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

Excel spreadsheets have several advantages over CSV spreadsheets:

1. \*\*Formatting and Styling\*\*: Excel supports cell formatting (fonts, colors, borders), which can enhance readability and presentation.

2. \*\*Formulas and Functions\*\*: Excel allows the use of complex formulas, functions, and calculations within the cells.

3. \*\*Multiple Sheets\*\*: Excel files can contain multiple sheets within a single workbook, allowing for better organization of related data.

4. \*\*Charts and Graphs\*\*: Excel can create a variety of charts and graphs from the data, providing visual insights.

5. \*\*Data Validation\*\*: Excel supports data validation to ensure that only specific types of data are entered.

6. \*\*Macros and Automation\*\*: Excel supports macros and VBA (Visual Basic for Applications) for automation of repetitive tasks.

7. \*\*Data Filtering and Pivot Tables\*\*: Excel offers powerful data filtering, sorting, and the ability to create pivot tables for data analysis.

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

You pass a file object to `csv.reader()` and `csv.writer()` to create reader and writer objects.

#### Example:

```python

import csv

# Creating a reader object

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

reader = csv.reader(file)

# Creating a writer object

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

writer = csv.writer(file)

```

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

- \*\*Reader objects\*\*: The file needs to be opened in read mode (`'r'`).

- \*\*Writer objects\*\*: The file needs to be opened in write mode (`'w'`).

For both, it's recommended to use the `newline=''` argument to prevent issues with newline characters.

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

The `writerow()` method of a `csv.writer` object takes a list argument and writes it to a CSV file.

#### Example:

```python

import csv

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

writer = csv.writer(file)

writer.writerow(['Name', 'Age', 'City']) # Writes a single row

```

### 5. What do the keyword arguments `delimiter` and `lineterminator` do?

- \*\*`delimiter`\*\*: Specifies the character that separates the fields in a CSV file. The default delimiter is a comma (`,`).

- \*\*`lineterminator`\*\*: Specifies the string that is used to terminate lines in the CSV file. The default is `'\r\n'` (carriage return and newline).

#### Example:

```python

import csv

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

writer = csv.writer(file, delimiter=';', lineterminator='\n')

writer.writerow(['Name', 'Age', 'City'])

```

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

The `json.loads()` function takes a string of JSON data and returns a Python data structure.

#### Example:

```python

import json

json\_string = '{"name": "Alice", "age": 30, "city": "New York"}'

data = json.loads(json\_string)

print(data) # Output: {'name': 'Alice', 'age': 30, 'city': 'New York'}

```

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

The `json.dumps()` function takes a Python data structure and returns a string of JSON data.

#### Example:

```python

import json

data = {'name': 'Alice', 'age': 30, 'city': 'New York'}

json\_string = json.dumps(data)

print(json\_string) # Output: '{"name": "Alice", "age": 30, "city": "New York"}'

```