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

**Ans:-** Excel spreadsheets have several advantages over CSV files;-

1. Excel spreadsheets can store multiple sheets within a single file, which can be useful for structuring and organizing data. On the other hand, CSV files are limited to one sheet per file.
2. Excel allows the use of formulas and functions for calculations and data analysis, but this feature is not available in the CSV file, which contains only raw data.
3. Excel provides various formatting options for data such as font size and color, alignment and conditional formatting. This makes it easier to create visually appealing reports and presentations. In contrast, CSV files do not support any formatting options.
4. Excel offers advanced filtering and sorting options to quickly analyze and organize data. These options are not available in the CSV file, which require additional programming to implement such functionality.
5. Excel provides data validation to ensure that the data entered into the spreadsheet is accurate and consistent. This feature is not available in the CSV file.

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

**Ans:-** To create a reader object using a “csv” module in python, you need to pass a file object that represents a csv file to the “csv.reader()” function. Example :-

import csv

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

csv\_reader = csv.reader(csv\_file)

In the above code, ‘example.csv’ is the name of the CSV file, that we want to read, and ‘csv\_file’ is a file object that represents the CSV file in read mode. We then pass this file object to the ‘csv.reader()’ function to create a reader object.

To create a writer object using a ‘csv’ module in python, you need to pass a file object that represents the output file, where you want to write your csv data to, and optionally, a list of parameters to configure the behaviour of the writer. Example ;-

import csv

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

csv\_writer = csv.writer(csv\_file, delimiter=',', quotechar='"', quoting=csv.QUOTE\_MINIMAL)

In the above code, ‘output.csv’ is the name of the output csv file that we want to write, and ‘csv file’ is a file object that represents the output csv file in write mode. We then pass this file object to the ‘csv.writer()’ function along with some optional parameters to configure the behaviour of the writer. In this example, we set the delimiter to a comma(‘,’) , the quote character to a double quote (‘“‘) and the quoting mode to ‘csv.QUOTE\_MINIMAL’, which means that quote will only be added around fields that contain special characters like comma or quotes.

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

**Ans:-** When working with file objects for reader and writer objects, the modes in which they need to be opened depend on the type of operation you want to perform on the file.

1. For a reader object, you need to open the file in read mode using ‘r’ or ‘rt’ mode. The ‘r’ mode allows you to read the file in a text mode, while ‘rt’ mode allows you to read the file in binary mode.
2. For a writer object, you need to open the file in write mode using ‘w’ or ‘wt’ mode. The ‘w’ mode allow you to write the file in text mode, while ‘wt’ mode allows you to write to the file in binary mode.
3. If you want to append to an existing file, you can open the file in append mode using ‘a’ or ‘at’ mode. The ‘a’ mode allows you to append to the file in text mode, while ‘at’ mode allows you to append to the file in binary mode.

It’s important to note that when opening a file in write mode(‘w’ or ‘wt’), the file will be overwritten, if it already exists. If you want to add new data to an existing file without overwritten its contents, you should open it in append mode(‘a’ or ‘at’).

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

**Ans:-** We can use the ‘csv’ module in python, to write a list to a csv file. Here’s an example code;-

import csv

def write\_to\_csv\_file(file\_path, data\_list):

with open(file\_path, mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerows(data\_list)

In the above code, ‘file\_path’ is the path of the csv file where you want to write the data, and ‘data\_list’ is the list of data that you want to write to the csv file.

The ‘writerows()’ method of the ‘csv.writer’ object writes each element of the ‘data-list’ as a row in the csv file.

You can call the “write\_to\_csv\_file()” function with the appropriate arguments to write the list to the csv file. Example;-

data\_list = [['John', 'Doe', 25], ['Jane', 'Doe', 30], ['Bob', 'Smith', 40]]

file\_path = 'data.csv'

write\_to\_csv\_file(file\_path, data\_list)

This will write the ‘data-list’ to the ‘data.csv’ file. Each sublist in the “data\_list” will be written as a row in the csv file.

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

**Ans:-** The ‘delimeter’ and ‘lineterminator’ are keyword arguments that can be used with the ‘csv’ module in python.

1. **Delimiter** specifies the character used to separate the fields in the csv file. By default, the comma(‘ , ‘) character is used as the delimiter. However, you can specify a different delimiter by setting the **delimiter** keyword argument when you create a ‘csv.writer’ object. Example ;-

import csv

data\_list = [['John', 'Doe', 25], ['Jane', 'Doe', 30], ['Bob', 'Smith', 40]]

file\_path = 'example.csv'

with open(file\_path, mode='w', newline='') as file:

writer = csv.writer(file, delimiter='|')

writer.writerows(data\_list)

1. **Lineterminator** specifies the character used to terminate each line in the CSV file. By default, the line terminator is the newline character(\n). However, you can specify a different line terminator by setting the **lineterminator** keyword argument when you create a ‘csv.writer’ object. Example :-

import csv

data\_list = [['John', 'Doe', 25], ['Jane', 'Doe', 30], ['Bob', 'Smith', 40]]

file\_path = 'example.csv'

with open(file\_path, mode='w', newline='') as file:

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

writer.writerows(data\_list)

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

**Ans:-** We can use the ‘json’ module in python, to convert a string of JSON data into a pytho data structure. The **‘json’** module provides two methods for this purpose;-

1. ‘**json.loads()’** :- This method is used to parse a JSON string and convert it into a python object. The resulting object can be a dictionary, list, string, number, boolean or null value depending on the JSON string. Example :-

import json

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

# convert JSON string to a Python dictionary

data\_dict = json.loads(json\_string)

# print the Python dictionary

print(data\_dict)

1. **‘json.load()’ :-** This method is used to read a JSON file and convert its content to a python object. This method is similar to ‘json.loads()’ but instead of a string argument, it accepts a file object as an argument. Example:-

import json

file\_path = 'data.json'

# read JSON data from file and convert to a Python object

with open(file\_path) as file:

data\_obj = json.load(file)

# print the Python object

print(data\_obj)

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

**Ans:-** We can use the **json** module in python to convert a python data structure into a string of JSON data. The **json** module provides two methods for this purpose :-

1. **json.dumps() :-** This method is used to convert a python object into a JSON string. The resulting string will contain the JSON representation of the python object. Example :-

import json

data\_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

# convert Python dictionary to JSON string

json\_string = json.dumps(data\_dict)

# print the JSON string

print(json\_string)

1. **json.dump() :-** This method is used to write a python object to a file in JSON format. This method is similar to ‘json.dumps()’ but instead of returning a string, it writes the json data to a file object. Example :-

import json

data\_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

file\_path = 'data.json'

# write Python dictionary to JSON file

with open(file\_path, 'w') as file:

json.dump(data\_dict, file)

# print the contents of the JSON file

with open(file\_path) as file:

json\_data = json.load(file)

print(json\_data)