1. What advantages do Excel spreadsheets have over CSV spreadsheets?

2.What do you pass to csv.reader() and csv.writer() to create reader and writer objects?

3. What modes do File objects for reader and writer objects need to be opened in?

4. What method takes a list argument and writes it to a CSV file?

5. What do the keyword arguments delimiter and line terminator do?

6. What function takes a string of JSON data and returns a Python data structure?

7. What function takes a Python data structure and returns a string of JSON data?

ANSWER

1. \*\*Advantages of Excel Spreadsheets over CSV Spreadsheets:\*\*

- Excel spreadsheets can contain multiple sheets or tabs within a single file, allowing you to organize and structure data more effectively.

- Excel supports various data types, including numbers, text, dates, and formulas, with more advanced calculations and functions.

- Excel provides advanced formatting and styling options for data presentation, including charts, graphs, and conditional formatting.

- Excel allows for cell merging, cell comments, and other features for data clarity and collaboration.

- Excel can handle more complex data validation rules and data analysis tools.

2. \*\*Creating CSV Reader and Writer Objects:\*\*

To create reader and writer objects using the `csv` module in Python, you pass a file-like object (typically a file opened in the appropriate mode) to `csv.reader()` and `csv.writer()`.

Example of creating reader and writer objects:

```python

import csv

# Creating a reader object from a file

with open('data.csv', 'r') as file:

csv\_reader = csv.reader(file)

# Creating a writer object to write to a file

with open('output.csv', 'w', newline='') as file:

csv\_writer = csv.writer(file)

```

3. \*\*File Object Modes for Reader and Writer:\*\*

- For reader objects (`csv.reader()`): Open the file in read mode ('r').

- For writer objects (`csv.writer()`): Open the file in write mode ('w'). It is also good practice to open the file in binary write mode ('wb') if you are writing binary data.

4. \*\*Method for Writing a List to a CSV File:\*\*

To write a list to a CSV file, you can use the `writerow()` method of the CSV writer object. Here's an example:

```python

data\_to\_write = ['John', 'Doe', 'johndoe@email.com']

csv\_writer.writerow(data\_to\_write)

```

5. \*\*Keyword Arguments `delimiter` and `lineterminator`:\*\*

- `delimiter`: This keyword argument specifies the character used to separate fields (columns) in the CSV file. By default, it is a comma (','), but you can change it to another character, such as a tab ('\t') or a semicolon (';').

- `lineterminator`: This keyword argument specifies the character(s) used to terminate lines (rows) in the CSV file. By default, it is the newline character ('\n'), but you can change it if needed.

6. \*\*Function for Converting JSON Data to a Python Data Structure:\*\*

You can use the `json.loads()` function to convert a string containing JSON data into a Python data structure (e.g., a dictionary, list, etc.). For example:

```python

import json

json\_data = '{"name": "John", "age": 30, "city": "New York"}'

python\_data = json.loads(json\_data)

```

7. \*\*Function for Converting a Python Data Structure to JSON Data:\*\*

You can use the `json.dumps()` function to convert a Python data structure into a JSON-formatted string. For example:

```python

import json

python\_data = {"name": "John", "age": 30, "city": "New York"}

json\_data = json.dumps(python\_data)

```