**Reading data from text file**

1. read()
2. readlines()

reading data from file is done in sequentially and randomly.

**read(*size=-1*, /)**

Read and return at most *size* characters from the stream as a single str. If *size* is negative or None, reads until EOF.

**Example:**

```
import sys
try:
   f=open("file1.txt","r")
   s=f.read()
   print(s)
except:
   t=sys.exc_info()
   print(t)
finally:
   f.close()
```

**Output**

```
Python3.12
```

**Example:**

```
# Count of vowels exists within file
import sys
try:
   f=open("file1.txt","r")
   c=0
   while True:
       ch=f.read(1)
       if ch=='':
           break
       elif ch in "aeiouAEIOU":
           c=c+1
   print(f'Count of vowels {c}')

except:
   t=sys.exc_info()
   print(t)
finally:
   f.close()
```

**Output**

```
Count of vowels 1
```

**seek(offset, whence=SEEK_SET, /)**

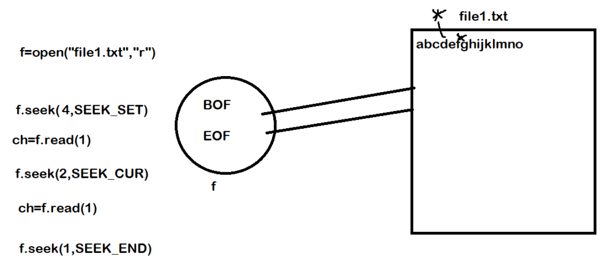
Change the stream position to the given offset. Behaviour depends on the whence parameter. The default value for whence is SEEK_SET.

SEEK_SET or 0: seek from the start of the stream (the default); offset must either be a number returned by TextIOBase.tell(), or zero. Any other offset value produces undefined behaviour.

SEEK_CUR or 1: “seek” to the current position; offset must be zero, which is a no-operation (all other values are unsupported).

SEEK_END or 2: seek to the end of the stream; offset must be zero (all other values are unsupported).

Return the new absolute position



**tell()**
Return the current stream position as an opaque number.

**Example:**
```
try:
    f=open("file1.txt","r")
    p=f.tell()
    print(p)
    ch=f.read(1)
    print(ch)
    p=f.tell()
    print(p)
    ch=f.read(2)
    print(ch)
    p=f.tell()
    print(p)
    f.seek(2,0)
    ch=f.read(1)
    print(ch)
    f.seek(0,1)
    ch=f.read(1)
    print(ch)
    f.seek(0,1)
    ch=f.read(1)
    print(ch)

except:
    print("error")
finally:
```

**Output**

```
0
P
1
yt
3
t
h
o
```

**readline**(*size=-1*, */*)
Read until newline or EOF and return a single str. If the stream is already at EOF, an empty string is returned.

**Example:**

```
# Reading lines

try:
    f=open("file2.txt","r")
    line1=f.readline()
    print(line1)
    line2=f.readline()
    print(line2)
    line3=f.readline()
    print(line3)
    line4=f.readline()
    print(line4)
    line5=f.readline()
    print(line5)

except:
    print("error")
```

**Output**

```
java
python
C++
C
```

**Example:**

```
# read data from student.txt and calculate total,avg and result

try:
    f=open("student.txt","r")
    while True:
        stud=f.readline()
        if stud=='':
            break
        else:
            list1=stud.split()
            rollno,name,sub1,sub2=list1
            total=int(sub1)+int(sub2)
            avg=total/2
            result="pass" if int(sub1)>=40 and int(sub2)>=40 else "fail"

print(f'{rollno}\t{name}\t{sub1}\t{sub2}\t{total}\t{avg}\t{result}')
```

```
except:
   print("error")
finally:
   f.close()
```

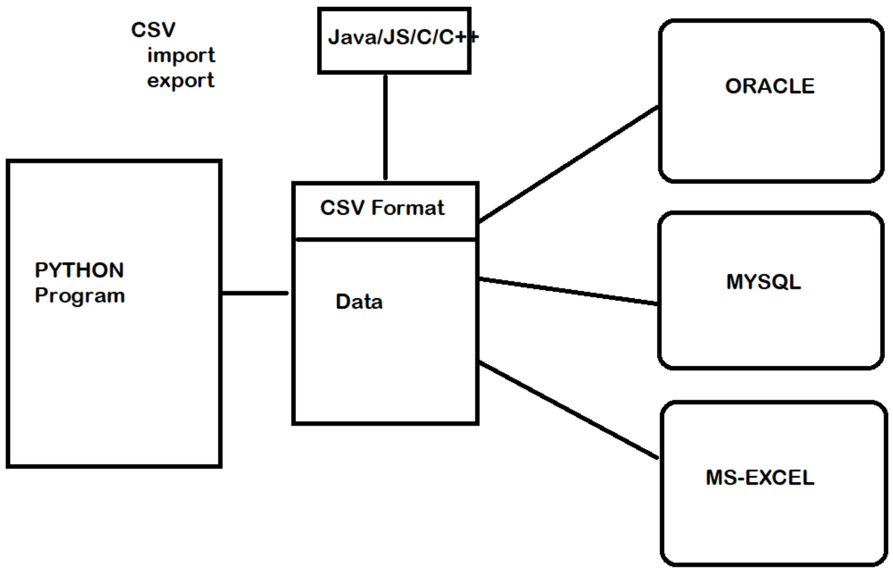
**Output**

```
1     naresh     60    70    130   65.0  pass
2     suresh     40    90    130   65.0  pass
3     ramesh     30    60    90    45.0  fail
```

**CSV Files (csv module)**

CSV stands for Comma Separated Values. It is a text file.
"csv" module, is a default module which comes with python software. "csv" module provides class and objects to work with csv files.
The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases

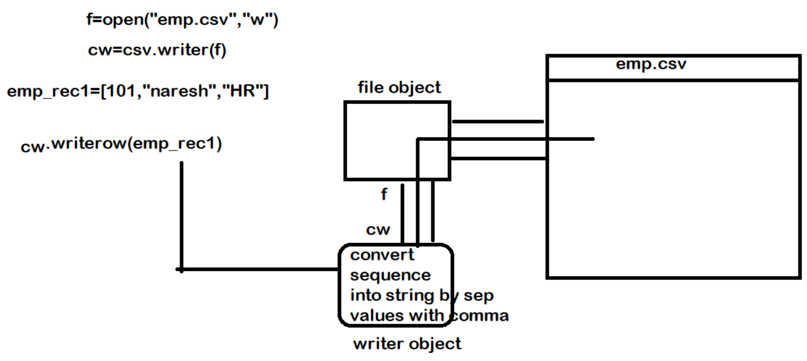
The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, "write this data in the format preferred by Excel," or "read data from this file which was generated by Excel," without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.



The csv module's reader and writer objects read and write sequences. Programmers can also read and write data in dictionary form using the DictReader and DictWriter classes.

**csv.writer(csvfile)**

Return a writer object responsible for converting the user's data into delimited strings on the given file-like object. csvfile can be any object with a write() method. If csvfile is a file object, it should be opened with newline=''



```
import csv

try:
   f=open("emp.csv","w",newline='')
   cw=csv.writer(f)
   cw.writerow(['empno','ename','salary'])
   while True:
      empno=int(input("EmployeeNo: "))
      ename=input("EmployeeName: ")
      salary=float(input("Salary: "))
      cw.writerow([empno,ename,salary])
      ans=input("Add another student?")
      if ans=="no":
         break
except:
   print("error")
finally:
   f.close()
```

**Output**

```
EmployeeNo: 1
EmployeeName: naresh
Salary: 5000
Add another student?yes
EmployeeNo: 2
EmployeeName: suresh
Salary: 6000
Add another student?yes
EmployeeNo: 3
EmployeeName: kishore
Salary: 9000
```

Add another student?no