

ip_addresses = ip_addresses.split()

# Crea una sentencia iterativa
# Nombra la variable de bucle `element`
# Recorre en bucle `remove_list`

for element in remove_list:

  # usa el método `.remove()` para eliminar
  # elementos de `ip_addresses`

    ip_addresses.remove(element)

# Vuelve a convertir `ip_addresses` en una cadena para que se pueda escribir en␣
 ↪un archivo de texto
```

```
ip_addresses = " ".join(ip_addresses)

# Crea una sentencia `with` para reescribir el archivo original

with open(import_file, "w") as file:

  # Para reescribir el archivo, reemplaza su contenido con `ip_addresses`

    file.write(ip_addresses)

# Crea la sentencia `with` para leer el archivo actualizado

with open(import_file, "r") as file:

  # Lee el archivo actualizado y almacena el contenido en `text`

    text = file.read()

# Muestra el contenido de `text`

print(text)
```

```
       [0;
↪31m---------------------------------------------------------------------------[0m

       [0;31mValueError[0m                                Traceback (most␣
↪recent call last)

       [0;32m<ipython-input-3-5ca175e506c2>[0m in [0;36m<module>[0;34m[0m
   [1;32m     28[0m   [0;31m# elements from `ip_addresses`[0m[0;34m[0m[0;
↪34m[0m[0;34m[0m[0m
   [1;32m     29[0m [0;34m[0m[0m
   [0;32m---> 30[0;31m     [0mip_addresses[0m[0;34m.[0m[0mremove[0m[0;
↪34m([0m[0melement[0m[0;34m)[0m[0;34m[0m[0;34m[0m[0m
   [0m[1;32m     31[0m [0;34m[0m[0m
   [1;32m     32[0m [0;31m# Convert `ip_addresses` back to a string so that it␣
↪can be written into the text file[0m[0;34m[0m[0;34m[0m[0;34m[0m[0m


       [0;31mValueError[0m: list.remove(x): x not in list
```

Pista 1

Para completar la primera línea de la sentencia `with`, llama a la función `open()` y pásale el nombre del archivo como primer parámetro y la letra `"r"` como segundo parámetro.

El parámetro `"r"` especifica que estás abriendo el archivo con el propósito de leerlo.

Pista 2

En la sentencia `with`, llama al método `.read()` para leer el contenido del archivo. Asigna la variable `text` al resultado.

Pista 3

Para mostrar el contenido de una variable, pásala como argumento a la función `print()`.

1.7 Tarea 9

El paso siguiente es llevar todo el código que escribiste hasta este punto y ponerlo todo en una sola función.

Define una función llamada `update_file()` que recibe dos parámetros. El primer parámetro es el nombre del archivo de texto que contiene las direcciones IP (llama a este parámetro `import_file`). El segundo parámetro es una lista que contiene las direcciones IP a eliminar (llama a este parámetro `remove_list`).

Asegúrate de reemplazar `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda. Ten en cuenta que esta celda de código no generará ninguna salida.

[ ]:
```
# Define una función llamada `update_file` que recibe dos parámetros:␣
↪`import_file` y `remove_list`
# y combina los pasos que escribiste en este laboratorio hasta el momento

def update_file(import_file, remove_list):

  # Crea la sentencia `with` para leer el contenido inicial del archivo

    with open(import_file, "r") as file:

    # Usa `.read()` para leer el archivo importado y almacenarlo en una␣
↪variable llamada `ip_addresses`

        ip_addresses = file.read()

  # Usa `.split()` para convertir `ip_addresses` de una cadena a una lista

    ip_addresses = ip_addresses.split()

  # Crea una sentencia iterativa
  # Nombra la variable de bucle `element`
  # Recorre en bucle `remove_list`

    for element in remove_list:

    # usa el método `.remove()` para eliminar
```

```
    # elementos de `ip_addresses`

        ip_addresses.remove(element)

  # Vuelve a convertir `ip_addresses` en una cadena para que se pueda escribir␣
↪en un archivo de texto

    ip_addresses = " ".join(ip_addresses)

  # Crea una sentencia `with` para reescribir el archivo original

    with open(import_file, "w") as file:

    # Para reescribir el archivo, reemplaza su contenido con `ip_addresses`

        file.write(ip_addresses)
```

Pista 1

La definición de la función comienza con la palabra clave `def`.

Pista 2

Después de la palabra clave `def`, especifica el nombre de la función, seguido de paréntesis y dos puntos. Dentro de los paréntesis, especifica los parámetros que toma la función.

Pista 3

Después de la palabra clave `def`, escribe `update_file(import_file, remove_list):` para completar la cabecera de definición de la función.

Pregunta 3 ¿Cuáles son los beneficios de incorporar el algoritmo en una sola función?

La incorporación del algoritmo en una sola función ayuda a organizar el código y hacerlo reutilizable. Si quieres ejecutar el algoritmo más de una vez, lo que tienes que hacer es llamar a la función que lo contiene.

1.8 Tarea 10

Finalmente, llama a la función `update_file()` que definiste. Aplica la función a `"allow_list.txt"` y pásale una lista de direcciones IP como segundo argumento.

Usa la siguiente lista de direcciones IP como segundo argumento:

`["192.168.25.60", "192.168.140.81", "192.168.203.198"]`

Después de llamar a la función, usa una sentencia `with` para leer el contenido de la lista de permisos. Luego, muestra el contenido de la lista de permisos. Ejecútalo para comprobar que la función actualizó el archivo.

Asegúrate de reemplazar `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[ ]:
```
# Define una función llamada `update_file` que recibe dos parámetros: `import_file` y `remove_list`
# y combina los pasos que escribiste en este laboratorio hasta el momento

def update_file(import_file, remove_list):

  # Crea la sentencia `with` para leer el contenido inicial del archivo

    with open(import_file, "r") as file:

    # Usa `.read()` para leer el archivo importado y almacenarlo en una variable llamada `ip_addresses`

        ip_addresses = file.read()

  # Usa `.split()` para convertir `ip_addresses` de una cadena a una lista

    ip_addresses = ip_addresses.split()

  # Crea una sentencia iterativa
  # Nombra la variable de bucle `element`
  # Recorre en bucle `remove_list`

    for element in remove_list:

    # usa el método `.remove()` para eliminar
    # elementos de `ip_addresses`

        ip_addresses.remove(element)

  # Vuelve a convertir `ip_addresses` en una cadena para que se pueda escribir en un archivo de texto

    ip_addresses = " ".join(ip_addresses)

  # Crea una sentencia `with` para reescribir el archivo original

    with open(import_file, "w") as file:

    # Para reescribir el archivo, reemplaza su contenido con `ip_addresses`

        file.write(ip_addresses)
```

```
# Llama a `update_file()` y pásale "allow_list.txt" como argumento y una lista␣
↪de las direcciones IP a eliminar

update_file("data/allow_list.txt", ["192.168.25.60", "192.168.140.81", "192.168.
↪203.198"])

# Crea la sentencia `with` para leer el archivo actualizado

with open("data/allow_list.txt", "r") as file:

  # Lee el archivo actualizado y almacena el contenido en `text`

    text = file.read()

# Muestra el contenido de `text`

print(text)
```

Pista 1

Para llamar a la función `update_file()`, escribe el nombre de la función seguido de los paréntesis y pásale el nombre del archivo y la lista de las direcciones IP en las que deseas probar la función. Asegúrate de separar los dos argumentos con una coma (`,`).

Pista 2

En la sentencia `with`, llama al método `.read()` para leer el contenido del archivo. Asigna la variable `text` al resultado.

Pista 3

Para mostrar el contenido de la variable `text`, pásala como argumento a la función `print()`.

1.9 Conclusión

¿Qué conclusiones clave obtuviste de este laboratorio?

— Python tiene funciones y sintaxis que te ayudan a importar y analizar archivos de texto. — La sentencia `with` te permite gestionar archivos de manera eficiente. — La función `open()` te permite importar o abrir un archivo. Toma el nombre del archivo como primer parámetro y una cadena que indica el propósito de abrir el archivo como segundo parámetro. — Especifica `"r"` como segundo parámetro si quieres abrir el archivo con fines de lectura. — Especifica `"w"` como segundo parámetro si quieres abrir el archivo con fines de escritura. — El método `.read()` te permite leer un archivo. — El método `.write()` te permite agregar datos o escribir en un archivo. — Puedes usar un bucle `for` para recorrer en iteración una lista. — Puedes usar el método `.split()` para convertir una cadena en una lista. — Puedes usar Python para comparar el contenido de un archivo de texto con los elementos de una lista. — Se pueden incorporar algoritmos a las funciones. Al definir una función, debes especificar los parámetros que toma y las acciones que debe ejecutar.