File Handling and Exception Handling

Course Unit 11: Week 14

Objectives:

1. Explain python file handling and exception handling.
2. Inspect what modules and packages need file handling and exception handling.
3. Prepare file handling and exception handling on the modules / packages.

File Handling in Python

Python File Handling

- allows users to handle files i.e., to **read** and **write** files, along with many other file handling options, to operate on files.
- Python treats files differently as text or binary and this is important. Each line of code includes a sequence of characters and they form a text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun.

Advantages of File Handling

Advantages of File Handling

Versatility

- allows you to perform a wide range of operations, such as creating, reading, writing, appending, renaming, and deleting files.

Flexibility

- allows you to work with different file types (e.g. text files, binary files, CSV files, etc.), and to perform different operations on files (e.g. read, write, append, etc.).

Advantages of File Handling

User–friendly

- Python provides a user-friendly interface for file handling, making it easy to create, read, and manipulate files.

Cross-platform

- Python file-handling functions work across different platforms (e.g. Windows, Mac, Linux), allowing for seamless integration and compatibility.

Disadvantages of File Handling

Disadvantages of File Handling

Error-prone

- if the code is not carefully written or if there are issues with the file system (e.g. file permissions, file locks, etc.).

Security risks

- if the program accepts user input that can be used to access or modify sensitive files on the system.

Disadvantages of File Handling

Complexity

- when working with more advanced file formats or operations. Careful attention must be paid to the code to ensure that files are handled properly and securely.

Performance

- can be slower than other programming languages, especially when dealing with large files or performing complex operations.

Functions used in File Handling

open()

read()

write()

open() Function in Python

- f = open(filename, mode)

Where the following mode is supported:

- r: open an existing file for a read operation.
- w: open an existing file for a write operation. If the file already contains some data then it will be overridden but if the file is not present then it creates the file as well.
- a: open an existing file for append operation. It won't override existing data.

open() Function in Python

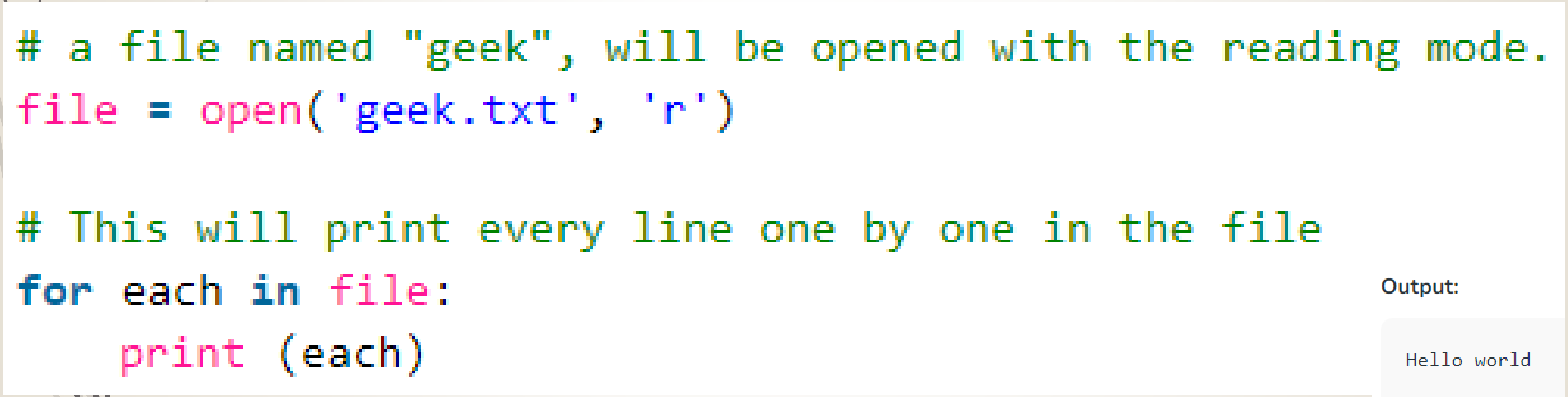
- f = open(filename, mode)

Where the following mode is supported:

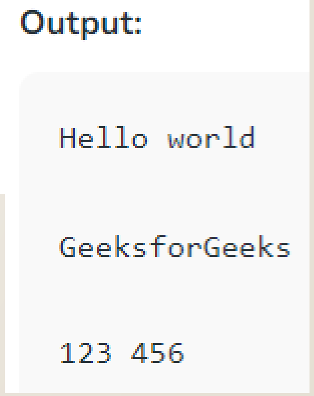
- **r+**: To read and write data into the file. The previous data in the file will be overridden.
- **w+**: To write and read data. It will override existing data.
- **a+**: To append and read data from the file. It won’t override existing data.

read a file in Python

- **Example 1:** The open command will open the file in the read mode and the for loop will print each line present in the file.



```python
# a file named "geek", will be opened with the reading mode.
file = open('geek.txt', 'r')

# This will print every line one by one in the file
for each in file:
    print (each)
```

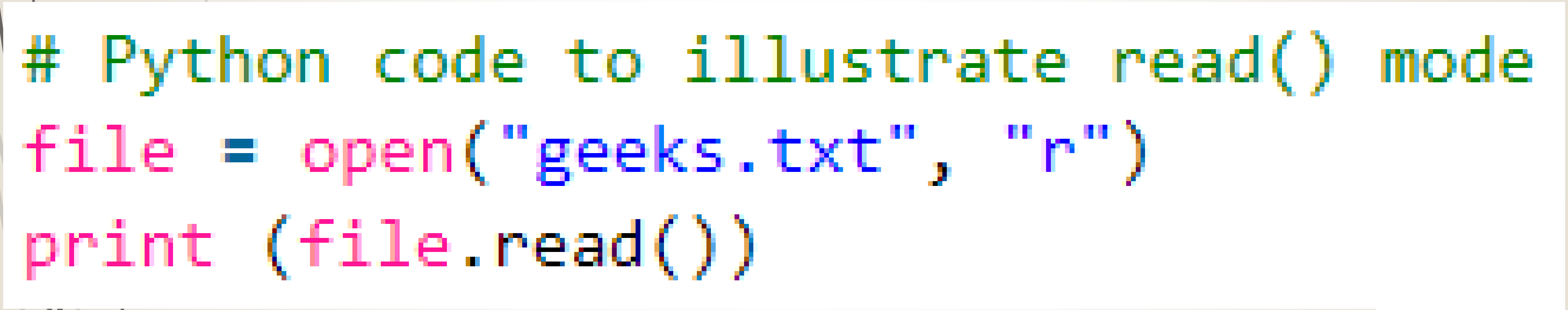


Output:

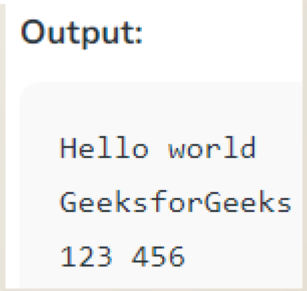
```
Hello world

GeeksforGeeks

123 456
```

read a file in Python

➢ **Example 2:** In this example, we will extract a string that contains all characters in the file then we can use **file.read().**



```python
# Python code to illustrate read() mode
file = open("geeks.txt", "r")
print (file.read())
```

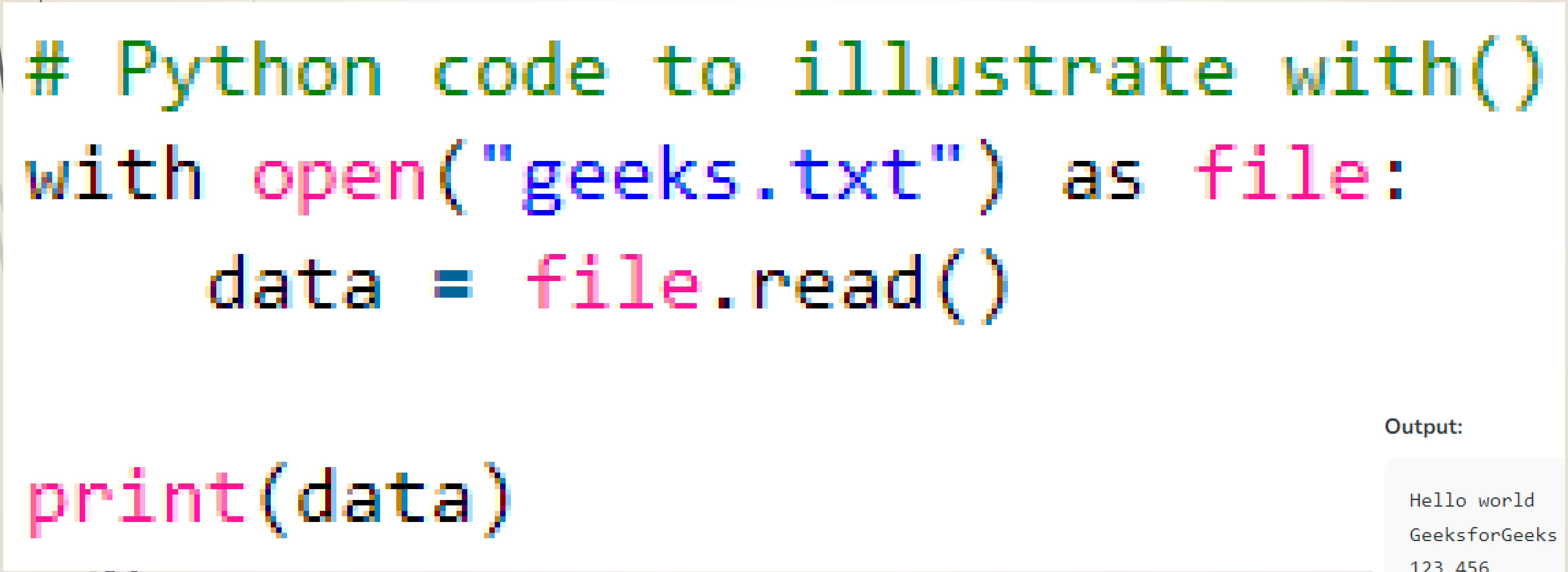


Output:

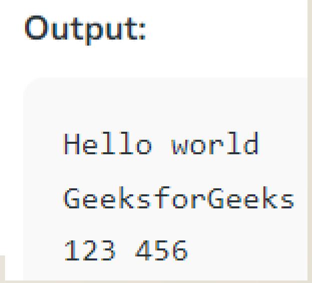
```
Hello world
GeeksforGeeks
123 456
```

read a file in Python

- **Example 3:** In this example, we will see how we can read a file using the **with statement.**



```python
# Python code to illustrate with()
with open("geeks.txt") as file:
    data = file.read()

print(data)
```

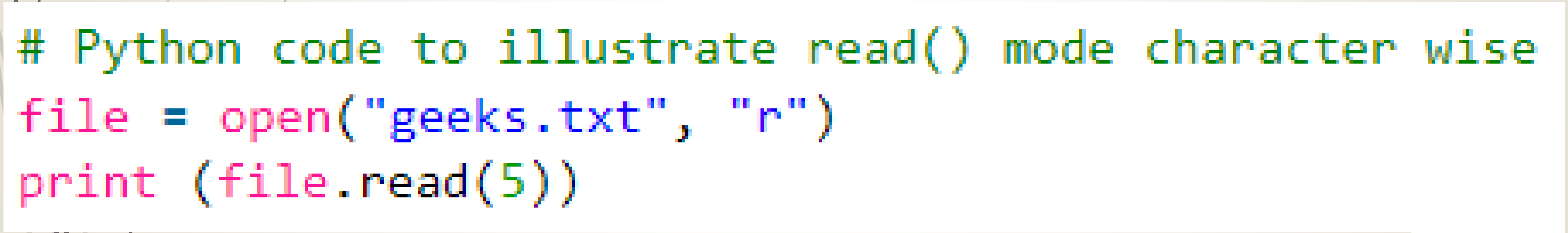


Output:

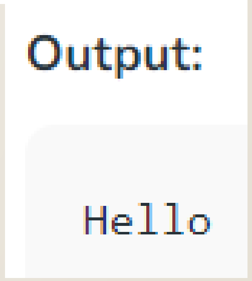
```
Hello world
GeeksforGeeks
123 456
```

read a file in Python

- **Example 4:** Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:



```
# Python code to illustrate read() mode character wise
file = open("geeks.txt", "r")
print (file.read(5))
```

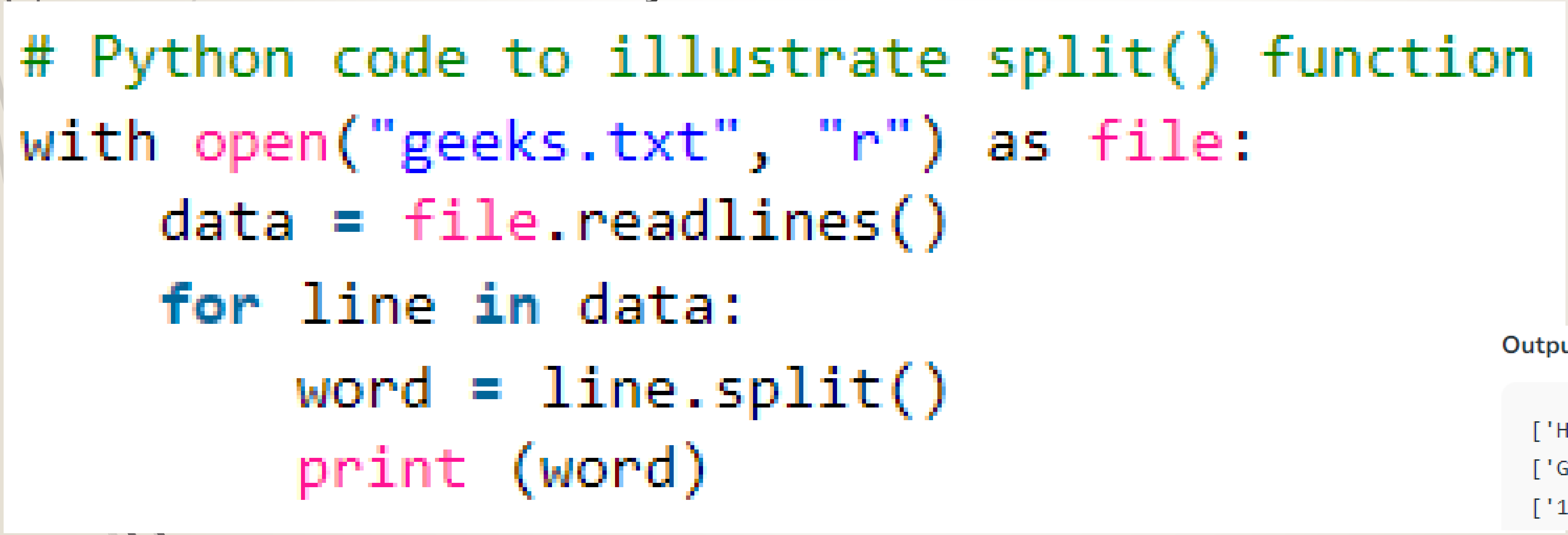


Output:

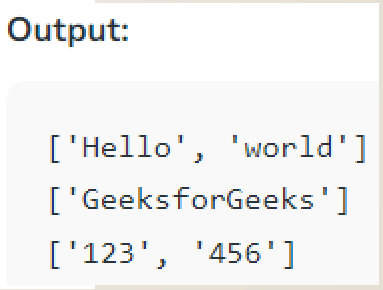
```
Hello
```

read a file in Python

- **Example 5:** We can also split lines while reading files in Python. The **split() function** splits the variable when space is encountered. You can also split using any characters as you wish.



```python
# Python code to illustrate split() function
with open("geeks.txt", "r") as file:
    data = file.readlines()
    for line in data:
        word = line.split()
        print (word)
```

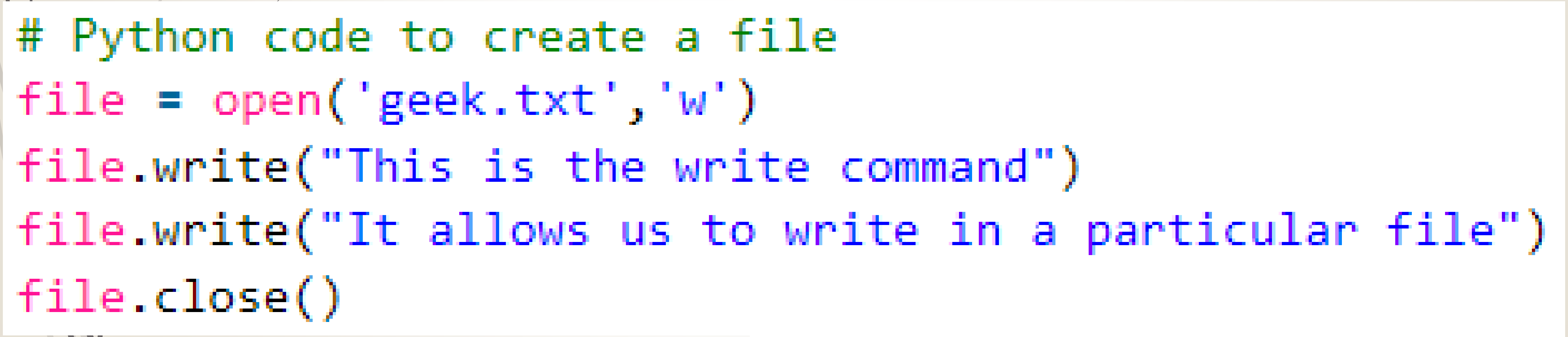


Output:

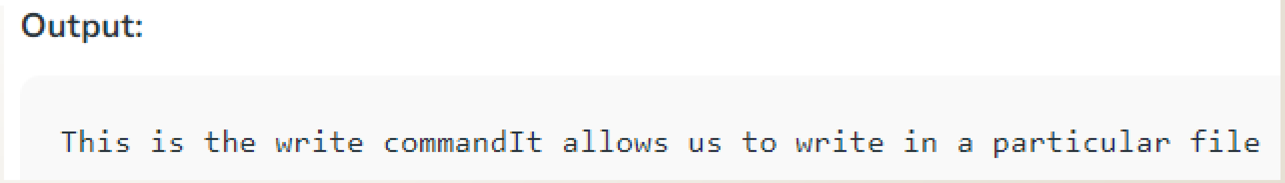
```
['Hello', 'world']
['GeeksforGeeks']
['123', '456']
```

File using the write() Function

- **Example 1:** In this example, we will see how the write mode and the write() function is used to write in a file. The close() command terminates all the resources in use and frees the system of this particular program.



```python
# Python code to create a file
file = open('geek.txt','w')
file.write("This is the write command")
file.write("It allows us to write in a particular file")
file.close()
```

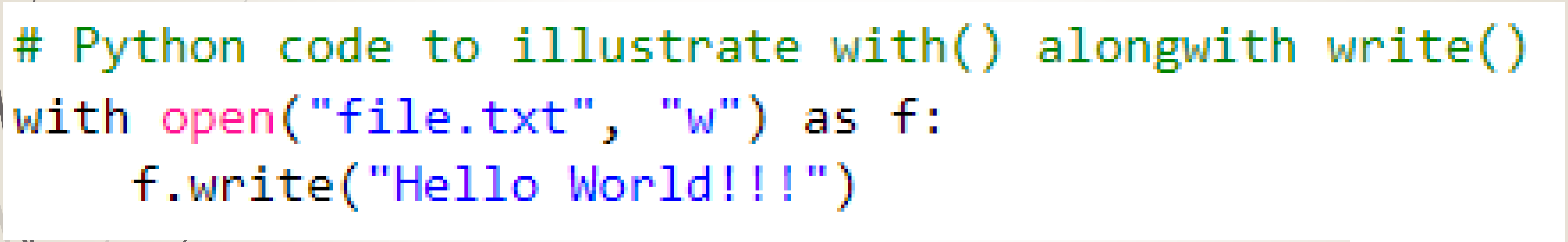


Output:

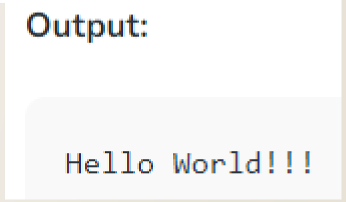
```
This is the write commandIt allows us to write in a particular file
```

File using the write() Function

- **Example 2:** We can also use the written statement along with the **with()** function.



```
# Python code to illustrate with() alongwith write()
with open("file.txt", "w") as f:
    f.write("Hello World!!!")
```

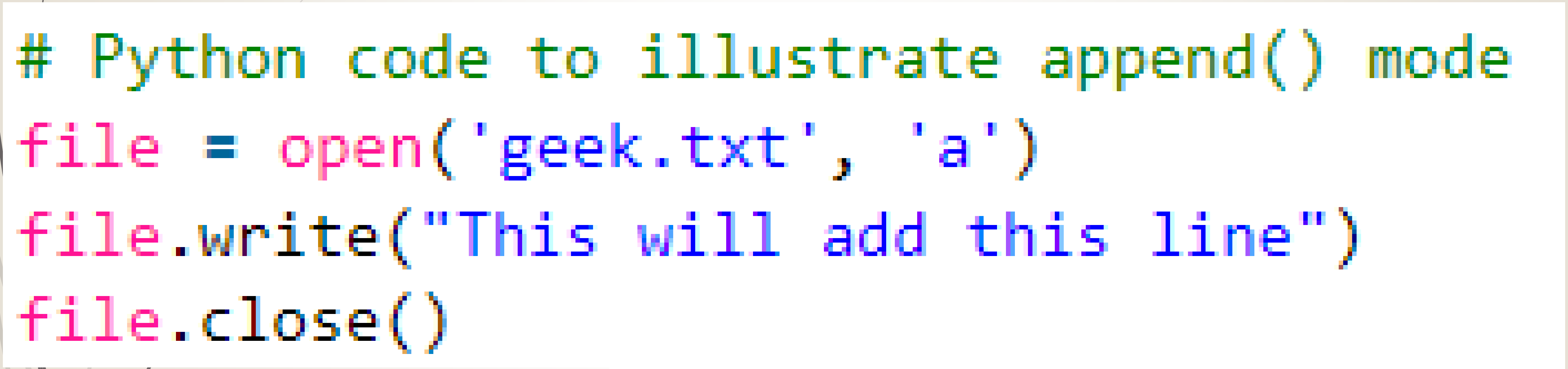


Output:

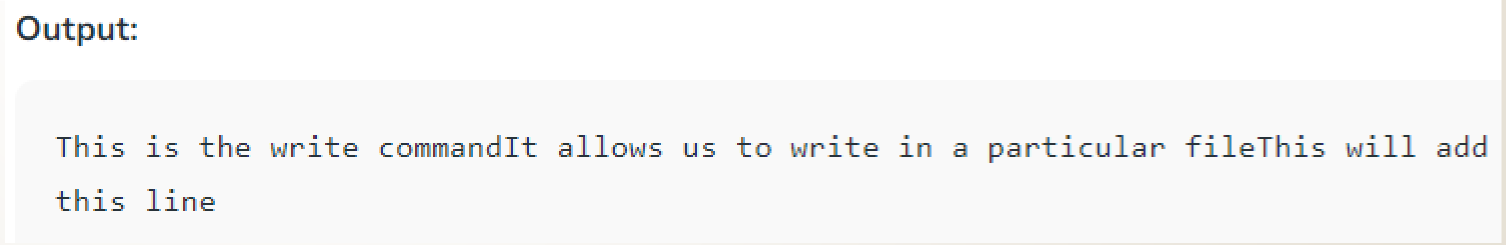
```
Hello World!!!
```

File using the write() Function

- Append Mode



```python
# Python code to illustrate append() mode
file = open('geek.txt', 'a')
file.write("This will add this line")
file.close()
```



Output:

```
This is the write commandIt allows us to write in a particular fileThis will add
this line
```

References:

- 8. Errors and Exceptions. (n.d.). https://docs.python.org/3/tutorial/errors.html
- File Handling in Python. (n.d.). https://www.geeksforgeeks.org/file-handling-python/
- File Handling in Python – How to Create, Read, and Write to a File. (2022).
- https://www.freecodecamp.org/news/file-handling-in-python/
- Learn Python Programming. (2023). https://www.tutorialsteacher.com/python
- Python Exception Handling. (n.d.). https://www.geeksforgeeks.org/python-exception-handling/
- Python Tutorial. (2022). https://www.w3resource.com/python/python-tutorial.php
- Python Tutorial. (n.d.). https://www.tutorialspoint.com/python/index.htm
- Python Tutorial. (n.d.). https://www.w3schools.com/python/default.asp