Artificial Intelligence Diploma

Python Session 5

NeuroTech

Agenda

1. File Handling
2. Read Data from File
3. Write Data into File
4. Handle Exception
5. Module

File Handling

NeuroTech

File Handling

File handling in Python refers to the process of working with files on the file system. It involves tasks such as reading from and writing to files, managing file access, and performing various operations on files. Python provides built-in functions and methods to facilitate file handling operations.

General Format:

file=open('filename.txt',mode)

- filename ='Name of file that you need to read data from it or write data on it.'
- mode=Refers to the specific purpose for which you are opening a file. It defines how you intend to interact with the file, whether it's for reading, writing, appending, or working with binary data.

File Handling

Mode :

1. Read Mode ('r’):

- Opens the file for reading (default mode).
- The file pointer is positioned at the beginning of the file.
- If the file does not exist, a FileNotFoundError is raised.

2. Write Mode ('w’):

- Opens the file for writing.
- If the file already exists, its contents are truncated (erased).
- If the file does not exist, a new empty file is created.
- The file pointer is positioned at the beginning of the file.

File Handling

Mode :

3. Append Mode ('a’):

- Opens the file for writing, but appends new data to the end of the file.
- If the file already exists, the existing content is preserved.
- If the file does not exist, a new empty file is created.
- The file pointer is positioned at the end of the file.

4. Binary Mode ('b’):

- Opens the file in binary mode.
- This mode is used for non-text files, such as images or audio files.
- For example, 'rb' for reading binary, 'wb' for writing binary.

File Handling

Mode :

5. Read and Write ('r+’):
- Opens the file for both reading and writing.
- The file pointer is positioned at the beginning of the file.
- Raises FileNotFoundError if the file does not exist.

6. Write and Read ('w+’):
- Opens the file for both writing and reading.
- If the file already exists, its contents are truncated (erased).
- If the file does not exist, a new empty file is created.
- The file pointer is positioned at the beginning of the file.

File Handling

Mode :

7.Append and Read ('a+’):

- Opens the file for both writing (appending) and reading.
- If the file already exists, the existing content is preserved.
- If the file does not exist, a new empty file is created.
- The file pointer is positioned at the end of the file.

Read Data From File

NeuroTech

File Handling

- To read data from files in Python, you can use the open() function along with various methods to read the content. Here are some common methods to read data from files.

1. Read the Entire File:

```python
with open('file.txt', 'r') as file:
    content = file.read()
print(content)
```

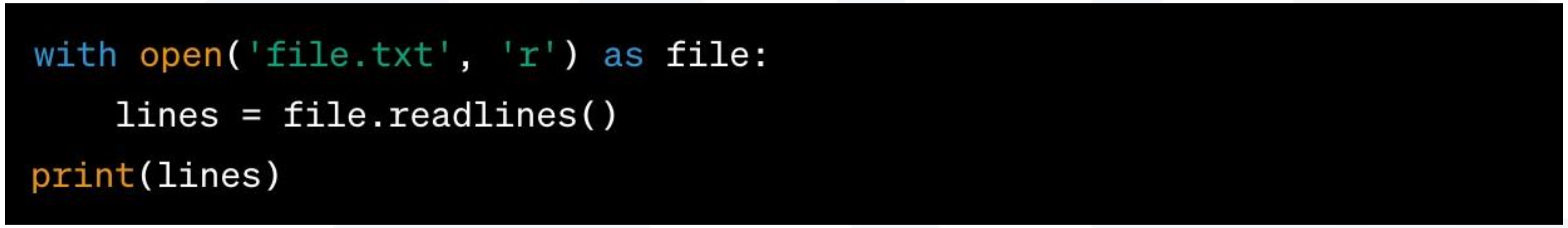
2. Read Line by Line:

```python
with open('file.txt', 'r') as file:
    for line in file:
        print(line)
```

NeuroTech

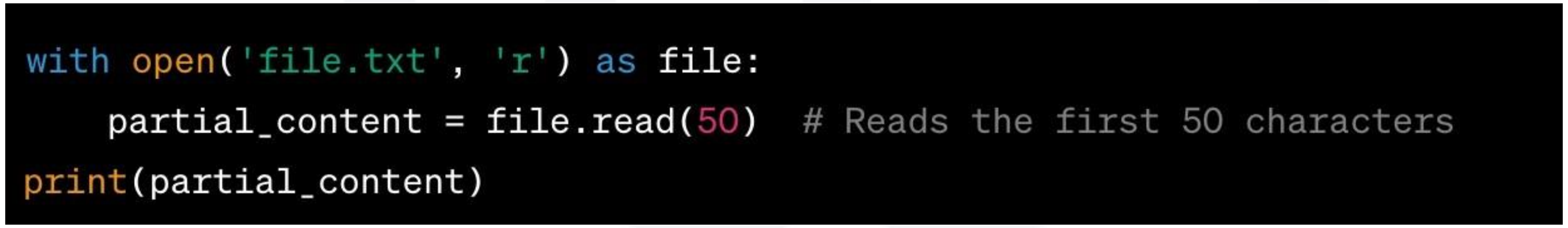
File Handling

3. Read Lines into a List



```
with open('file.txt', 'r') as file:
    lines = file.readlines()
print(lines)
```

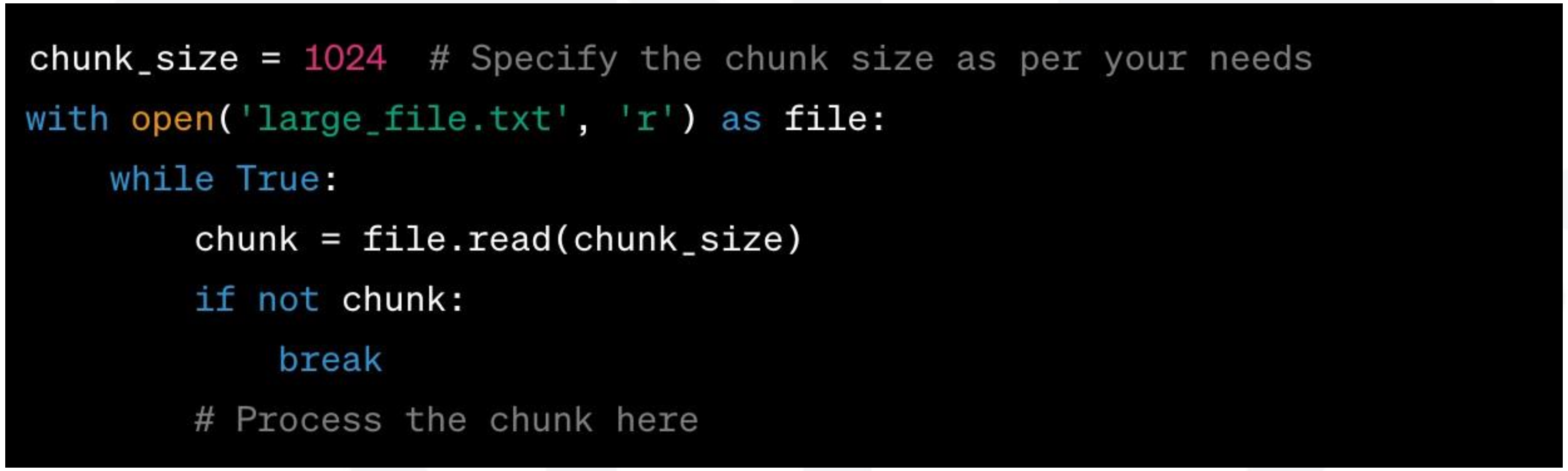
4. Read a Specific Number of Characters:



```
with open('file.txt', 'r') as file:
    partial_content = file.read(50)  # Reads the first 50 characters
print(partial_content)
```

File Handling

7. Read and Process Large Files Chunk by Chunk:



```
chunk_size = 1024  # Specify the chunk size as per your needs
with open('large_file.txt', 'r') as file:
    while True:
        chunk = file.read(chunk_size)
        if not chunk:
            break
        # Process the chunk here
```

Write Data into File

NeuroTech

File Handling

- To write data into files in Python, you can use the open() function along with the write() method. Here are some examples of how to write data into files:

1. Write Text to a File:

```
with open('output.txt', 'w') as file:
    file.write('Hello, world!\n')
    file.write('This is a new line.')
```

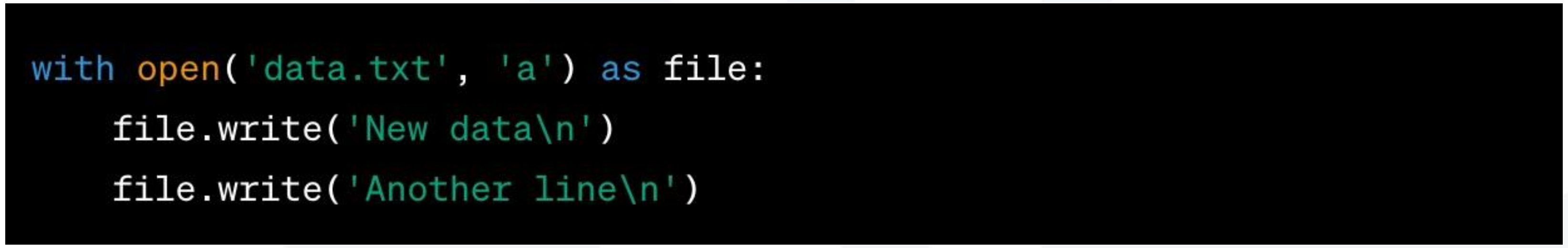
2. Write Multiple Lines to a File:

```
lines = ['Line 1', 'Line 2', 'Line 3']
with open('output.txt', 'w') as file:
    for line in lines:
        file.write(line + '\n')
```

NeuroTech

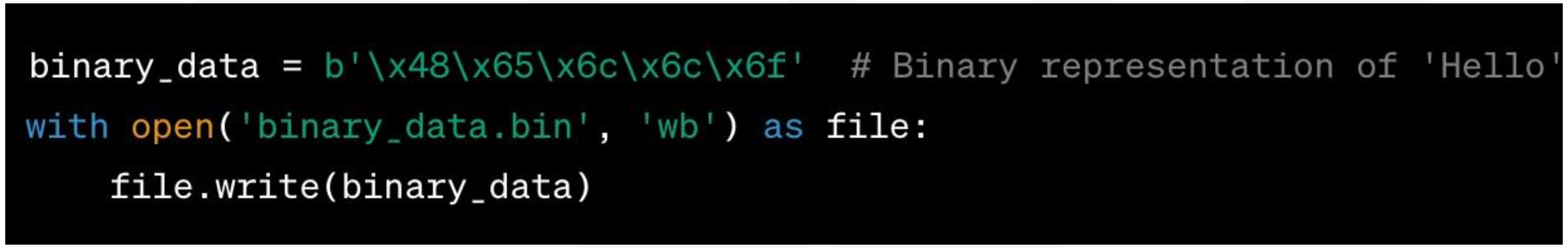
File Handling

3. Append Data to an Existing File:



```python
with open('data.txt', 'a') as file:
    file.write('New data\n')
    file.write('Another line\n')
```

4. Write Binary Data to a File:

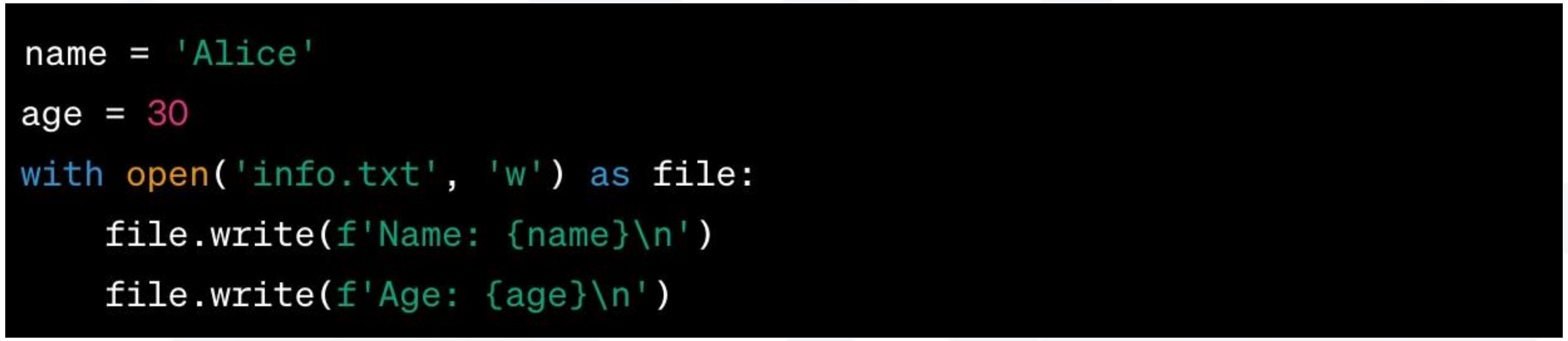


```python
binary_data = b'\x48\x65\x6c\x6c\x6f'  # Binary representation of 'Hello'
with open('binary_data.bin', 'wb') as file:
    file.write(binary_data)
```

NeuroTech

File Handling

5. Write Formatted Data to a File:



```python
name = 'Alice'
age = 30
with open('info.txt', 'w') as file:
    file.write(f'Name: {name}\n')
    file.write(f'Age: {age}\n')
```

Handling Exception

NeuroTech

File Handling

Exception handling is crucial when working with files in Python. It helps your program gracefully handle errors that may occur during file operations. Here's how you can handle exceptions in file handling:

1. **Using try.. Except :**

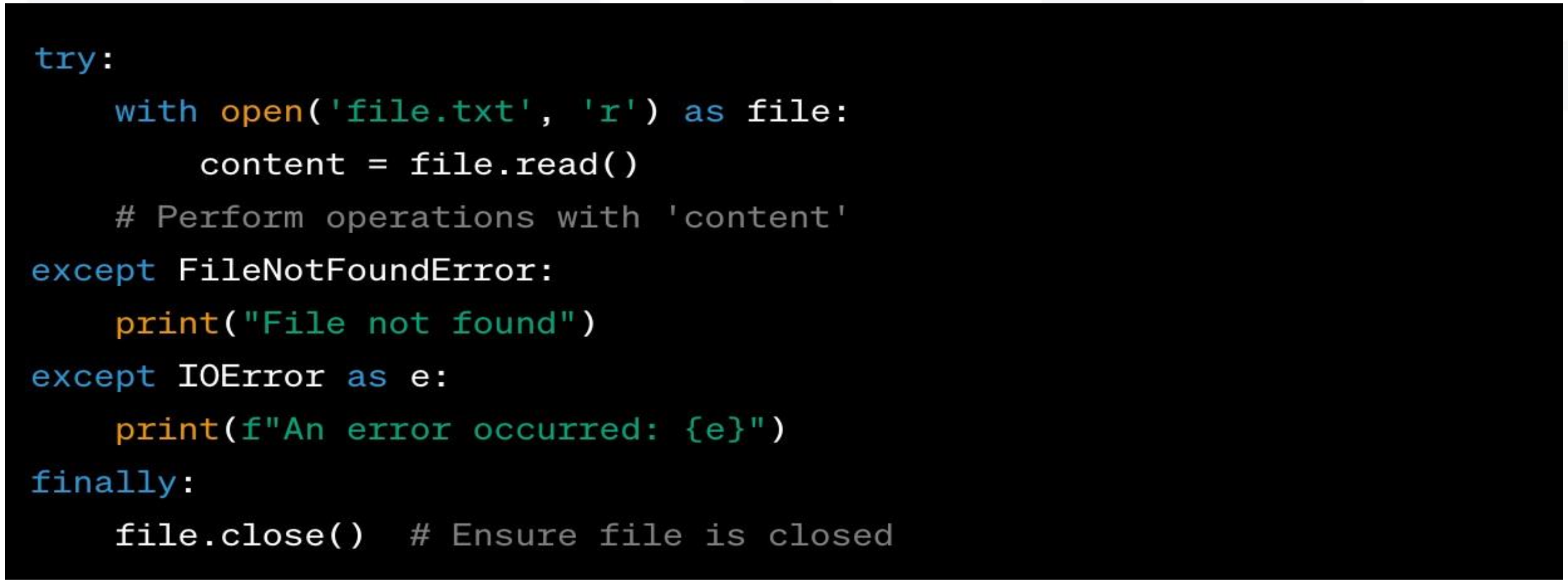
```python
try:
    with open('file.txt', 'r') as file:
        content = file.read()
    # Perform operations with 'content'
except FileNotFoundError:
    print("File not found")
except IOError as e:
    print(f"An error occurred: {e}")
```

In this example, the program tries to open and read the file. If the file does not exist (raises FileNotFoundError) or if any other input/output error occurs (raises IOError), the corresponding except block will be executed.

File Handling

2. Using finally:

The finally block is used to execute code that must run regardless of whether an exception occurred. It's often used for cleanup tasks like closing the file.



```python
try:
    with open('file.txt', 'r') as file:
        content = file.read()
    # Perform operations with 'content'
except FileNotFoundError:
    print("File not found")
except IOError as e:
    print(f"An error occurred: {e}")
finally:
    file.close()  # Ensure file is closed
```

File Handling

3. Using else:

The else block is executed if no exceptions are raised in the try block.

```python
try:
    with open('file.txt', 'r') as file:
        content = file.read()
except FileNotFoundError:
    print("File not found")
except IOError as e:
    print(f"An error occurred: {e}")
else:
    print("File read successfully:", content)
finally:
    file.close()  # Ensure file is closed
```

Modules

NeuroTech

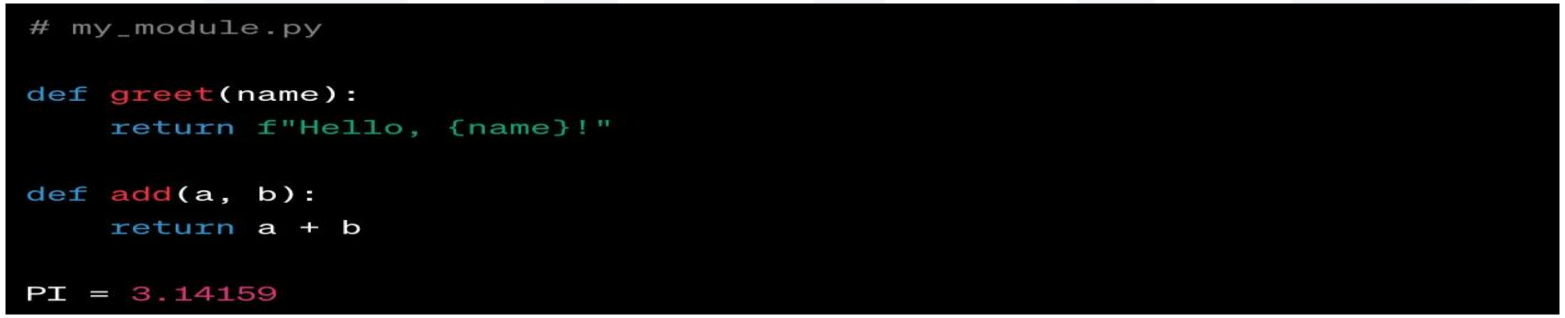
Module

- In Python, a "module" refers to a file containing Python definitions and statements. These files can include functions, classes, and variables, which can be used in other Python programs by importing the module. Modules allow you to organize your code into reusable and manageable pieces, enhancing code readability, maintenance, and collaboration.

- Here's a more detailed explanation of modules in Python:

1. Creating a Module:

To create a module, you simply write your Python code in a .py file. For example, if you create a file named my_module.py, you can define functions, classes, and variables within it.



```python
# my_module.py

def greet(name):
    return f"Hello, {name}!"

def add(a, b):
    return a + b

PI = 3.14159
```

NeuroTech

Module

3. **Using a Module**:

To use the definitions from a module in another Python script or program, you need to import the module using the import statement.

```python
# main_program.py
import my_module

result = my_module.add(5, 3)
print(result)
```

Alternatively, you can import specific definitions from a module using the from ... import statement.

```python
# main_program.py
from my_module import greet

message = greet("Alice")
print(message)
```

Module

1. Standard Library Modules:

- Python comes with a rich standard library that includes various pre-built modules for common tasks, such as working with files, processing strings, performing mathematical operations, and more. You can directly use these modules in your programs.

2. Third-Party Modules:

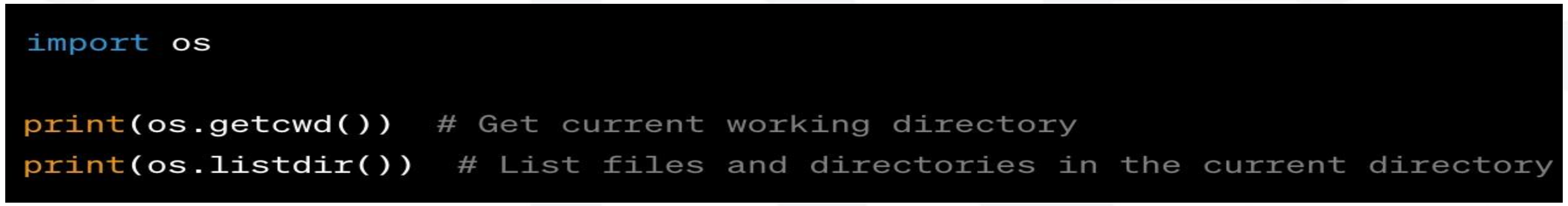
- In addition to the standard library, there are thousands of third-party modules available that you can install and use in your projects. These modules can provide specialized functionality, such as web frameworks, data manipulation, machine learning, and more. You can install third-party modules using tools like pip.

Module

Standard Library Modules:

- Certainly, here are some common standard libraries and their examples of usage in Python:

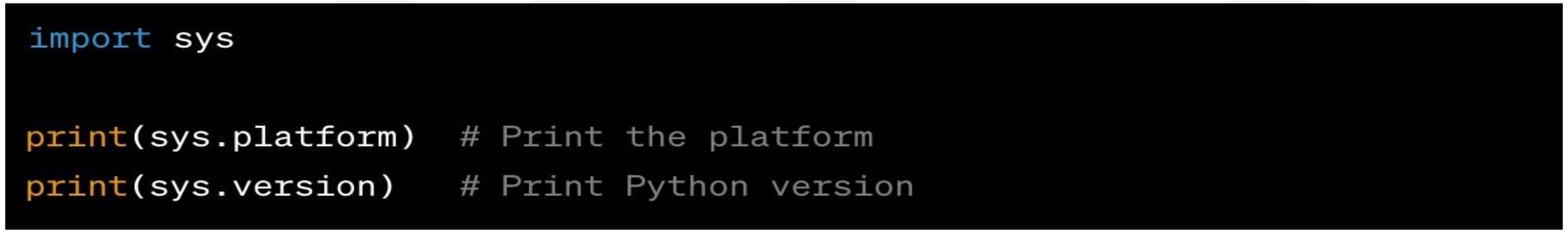
1. **‘os’ and ‘os.path’**: Operating system interfaces for file and directory manipulation.



```python
import os

print(os.getcwd())  # Get current working directory
print(os.listdir())  # List files and directories in the current directory
```

2. **‘sys’**: System-specific parameters and functions.



```python
import sys

print(sys.platform)  # Print the platform
print(sys.version)   # Print Python version
```

Module

Standard Library Modules:

3 . ‘**math**’: Mathematical functions and constants.

```python
import math

print(math.sqrt(25))  # Calculate square root
print(math.pi)        # Print the value of pi
```

4. ‘**datetime**’: Date and time manipulation.

```python
from datetime import datetime

now = datetime.now()
print(now.strftime('%Y-%m-%d %H:%M:%S'))  # Format current date and time
```

Module

Standard Library Modules:

5 . ‘**random**’: Generate pseudo-random numbers.

```python
import random

print(random.randint(1, 6))  # Generate a random integer between 1 and 6
```

6. ‘**json**’: Encode and decode JSON data.

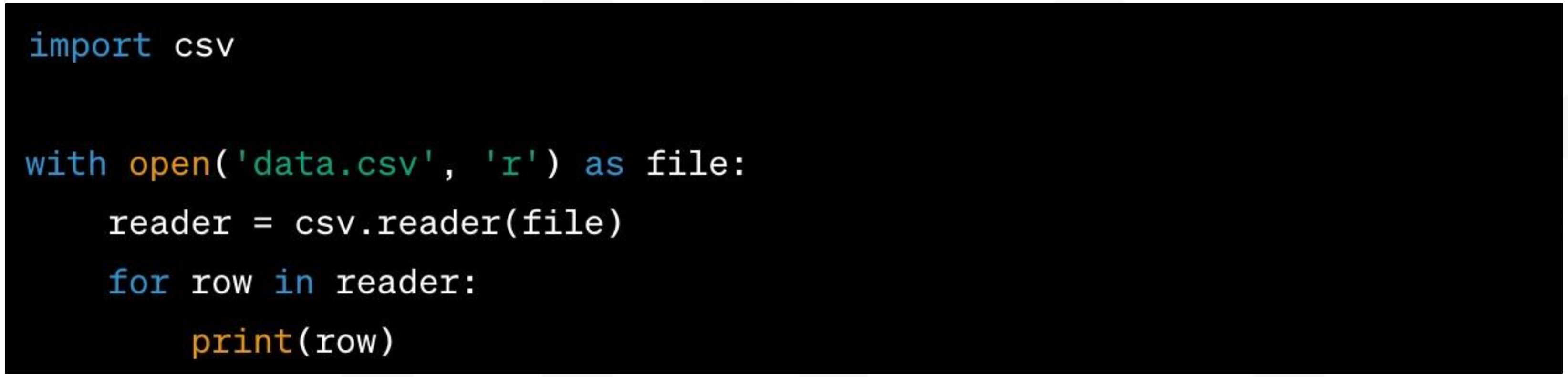
```python
import json

data = {'name': 'Alice', 'age': 30}
json_data = json.dumps(data)  # Convert dictionary to JSON string
```

Module

Standard Library Modules:

7 . ‘**csv**’: CSV (Comma Separated Values) file reading and writing..



```python
import csv

with open('data.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

Any Question

NeuroTech

Thanks

NeuroTech