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

**ANS:-**

**Excel:-**

* **Excel files can only be read by applications and not on any other systems, application or data platform**
* **Excel consumes more memory than CSV**
* **Excel is generally larger, slower and complicated when compared to CSV**
* **Files saved in Excel format cannot be edited by text editors and also Excel file can be password protected**
* **Excel can perform VBA Coding, functions, formulas and create summaries easily when compared to CSV**
* **Reading large data set is much easier for the end user to read, comprehended and understand when compared to CSV**
* **Excel can be opened with Microsoft Office only and does not support any other platform when compared to CSV**
* **Excel also provides the user option for external linking of data from other sources and also the user can do custom add-ins**

**CSV:-**

* **CSV file does not need any specific platform to open it can also be opened in a simple text editor and the user can simply read and understand it**
* **CSV consumes less memory than Excel**
* **CSV is generally faster and less complicated when compared to Excel**
* **Files saved in CSV format can be edited by text editors and also CSV file cannot be password protected**
* **CSV cannot perform VBA coding, functions, formulas and create summaries easily when compared to Excel**
* **Reading large data set is much complicated for the end user to read, comprehend and understand when compared to Excel**
* **CSV format is compatible and can be used on several platform like notepad, access etc when compared to Excel**
* **No such feature is available in CSV format when it is compared with Excel**

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

**ANS:-**

## How to Read CSV Files in Python with Procedural Steps?

There are many different ways to read data in a CSV file, which we will now see one by one.

### Steps to Read CSV Files in Python Using csv.reader

You can read CSV files using the **csv.reader** object from Python’s **csv** module. Steps to read a CSV file using csv reader:

1. **Import the CSV library**

import csv

1. **Open the CSV file**

The .**open()** method in python is used to open files and return a file object.  
  
file = open('Salary\_Data.csv')  
type(file)  
  
The type of file is “**\_io.TextIOWrapper**” which is a file object that is returned by the **open()**method.

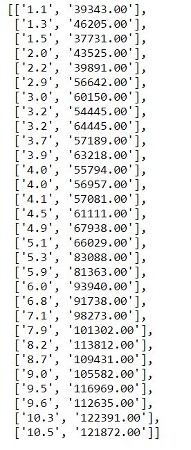
1. **Use the csv.reader object to read the CSV file**

csvreader = csv.reader(file)

1. **Extract the field names**

Create an empty list called a header. Use the next() method to obtain the header.  
The .next() method returns the current row and moves to the next row.  
The first time you run next(), it returns the header, and the next time you run, it returns the first record, and so on.  
  
header = []  
header = next(csvreader)  
header  
  
Field names in CSV header [python read csv]

1. **Extract the rows/records**

Create an empty list called rows and iterate through the csvreader object and append each row to the rows list.  
  
rows = []  
for row in csvreader:  
rows.append(row)  
rows  
  


1. **Close the file**

**.close()** method is used to close the opened file. Once it is closed, we cannot perform any operations on it.  
  
file.close()

### Complete Code for Read CSV Python

#### ****Python Code****

Naturally, we might forget to close an open file. To avoid that, we can use the **with()**statement to automatically release the resources. In simple terms, there is no need to call the .close() method if we are using with() statement.

**Implementing the above code using with() statement:**

**Basic Syntax:**with open(filename, mode) as alias\_filename:

**Modes:**

* ‘r’ – to read an existing file,
* ‘w’ – to create a new file if the given file doesn’t exist and write to it,
* ‘a’ – to append to existing file content,
* ‘+’ –  to create a new file for reading and writing

**import** csv

rows = []

**with** open("Salary\_Data.csv", 'r') **as** file:

csvreader = csv.reader(file)

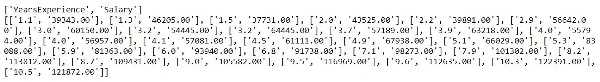
header = next(csvreader)

**for** row **in** csvreader:

rows.append(row)

print(header)

print(rows)



**Also Read:**[**The Evolution and Future of Data Science Innovation**](https://www.analyticsvidhya.com/blog/2023/04/innovations-in-data-science/)

### How to read CSV Files in Python Using .readlines()?

Now the question is – “Is it possible to fetch the header and rows using only open() and with() statements and without the csv library?” Let’s see…

.**readlines()** method is the answer. It returns all the lines in a file as a list. Each item on the list is a row of our CSV file.

The first row of the file.readlines() is the header, and the rest are the records.

**with** open('Salary\_Data.csv') **as** file:

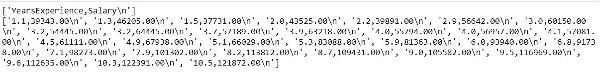
content = file.readlines()

header = content[:1]

rows = content[1:]

print(header)

print(rows)



\*\*The ‘n’ from the output can be removed using .strip() method.

What if we have a huge dataset with hundreds of features and thousands of records? Would it be possible to handle lists??

Here comes pandas library into the picture.

### How to Read CSV Files in python Using Pandas?

Let’s have a look at how pandas are used to read data in a CSV file.

#### ****1. Import pandas library****

**import** pandas **as** pd

#### ****2. Load CSV files to pandas using read\_csv()****

**Basic Syntax:**[pandas.read](http://pandas.read/" \t "_blank)\_csv(filename, delimiter=’,’)

**data**= pd.read\_csv("Salary\_Data.csv")

**data**



#### ****3. Extract the field names****

**.columns**is used to obtain the header/field names.

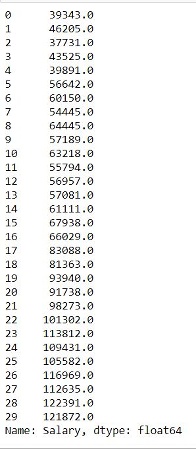
**data**.columns

.columns in csv python pandas [python read csv]

#### ****4. Extract the rows****

All the data of a data frame can be accessed using the field names.

**data**.Salary



### Read CSV file in python using csv.DictReader

A dictionary in how to Read CSV file in Python is like a hash table, containing keys and values. To create a dictionary, you use the dict() method with specified keys and values. If you’re working with CSV files in Python, the csv module’s .DictReader comes in handy for reading them. Here’s a simple guide on how to use Python to read CSV file

#### ****1. Import the csv module****

**import** csv

#### ****2. Open the CSV file using the****.****open() function with the mode set to ‘r’ for reading.****

**with** open('Salary\_Data.csv', 'r') **as** csvfile:

**3. Create a DictReader object using the csv.DictReader() method.**

reader = csv.DictReader(csvfile)

**4. Use the csv.DictReader object to read the CSV file.**

Iterate through the rows of the CSV file using a ‘for’ loop and the DictReader object to see the field names as keys along with their respective values.

**for** row **in** reader:

print(row)

## List of Methods to Write a CSV file in python

* **Write CSV file using csv.writer**
* **Write CSV file using writelines() function**
* **Write CSV file using Pandas**
* **Write CSV file using csv.DictWriter**

## How to Write to a Python CSV?

We can write to a CSV file in multiple ways.

### Write CSV file Using csv.writer

The csv.writer() function returns a writer object that converts the input data into a delimited string.  
For example, let’s assume we are recording the data of 3 students (Name, M1 Score, M2 Score)

header = ['Name', 'M1 Score', 'M2 Score']

data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]

Now let’s see how this data can be written to a CSV file using csv.writer:

**1. Import csv library.**

**import** csv

**2. Define a filename and Open the file using open().**  
**3. Create a csvwriter object using csv.writer().**  
**4. Write the header.**  
**5. Write the rest of the data.**

Code for steps 2-5

filename = 'Students\_Data.csv'

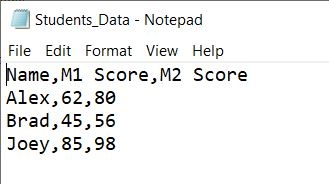
**with** open(filename, 'w', newline="") **as** file:

csvwriter = csv.writer(file) # 2. create a csvwriter object

csvwriter.writerow(header) # 4. write the header

csvwriter.writerows(data) # 5. write the rest of the data

Below is how our CSV file looks.



### Write CSV File Using .writelines()

.writelines() iterates through each list, converts the list elements to a string, and then writes it to the csv file.

header = ['Name', 'M1 Score', 'M2 Score']

data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]

filename = 'Student\_scores.csv'

with open(filename, 'w') as file:

**for** header **in** header:

file.write(str(header)+', ')

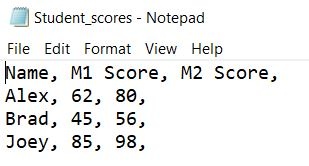
file.write('n')

**for** row **in** data:

**for** x **in** row:

file.write(str(x)+', ')

file.write('n')



### Write CSV Using Pandas

Follow these steps to write to a CSV file using pandas:

#### ****1. Import pandas library****

**import** pandas **as** pd

#### ****2. Create a pandas dataframe using pd.DataFrame****

**Syntax:**pd.DataFrame(data, columns)

The data parameter takes the records/observations, and the columns parameter takes the columns/field names.

header = ['Name', 'M1 Score', 'M2 Score']

data = [['Alex', 62, 80], ['Brad', 45, 56], ['Joey', 85, 98]]

data = pd.DataFrame(data, columns=header)

#### ****3. Write to a CSV file using to\_csv()****

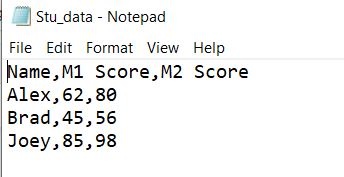
**Syntax:**DataFrame.to\_csv(filename, sep=’,’, index=False)

\*\*separator is ‘,’ by default.

index=False to remove the index numbers.

**data**.to\_csv('Stu\_data.csv', index=False)

Below is how our CSV looks like



### Write CSV File Using csv.DictWriter

You can write data into a CSV file using the csv module .DictReader following the below steps.

#### ****1. Import the csv module****

**import** csv

#### ****2. Using the****.****open() function, create a new file object with the mode as ‘w’ for writing****

Create a new file object using the open() function, specifying the file name with the mode set as ‘w’ for writing.

**with** open('Students\_Data.csv', 'w', newline='') **as** csvfile:

#### ****3. Type in the data you want to write to the CSV file as a list of dictionaries****

data = [{'Name': 'Alex', 'M1 Score': 62, 'M2 Score': 80},

{'Name': 'Brad', 'M1 Score': 45, 'M2 Score': 56},

{'Name': 'Joey', 'M1 Score': 85, 'M2 Score': 98}]

#### ****4. Create a csv.DictWriter object specifying the file object, the fieldname parameters, and the delimiter****

Note that the delimiter by default is ‘,’

fieldnames = ['Name', 'M1 Score', 'M2 Score'] writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

#### ****5. Write the header row using the writeheader() method.****

writer.writeheader()

#### ****6. Use the writerows() method to write the data to the CSV file****

writer.writerows(**data**)

This will create a new file named ‘Students\_Data.csv’ with Name, M1 Score, and M2 Score as the header/column names and the data values under the data variable.

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

**ANS:-**

'r'- It opens a file for reading only.

'w'- It opens a file for writing. If the file exists, it overwrites it, otherwise, it creates a new file.

'a'- It opens a file for appending only. If the file doesn't exist, it creates the file.

'x'- It creates a new file. If the file exists, it fails.

'+' - It opens a file for updating.

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

**ANS:-**

**The csv.writer() method in Python takes a list argument and writes it to a CSV (Comma-Separated Values) file. It is part of the csv module in Python's standard library.**

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

**ANS:-**

## The delimiter and lineterminator Keyword Arguments

Say you want to separate cells with a tab character instead of a comma and you want the rows to be double-spaced. You could enter something like the following into the interactive shell:

>>> **import csv**

>>> **csvFile = open('example.tsv', 'w', newline='')**

❶ >>> **csvWriter = csv.writer(csvFile, delimiter='\t', lineterminator='\n\n')**

>>> **csvWriter.writerow(['apples', 'oranges', 'grapes'])**

24

>>> **csvWriter.writerow(['eggs', 'bacon', 'ham'])**

17

>>> **csvWriter.writerow(['spam', 'spam', 'spam', 'spam', 'spam', 'spam'])**

32

>>> **csvFile.close()**

This changes the delimiter and line terminator characters in your file. The delimiter is the character that appears between cells on a row. By default, the delimiter for a CSV file is a comma. The line terminator is the character that comes at the end of a row. By default, the line terminator is a newline. You can change characters to different values by using the delimiter and lineterminator keyword arguments with csv.writer().

Passing delimeter='\t' and lineterminator='\n\n' ❶ changes the character between cells to a tab and the character between rows to two newlines. We then call writerow() three times to give us three rows.

This produces a file named example.tsv with the following contents:

apples oranges grapes

eggs bacon ham

spam spam spam spam spam spam

Now that our cells are separated by tabs, we’re using the file extension .tsv, for tab-separated values.

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

**ANS:-**

The methods are called *serialization* *and* *deserialization of json*. What you are asking about is called **Deserialization**. Let me tell you about both.

Simply put -

**Serialization** means conversion from Python to Json format. Example -

1. **>>> import json**
2. **>>>**
3. **>>> # create a python dictionary**
4. **>>> myDict = {**
5. **... "name": "Ashutosh Tripathy",**
6. **... "interests": ["Python","Web Dev","Data Science","Linux"]**
7. **... }**
8. **>>>**
9. **>>> # convert dictionary into a string in json format**
10. **>>> jsonStr = json.dumps(myDict)**
11. **>>>**
12. **>>> # check the output**
13. **>>> print(jsonStr)**
14. **{"name": "Ashutosh Tripathy", "interests": ["Python", "Web Dev", "Data Science", "Linux"]}**
15. **>>>**
16. **>>> # you can also save it with added indentation**
17. **>>> jsonStr = json.dumps(myDict, indent=4)**
18. **>>>**
19. **>>> # check the indented output**
20. **>>> print(jsonStr)**
21. **{**
22. **"name": "Ashutosh Tripathy",**
23. **"interests": [**
24. **"Python",**
25. **"Web Dev",**
26. **"Data Science",**
27. **"Linux"**
28. **]**
29. **}**

**Deserialization** means conversion from Json to Python format (as in your question). Example -

1. >>> import json
2. >>>
3. >>> # store the json data in a variable named jsonData
4. >>> jsonData = '{ "firstName": "Ashutosh", "lastName": "Tripathy", "hobbies": ["coding", "singing", "gaming"], "gender": "male" }'
5. >>>
6. >>> # convert jsonData into a python dictionary
7. >>> myData = json.loads(jsonData)
8. >>>
9. >>> # finally use myData as a python dictionary
10. >>> print(myData["firstName"])
11. Ashutosh
12. >>> print(myData["hobbies"])
13. ['coding', 'singing', 'gaming']

This is pretty much covering some of the basic stuffs of the **json**module in Python.

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

**ANS:-**

json.dumps() function provided by the JSON module to create JSON from a [Python dictionary](https://blog.ashutoshkrris.in/everything-you-need-to-know-about-python-dictionaries). This function takes a Python object, typically a dictionary, and converts it into a JSON string representation.

import json

# Python dictionary

data = {

"name": "Ashutosh Krishna",

"age": 23,

"email": "ashutosh@example.com"

}

# Convert dictionary to JSON string

json\_str = json.dumps(data)

# Print the JSON string

print(json\_str)

In this example, you have a Python dictionary data representing some data. By calling json.dumps(data), you convert the dictionary into a JSON string.

The output will be:

{"name": "Ashutosh Krishna", "age": 23, "email": "ashutosh@example.com"}