Creating csv file and writing data using different delimiter.

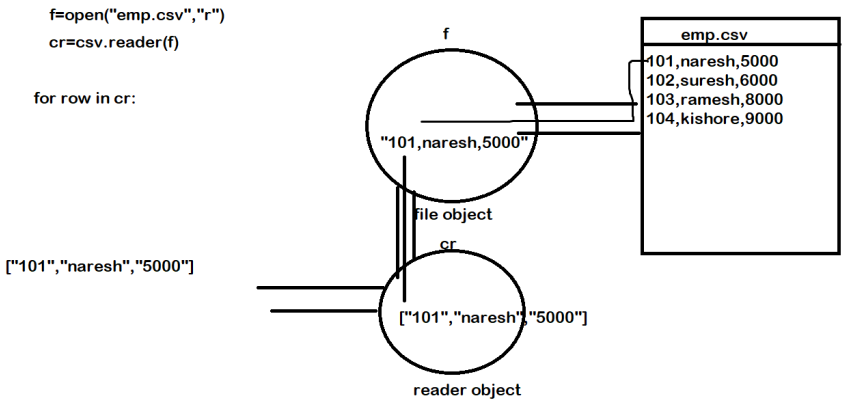
```
import csv
import sys
try:
    f=open("student.csv","w",newline='')
    cw=csv.writer(f,delimiter=' ')
    cw.writerow([101,"naresh","python"])
    cw.writerow([102,"suresh","java"])
    cw.writerow([103,"ramesh","c++"])
    print("data is saved")
except:
    t=sys.exc_info()
    print(t)
finally:
    f.close()
```

Output

data is saved

csv.reader(csvfile)

Return a reader object that will process lines from the given csvfile. A csvfile must be an iterable of strings, each in the reader's defined csv format. A csvfile is most commonly a file-like object or list. If csvfile is a file object, it should be opened with newline=''



Example:

```
import csv

with open("emp.csv") as f: # Creating context manager
    cr=csv.reader(f)
    for row in cr:
        print(row)
```

Output

```
['empno', 'ename', 'salary']
['1', 'naresh', '5000.0']
['2', 'suresh', '6000.0']
['3', 'kishore', '9000.0']
```

csv.DictWriter(f, fieldname)

Create an object which operates like a regular writer but maps dictionaries onto output rows. The fieldnames parameter is a sequence of keys that identify the order in which values in the dictionary passed to the writerow() method are written to file f.

Example:

```
import csv

with open("sales.csv","w",newline='') as f:
    dw=csv.DictWriter(f,fieldnames=["year","sales"])
    dw.writeheader()
    while True:
        y=int(input("Enter Year :"))
        s=float(input("Enter Sales :"))
        d={'year':y,'sales':s}
        dw.writerow(d)
        ans=input("Add another sales ?")
        if ans=="no":
            break

print("Data is Written")
```

Output

```
Enter Year :2001
Enter Sales :54000
Add another sales ?yes
Enter Year :2002
Enter Sales :60000
Add another sales ?yes
Enter Year :2003
Enter Sales :70000
Add another sales ?yes
Enter Year :2004
Enter Sales :90000
Add another sales ?yes
Enter Year :2005
Enter Sales :76000
Add another sales ?yes
```

Enter Year :2006
Enter Sales :80000
Add another sales ?no
Data is Written

csv.DictReader(f, fieldnames=None)
Create an object that operates like a regular reader but maps the information in each row to a dict whose keys are given by the optional fieldnames parameter.

Example:

```
import csv

with open("sales.csv","r") as f:
    cr=csv.DictReader(f)
    total=0
    for row in cr:
        print(row['year'],row['sales'])
        total=total+float(row['sales'])

    print(f'Total sales {total:.2f}')
```

Output

```
2001 54000.0
2002 60000.0
2003 70000.0
2004 90000.0
2005 76000.0
2006 80000.0
Total sales 430000.00
```

Example:

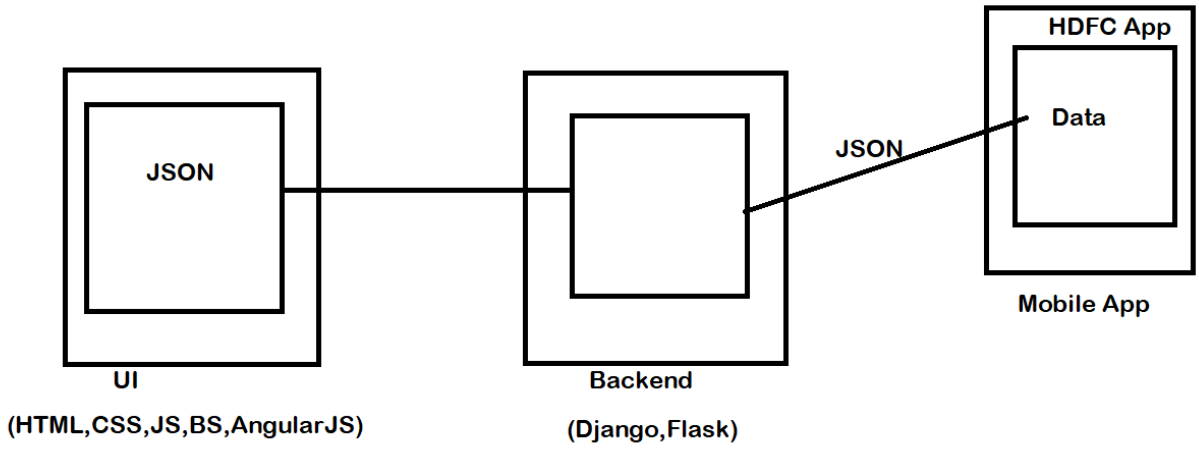
```
# Replacing data/updating data
f=open("file1.txt","r+")
ch=f.read(1)
print(ch)
f.seek(0,0)
f.write('PYTHON')
f.close()
```

JSON file

JSON stands for Java Script Object Notation.
JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

To work with json file , python provides a predefined module called "json".

Json module provides encoders and decoders.



In json data is written as key and value pair.