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

Answer=

* It is a binary file that holds information about all the worksheets in a workbook
* An Excel not only stores data but can also do operations on the data
* Files saved in excel cannot be opened or edited by text editors
* large files user is much easier in Excel for the end user. Also, you can have additional functions like selecting individual cells for import, convert dates and time automatically, reading formulas and their results, filters, sorting, etc
* Apart from text, data can also be stored in form of charts and graphs
* Excel can connect to external data sources to fetch data. You can use custom add-in in Excel to increase its functionality.
* Excel allows for Review of Data with detailed tracking and commenting feature
* In Excel, spreadsheets can have values of data types other than strings; cells can have different fonts, sizes, or color settings; cells can have varying widths and heights; adjacent cells can be merged

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

Answer=

To create a csv.reader object, you would pass an open file object, such as one returned by open(filename, 'r'), to the csv.reader() function.

Similarly, to create a csv.writer object, you would pass an open file object, such as one returned by open(filename, 'w'), to the csv.writer() function.

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

Answer=

To create a csv.reader object, the file should be opened in "reading" mode, which is represented by the string 'r' passed as the second argument to the open() function. For example, open(filename, 'r').

To create a csv.writer object, the file should be opened in "writing" mode, which is represented by the string 'w' passed as the second argument to the open() function. For example, open(filename, 'w').

It's also worth noting that you can pass 'a' as the second argument to open() function to open the file in "append" mode, which can be used to add new rows to a existing csv file.

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

Answer=

The csv.writer.writerow() method takes a list as an argument and writes it to a CSV file as a new row. This method is used in conjunction with a csv.writer object, which is created by passing an open file object (opened in write mode) to the csv.writer() function.

import csv  
  
data = [['Name', 'Age', 'Gender'], ['Alice', 25, 'Female'], ['Bob', 30, 'Male']]  
  
with open('data1.csv', 'w', newline='') as f:  
 csv\_writer = csv.writer(f)  
 for row in data:  
 csv\_writer.writerow(row)

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

Answer=

The delimiter argument specifies the character that separates fields (columns) in the CSV file. The default delimiter is a comma (,). You can change it to other character like ';' or '\t' depending on your CSV file.

The line terminator argument specifies the string that separates rows in the CSV file. The default line terminator is the newline character ('\n') on most systems. You can change it to '\r\n' if you want to use Windows line terminators.

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

Answer=

The json.loads() function takes a string of JSON data and returns a Python data structure, such as a dictionary or a list.

import json  
  
json\_data = '{"name": "Alice", "age": 25, "gender": "Female"}'  
python\_data = json.loads(json\_data)  
print(python\_data)

{'name': 'Alice', 'age': 25, 'gender': 'Female'}

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

Answer=

The json.dumps() function takes a Python data structure (such as a dictionary or a list) and returns a string of JSON data.

import json  
  
python\_data = {'name': 'Alice', 'age': 25, 'gender': 'Female'}  
json\_data = json.dumps(python\_data)  
print(json\_data)

{"name": "Alice", "age": 25, "gender": "Female"}