Python Assignment 13

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

Ans:-- Excel spreadsheets have several advantages over CSV (Comma-Separated Values) spreadsheets:

1. Rich Formatting : Excel allows for extensive formatting options, making it suitable for creating visually appealing reports and charts.

2. Multiple Worksheets : Excel supports multiple sheets in a single file, aiding organization.

3. Formulas and Functions : Excel provides built-in calculations and formula capabilities.

4. Charts and Graphs : Excel offers chart creation and data visualization tools.

5. Data Validation : It allows setting data entry rules for accuracy.

6. PivotTables and PivotCharts : For easy data summarization and analysis.

7. Cell Comments : You can add comments for explanations.

8. Data Sorting and Filtering : Built-in sorting and filtering features.

9. Password Protection : Files can be protected with passwords.

10. Formula Auditing : Tools for formula debugging.

11. Data Forms : Simplified data entry for tables.

12. Add-Ins and Macros : Custom automation and extensions.

13. Data Connection : Integration with external data sources.

CSV files are simpler but lack these features. The choice depends on data needs.

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

Ans:--- To create reader and writer objects using the `csv.reader()` and `csv.writer()` functions in Python's `csv` module, you pass a file object as an argument to these functions. Here's how you do it:

1. Creating a CSV Reader (csv.reader()) :

- To create a CSV reader, you pass a file object opened in read mode (`'r'`) to `csv.reader()`.

- You can open the file using the built-in `open()` function.

```python

import csv

# Open a CSV file for reading

with open('data.csv', 'r', newline='') as csv\_file:

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

# Now 'csv\_reader' is a reader object for 'data.csv'

```

2. Creating a CSV Writer (csv.writer()) :

- To create a CSV writer, you pass a file object opened in write mode (`'w'`) to `csv.writer()`.

- You can open the file using the built-in `open()` function.

```python

import csv

# Open a CSV file for writing

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

csv\_writer = csv.writer(csv\_file)

# Now 'csv\_writer' is a writer object for 'output.csv'

```

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

Ans:-- File objects for reader and writer objects in Python's `csv` module need to be opened with specific modes as follows:

1. CSV Reader (csv.reader()) :

- File objects for CSV readers need to be opened in read mode (`'r'`).

- You use this mode when you want to read data from an existing CSV file.

```python

import csv

# Open a CSV file for reading

with open('data.csv', 'r', newline='') as csv\_file:

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

```

2. CSV Writer (csv.writer()) :

- File objects for CSV writers need to be opened in write mode (`'w'`).

- You use this mode when you want to create a new CSV file or overwrite an existing one with new data.

```python

import csv

# Open a CSV file for writing

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

csv\_writer = csv.writer(csv\_file)

```

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

Ans:-- The method that takes a list argument and writes it to a CSV file using the `csv` module in Python is the `writerow()` method. This method is typically called on a CSV writer object, and it writes a single row (list) of data to the CSV file. Here's how it works:

```python

import csv

# Open a CSV file for writing

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

csv\_writer = csv.writer(csv\_file)

# Write a list of data (a row) to the CSV file

data\_row = ['John', 'Doe', 30]

csv\_writer.writerow(data\_row)

```

In this example, `csv\_writer.writerow(data\_row)` writes the list `data\_row` as a single row in the CSV file 'output.csv'. The elements of the list become the values in the corresponding columns of the CSV file.

You can call `writerow()` multiple times to write successive rows to the CSV file. Each time you call it with a new list of data, a new row is added to the CSV file.

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

The `delimiter` and `line\_terminator` keyword arguments are used when working with the `csv` module in Python to specify how the CSV data should be formatted and how the lines should be terminated in the CSV file:

1. delimiter :

- The `delimiter` argument specifies the character or string that separates the values (fields) within each row of the CSV file.

- By default, the comma (`,`) is used as the delimiter, but you can specify a different character or string to customize the delimiter.

- For example, if you want to use a semicolon as the delimiter, you can set `delimiter=';'` when creating the CSV writer.

```python

import csv

# Open a CSV file for writing with a semicolon (;) delimiter

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

csv\_writer = csv.writer(csv\_file, delimiter=';')

data\_row = ['John', 'Doe', 30]

csv\_writer.writerow(data\_row)

```

2. line\_terminator :

- The `line\_terminator` argument specifies the character or string to be used as the line terminator at the end of each row in the CSV file.

- By default, the line terminator is `'\n'` (newline character), which is common on most systems. However, you can customize it to use a different character or string.

- For example, if you want to use a custom line terminator such as `'\r\n'` for compatibility with Windows-style line endings, you can set `line\_terminator='\r\n'` when creating the CSV writer.

```python

import csv

# Open a CSV file for writing with a custom line terminator (\r\n)

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

csv\_writer = csv.writer(csv\_file, line\_terminator='\r\n')

data\_row = ['John', 'Doe', 30]

csv\_writer.writerow(data\_row)

```

Customizing the `delimiter` and `line\_terminator` allows you to adapt the CSV formatting to your specific requirements or to ensure compatibility with different systems and applications.

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

Ans:-- The function that takes a string of JSON data and returns a Python data structure is `json.loads()`. This function is part of the `json` module in Python and is used to deserialize (parse) a JSON-formatted string into a corresponding Python object.

Here's how you can use `json.loads()`:

```python

import json

# JSON-formatted string

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

# Deserialize JSON data into a Python dictionary

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

# 'python\_data' now contains a Python dictionary

print(python\_data)

```

In this example:

1. We import the `json` module.

2. We have a JSON-formatted string `json\_data` containing key-value pairs.

3. We use `json.loads(json\_data)` to parse the JSON string and convert it into a Python dictionary (`python\_data`).

After calling `json.loads()`, you can work with the Python data structure (`python\_data`) as you would with any Python dictionary or data object.

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

Ans:-- The function that takes a Python data structure and returns a string of JSON data is `json.dumps()`. This function is part of the `json` module in Python and is used to serialize (convert) a Python object into a JSON-formatted string.

Here's how you can use `json.dumps()`:

```python

import json

# Python dictionary

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

# Serialize Python data into a JSON-formatted string

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

# 'json\_data' now contains a JSON-formatted string

print(json\_data)

```

In this example:

1. We import the `json` module.

2. We have a Python dictionary `python\_data` containing key-value pairs.

3. We use `json.dumps(python\_data)` to serialize the Python dictionary into a JSON-formatted string (`json\_data`).

After calling `json.dumps()`, `json\_data` contains a string representation of the Python data structure in JSON format, which can be easily stored in a file, transmitted over a network, or used in any context that requires JSON data.