Week Two

CSOH| Python Study Group

Cloud Security Office Hours

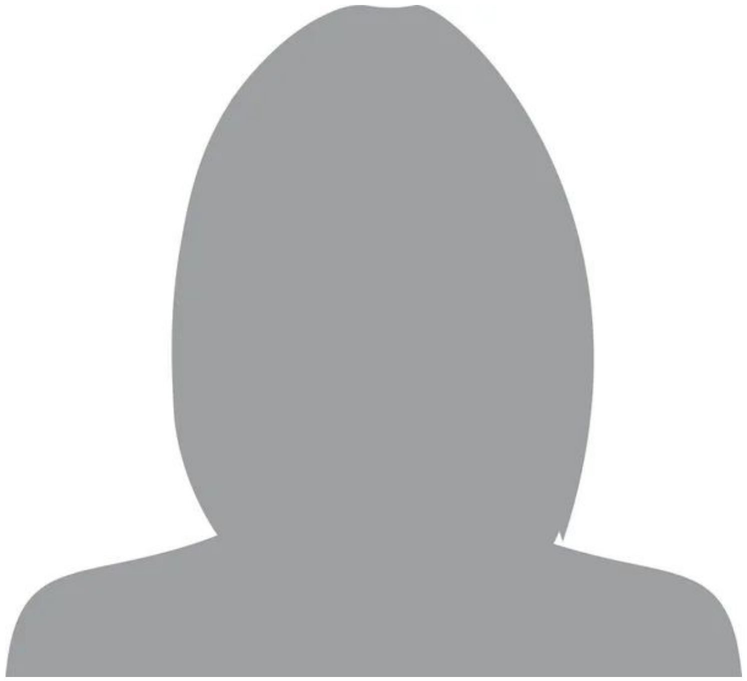
Thanks for the space!

Once a week, every Friday, we host a Zoom call that is an open forum for novices and experts in Cloud Security to collaborate and encourage each other in this field.

Fridays @ 10:00 AM
Same Zoom

https://csoh.org/

id



i am D (they/them) or Hetz

recent cybersecurity grad

interested in malware analysis
and application security

not an expert, just an enthusiast
tired of tutorial hell

and chihuahuas…lots of
chihuahuas

Grounding Assumptions

1. **We’re all here to learn.**
   Everyone starts somewhere, it’s okay to ask questions and make mistakes. This is a space for us to work and learn together.

2. **No prior coding experience required.**
   We’ll explain each project and ask our advanced group members for support, when needed.

3. **Python is forgiving.**
   Small errors happen! Syntax mistakes are part of learning how code “thinks.”

4. **We’ll focus on understanding, not memorizing.**
   Knowing *why* something works is more valuable than remembering every detail.

5. **Try it yourself.**
   The best way to learn coding is by typing, testing, and experimenting.

6. **Respectful collaboration.**
   Share ideas, help each other, and be patient- we’re learning together.

7. **Keep it simple today.**
   We’ll start with the basics and build up over time — no need to rush.

Schedule

1. Codespace Review (10 min)
2. Lesson Review (25 min)
3. Mini Project: Input & Output (30 min)
4. Show & Tell (25 min)

Focus & Objectives

Focus:

Open text files, filter keywords, detect patterns like "DROP" or "DENY" in fake firewall logs.

Objectives:

- Read files line by line
- Filter lines with keywords
- Write filtered results to a new file

open(“ing”) Files

open(“ur file here”) function:

```
f = open(“firewall.txt”)
print(f.read())
```

returns a file object, which has a read() method for reading the content of the file.

Open a file on a different location:

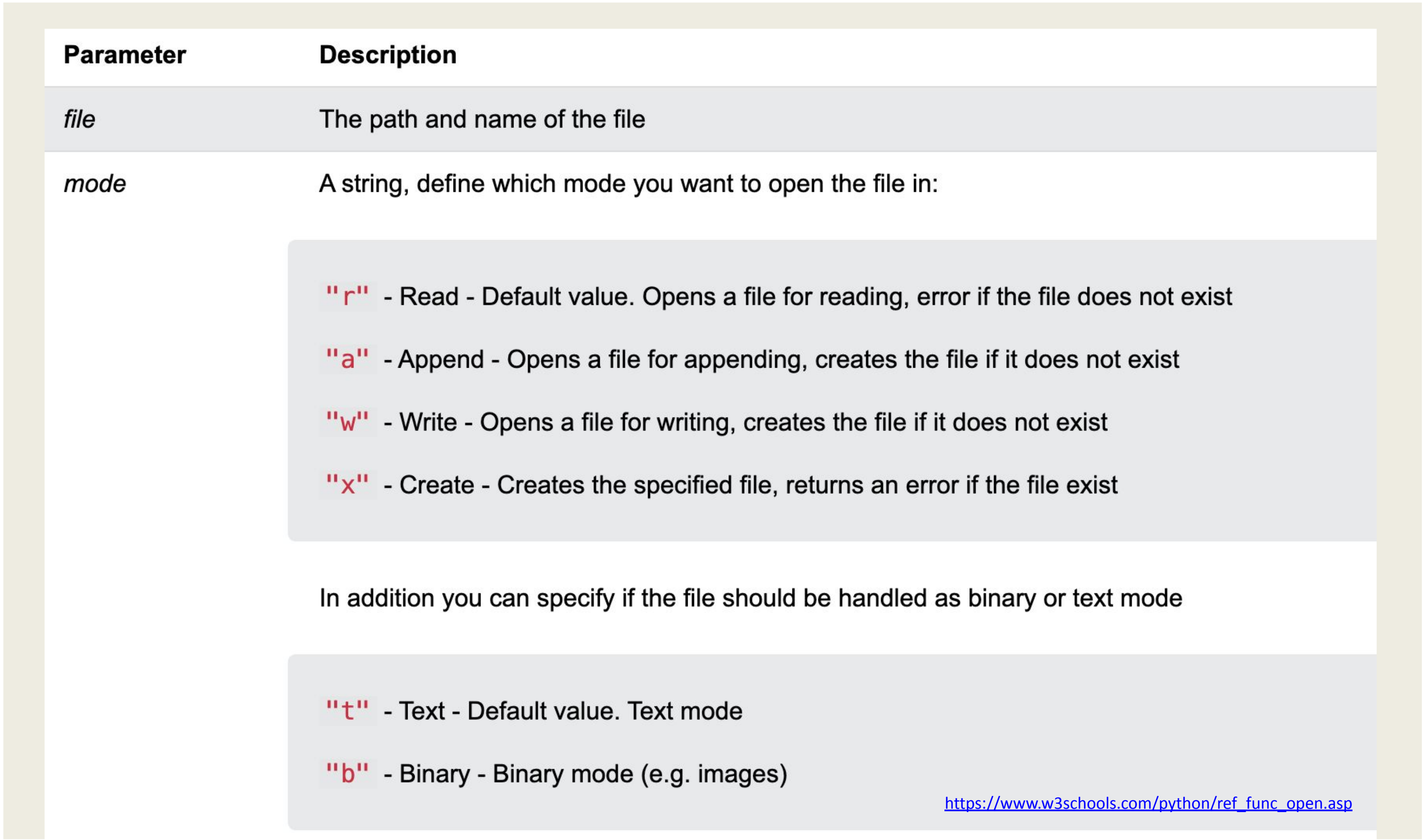
```
f =
open("D:\\myfiles\welcome.txt")

print(f.read())
```

https://www.w3schools.com/python/python_file_open.asp

open(“Parameter_Values”)

```
f = open("demofile.txt", "rt")
```



Parameter	Description
file	The path and name of the file
mode	A string, define which mode you want to open the file in: "r" - Read - Default value. Opens a file for reading, error if the file does not exist "a" - Append - Opens a file for appending, creates the file if it does not exist "w" - Write - Opens a file for writing, creates the file if it does not exist "x" - Create - Creates the specified file, returns an error if the file exist In addition you can specify if the file should be handled as binary or text mode "t" - Text - Default value. Text mode "b" - Binary - Binary mode (e.g. images)

https://www.w3schools.com/python/ref_func_open.asp

f.expressions()

When you call open(), it returns this file object, which is an interface that allows you to interact with the opened file. You can then use methods of this file object to perform operations.

f: represents a file object. It is also commonly referred to as a file handle or file pointer.

```
f = open("D:\\myfiles\welcome.txt")
```

```
print(f.read())
```

```
Reading: f.read(), f.readline(), f.readlines()
Writing: f.write(), f.writelines()
Seeking: f.seek() (change the current position
within the file)
Closing: f.close() (with open(...) as f: handles
this automatically)
```

The with Statement

You use the **with** statement to open a file safely. Once the block of code inside it finishes running, Python automatically closes the file for you.

```
with open("demofile.txt") as f:
  print(f.read())
```

this becomes name of your new file!

Read Only Parts of the File

By default the **read()** method returns the whole text, but you can also specify how many characters you want to return.

Return the 5 first characters of the file:

```
with open("demofile.txt") as f:

  print(f.read(5))
```

If you are not using the `with` statement, you must write a close statement in order to close the file

https://www.w3schools.com/python/python_file_open.asp

Python csv Module

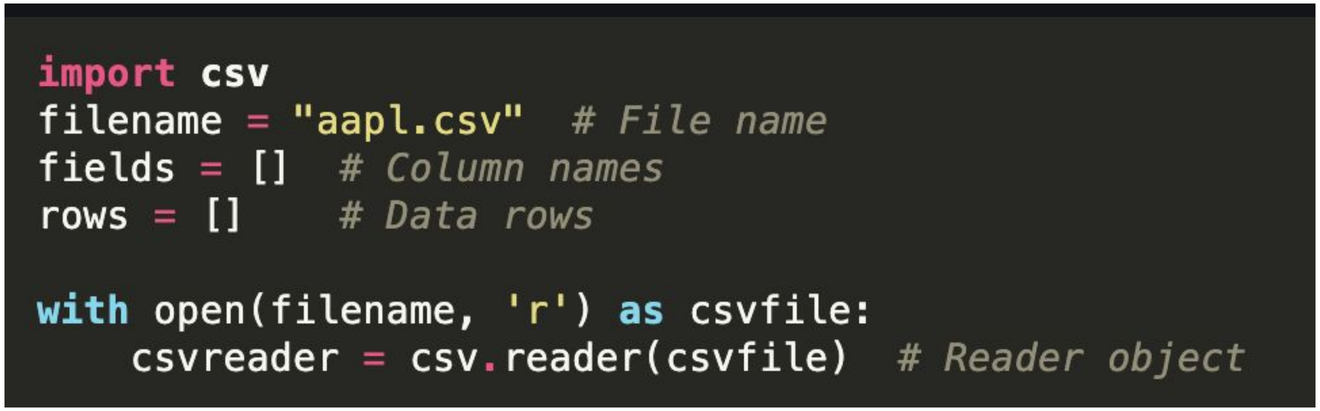
- The csv module implements classes to read and write tabular data in CSV format.
- It is a built-in module: `import csv`
- The module's reader and writer objects read and write sequences.
- Read and write data in dictionary form using DictReader and DictWriter classes.

https://docs.python.org/3/library/csv.html#module-csv

Reading a csv

csv.reader: returns a reader object that will process lines from csv file.

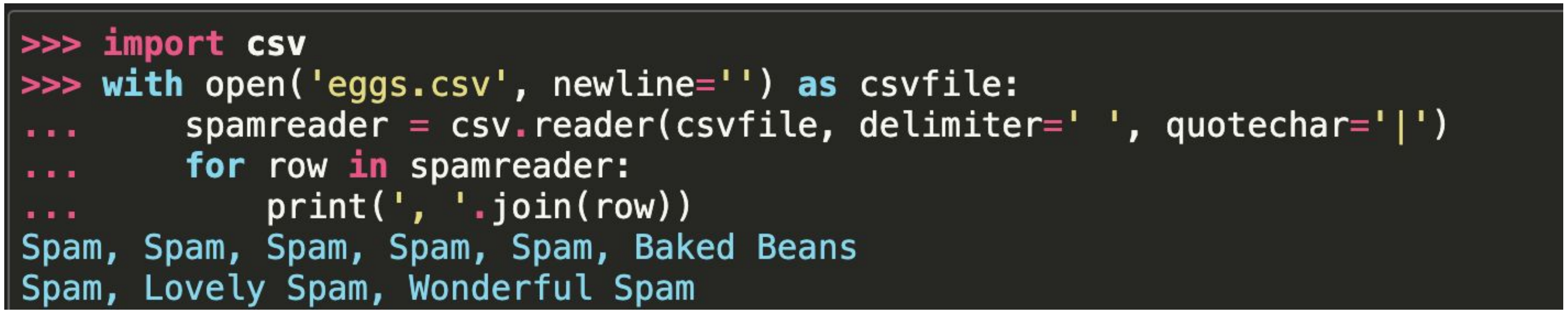
The CSV file is opened as a text file with Python's built-in open() function, which returns a file object.



```python
import csv
filename = "aapl.csv"  # File name
fields = []  # Column names
rows = []    # Data rows

with open(filename, 'r') as csvfile:
    csvreader = csv.reader(csvfile)  # Reader object
```

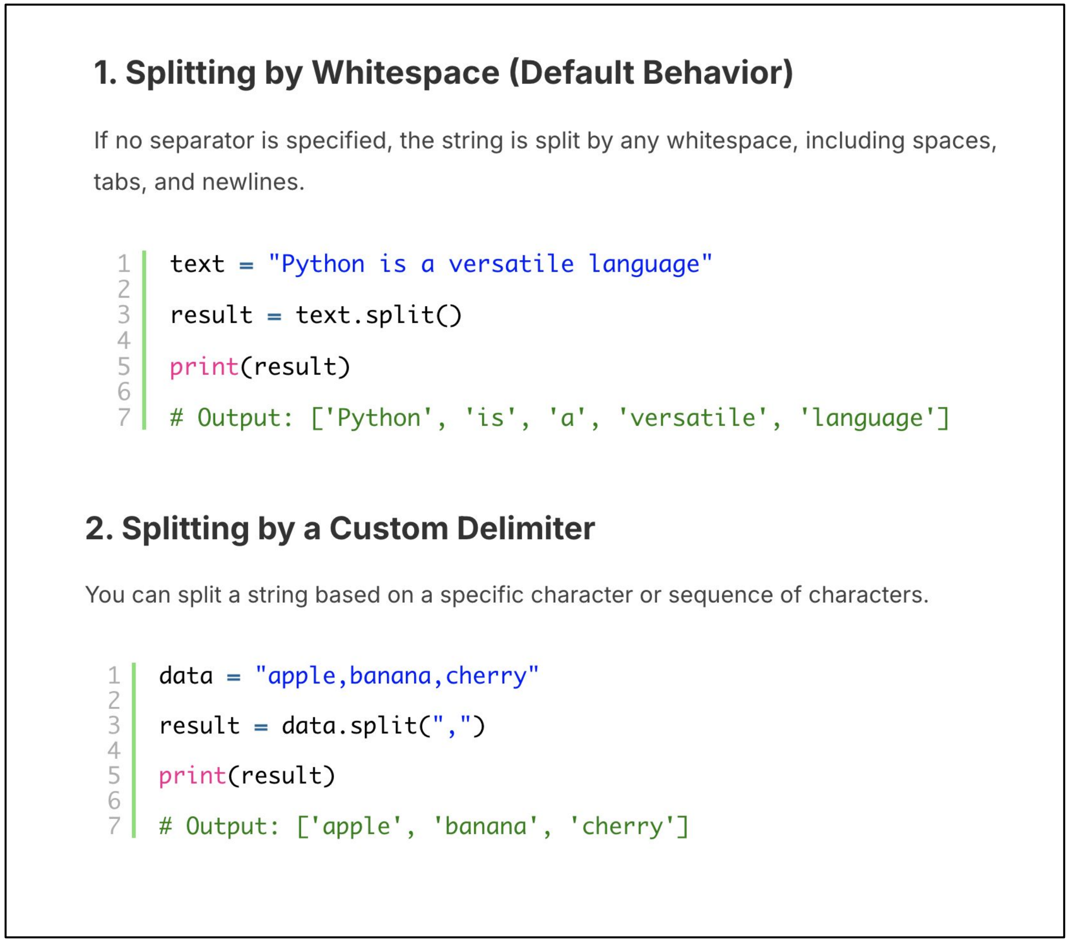
https://www.geeksforgeeks.org/python/working-csv-files-python/



```
>>> import csv
>>> with open('eggs.csv', newline='') as csvfile:
...     spamreader = csv.reader(csvfile, delimiter=' ', quotechar='|')
...     for row in spamreader:
...         print(', '.join(row))
Spam, Spam, Spam, Spam, Spam, Baked Beans
Spam, Lovely Spam, Wonderful Spam
```

https://docs.python.org/3/library/csv.html#module-csv

string split() method



1. Splitting by Whitespace (Default Behavior)

If no separator is specified, the string is split by any whitespace, including spaces, tabs, and newlines.

```python
text = "Python is a versatile language"

result = text.split()

print(result)

# Output: ['Python', 'is', 'a', 'versatile', 'language']
```

2. Splitting by a Custom Delimiter

You can split a string based on a specific character or sequence of characters.

```python
data = "apple,banana,cherry"

result = data.split(",")

print(result)

# Output: ['apple', 'banana', 'cherry']
```

https://www.mygreatlearning.com/blog/python-string-split-method/

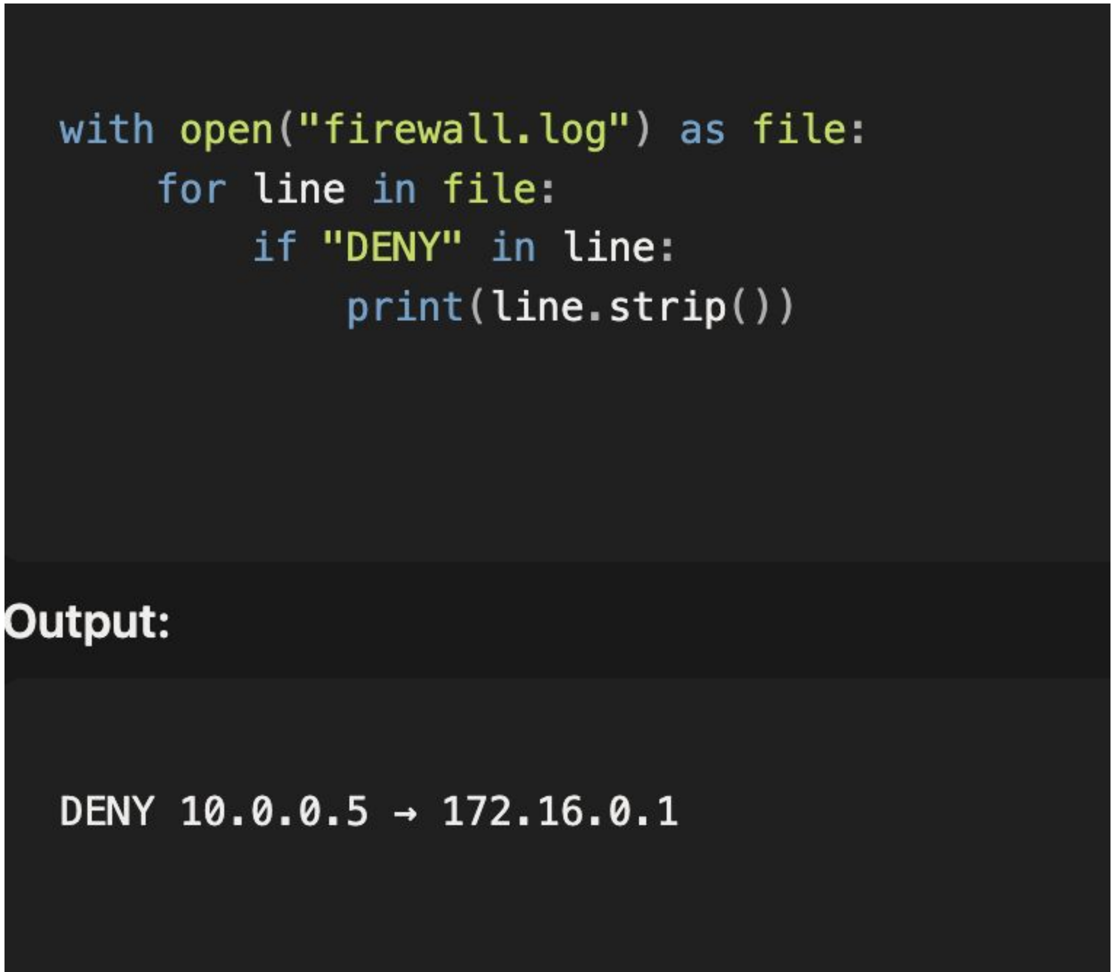
```
string.split(separator, maxsplit)
```

The **split()** method is used to break up a string into a list of substrings based on a specified delimiter. By default, it splits the string by whitespace, but you can specify any delimiter, such as a comma, semicolon, or custom character.

- **separator** (optional): Specifies the delimiter. The default is whitespace.
- **maxsplit** (optional): Specifies the maximum number of splits. Default is -1, which means “no limit.”

For Loops

- Used for iterating over a sequence
- Can execute a set of statements once for each item in a list, tuple, set etc.



```python
with open("firewall.log") as file:
    for line in file:
        if "DENY" in line:
            print(line.strip())
```

Output:

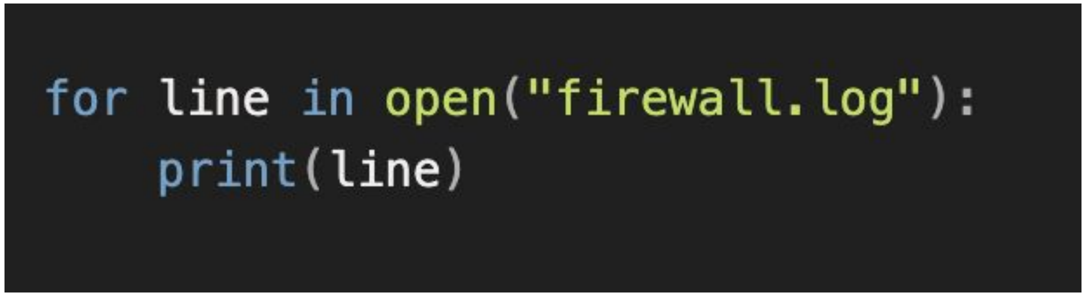
```
DENY 10.0.0.5 → 172.16.0.1
```

The `strip()` method removes any leading, and trailing whitespaces.

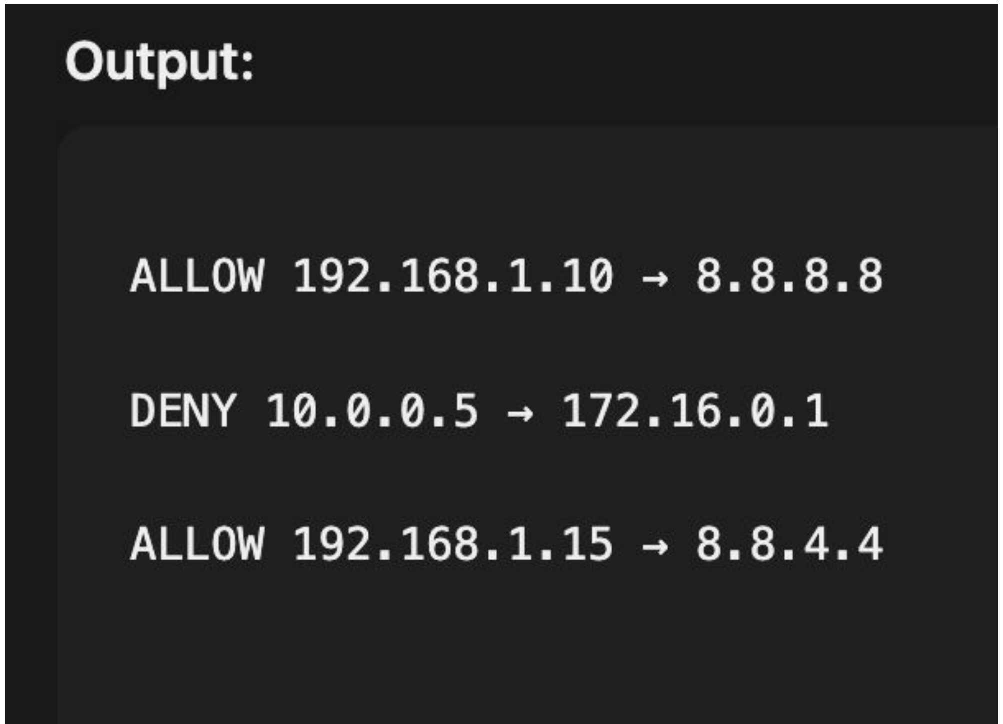
Leading means at the beginning of the string, trailing means at the end.

https://www.w3schools.com/python/ref_string_strip.asp

reading line by line



```
for line in open("firewall.log"):
    print(line)
```



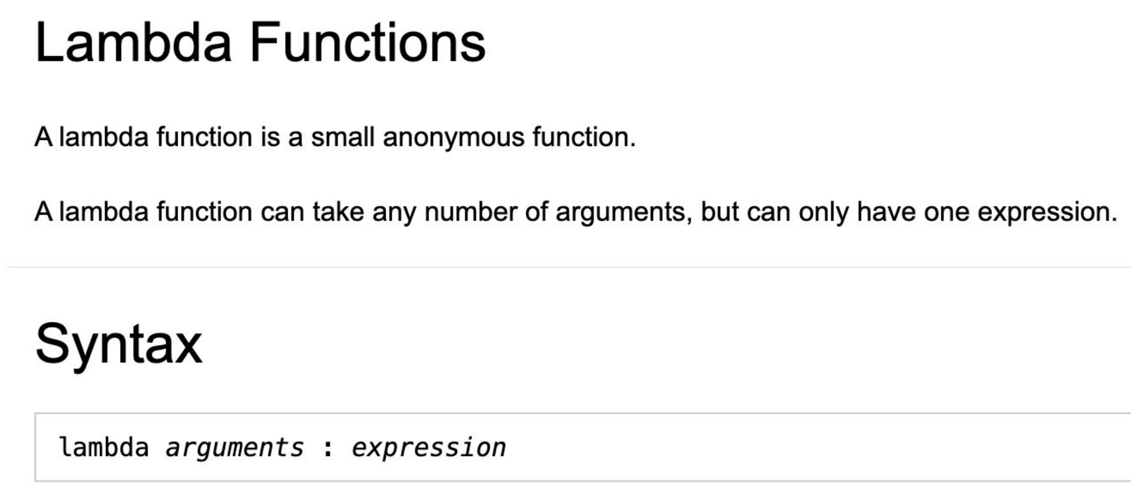
Output:

```
ALLOW 192.168.1.10 → 8.8.8.8

DENY 10.0.0.5 → 172.16.0.1

ALLOW 192.168.1.15 → 8.8.4.4
```

Lambda



Lambda Functions

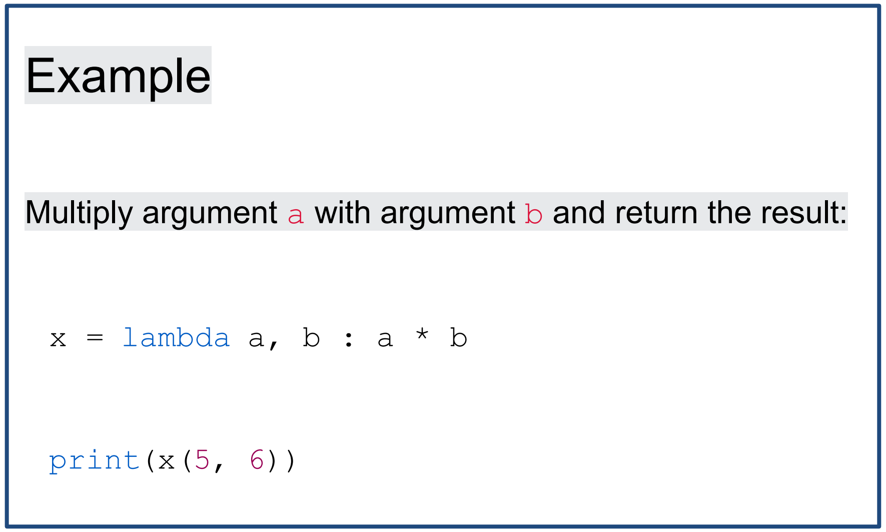
A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

Syntax

```
lambda arguments : expression
```

Lambda functions can take any number of arguments:



Example

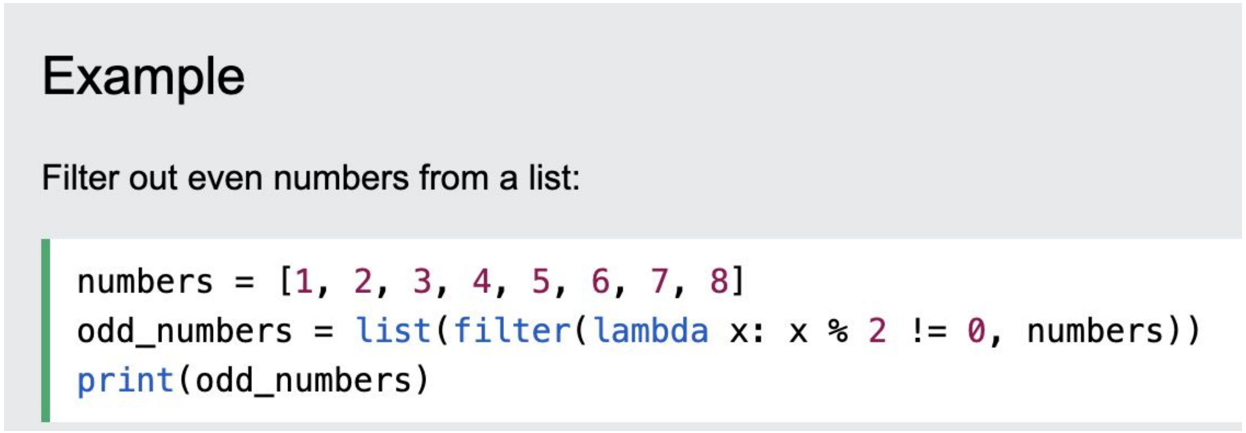
Multiply argument `a` with argument `b` and return the result:

```
x = lambda a, b : a * b
print(x(5, 6))
```

https://www.w3schools.com/python/python_lambda.asp

Lambda Built-In Functions

The `filter()` function creates a list of items for which a function returns `True`.

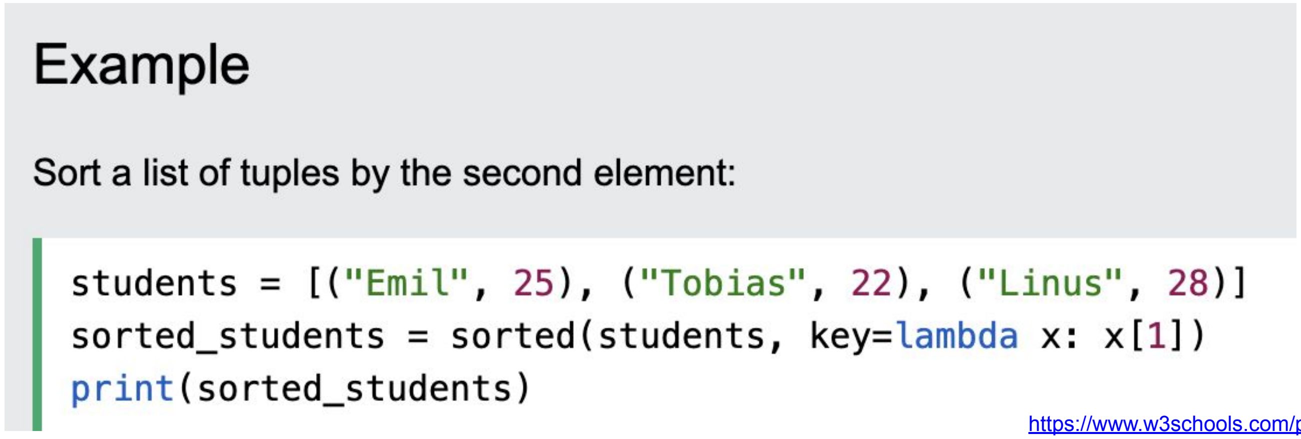


Example

Filter out even numbers from a list:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8]
odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
print(odd_numbers)
```

The `sorted()` function can use a lambda as a key for custom sorting.



Example

Sort a list of tuples by the second element:

```
students = [("Emil", 25), ("Tobias", 22), ("Linus", 28)]
sorted_students = sorted(students, key=lambda x: x[1])
print(sorted_students)
```

https://www.w3schools.com/python/python_lambda.asp

Projects (Suggestions)

- **Count DENY vs ACCEPT entries**
- **List 5 source IPs that generated the most DENY entries**
- **Find repeated logins**
- **Detect Port Scans**

Optional Challenges

1. Use Pandas
2. Create functions and use Python's filter()
3. Try String Methods - .upper(), .lower(), .title(), .replace()
4. Create a solution to a business problem

Share your work, if you’d like!