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

**Answer:**

Advantages of Excel spreadsheets over CSV spreadsheets include:

1. Formatting: Excel allows for formatting of text, numbers, and cells, whereas CSV files only contain plain text and lack any formatting options. This means that Excel spreadsheets can have more visually appealing and organized data.

2. Complex calculations: Excel has built-in formulas and functions that can perform complex calculations, whereas CSV files do not have any built-in functionality for calculations.

3. Multiple worksheets: Excel allows for multiple worksheets within a single file, which can be useful for organizing large amounts of data. CSV files only contain a single worksheet.

4. Data validation: Excel has data validation options that allow you to set rules and constraints on the data entered into cells, such as requiring a specific format or restricting certain values. This helps to ensure the accuracy and consistency of the data.

5. Charts and graphs: Excel has built-in chart and graph tools that allow you to visualize data, whereas CSV files do not have this capability.

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

**Answer:** To create reader and writer objects in Python's csv module, you need to pass a file object to their constructors, csv.reader() and csv.writer(), respectively.For example,

**To create a reader object, you can pass an open file object to csv.reader() like this:**

import csv

with open('my\_file.csv', newline='') as csvfile:

reader = csv.reader(csvfile)

for row in reader:

print(row)

**To create a writer object, you can also pass an open file object to csv.writer(), like this:**

import csv

with open('my\_file.csv', 'w', newline='') as csvfile:

writer = csv.writer(csvfile)

writer.writerow(['Column 1', 'Column 2', 'Column 3'])

writer.writerow(['Value 1', 'Value 2', 'Value 3'])

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

**Answer**: To create reader and writer objects in Python's csv module, you need to pass a file object to their constructors, csv.reader() and csv.writer(), respectively. The file object should be opened in the appropriate mode for reading or writing CSV files.

For reading a CSV file using csv.reader(), the file object should be opened in text mode ('r').

For writing to a CSV file using csv.writer(), the file object should be opened in text mode ('w').

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

**ANSWER:** To write data to a CSV file using Python's csv module, you can use the writerow() method of a csv.writer object. This method takes a list as an argument and writes the elements of the list as a single row to the CSV file.

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

**Answer: delimiter:** specifies the character used to separate fields in the output CSV file. The default value is `,` (comma). For example, if you want to use a semicolon (;) as a delimiter instead of a comma, you can pass it as a keyword argument.

**Lineterminator :** specifies the character used to terminate rows in the output CSV file. The default value is `'\r\n'` on Windows, and `'\n'` on other platforms. For example, if you want to use a carriage return ('\r') as a line terminator instead of a newline character ('\n'), you can pass it as a keyword argument.

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

**Answer:** The json.loads() function can be used to parse a string of JSON data and convert it into a corresponding Python data structure. The function takes a JSON-formatted string as input and returns a Python object.

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

**Answer:** The json.dumps() function can be used to serialize a Python data structure into a corresponding JSON-formatted string. The function takes a Python object as input and returns a JSON-formatted string.