Python
File I/O

Python File methods

EOF – Detect End of File

programmer takes care of

Open, close and reading

Other methods:

with/for

while

for

Reminder

An empty string contains a NULL character

Define a null character:

str909 = “”

or

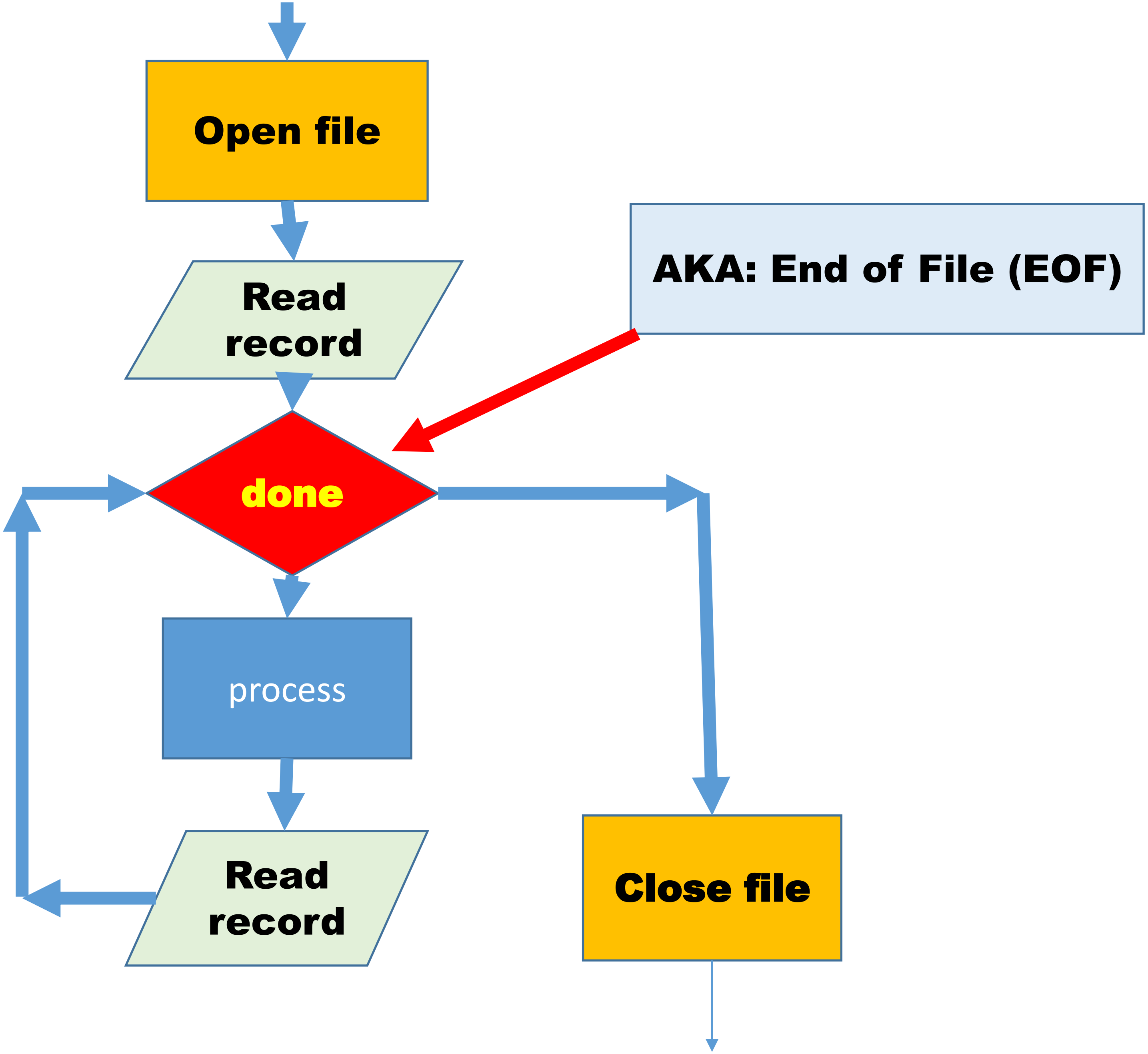
str808 = ‘’

no spaces between quotes

EOF Method

This method is used by Python and most programming languages

C++	is.eof()
php	!feof($file)
Java	line != null
VB	Not EOF(1)
Perl	eof(fh)


Open file
Read record
AKA: End of File (EOF)
done
process
Read record
Close file

File Handling Commands

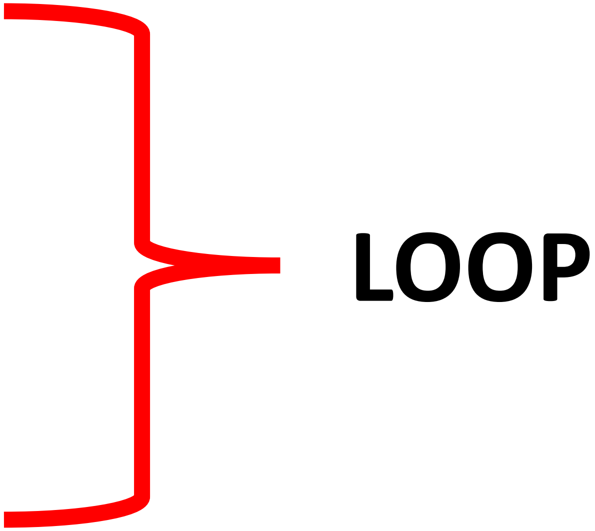
open - do once

EOF - once per read

read - one or more

write - one of more

append – one of more



LOOP

close - do once

open

Path and File Name

Normal:

c:\temp\filelist.txt

Python Version:

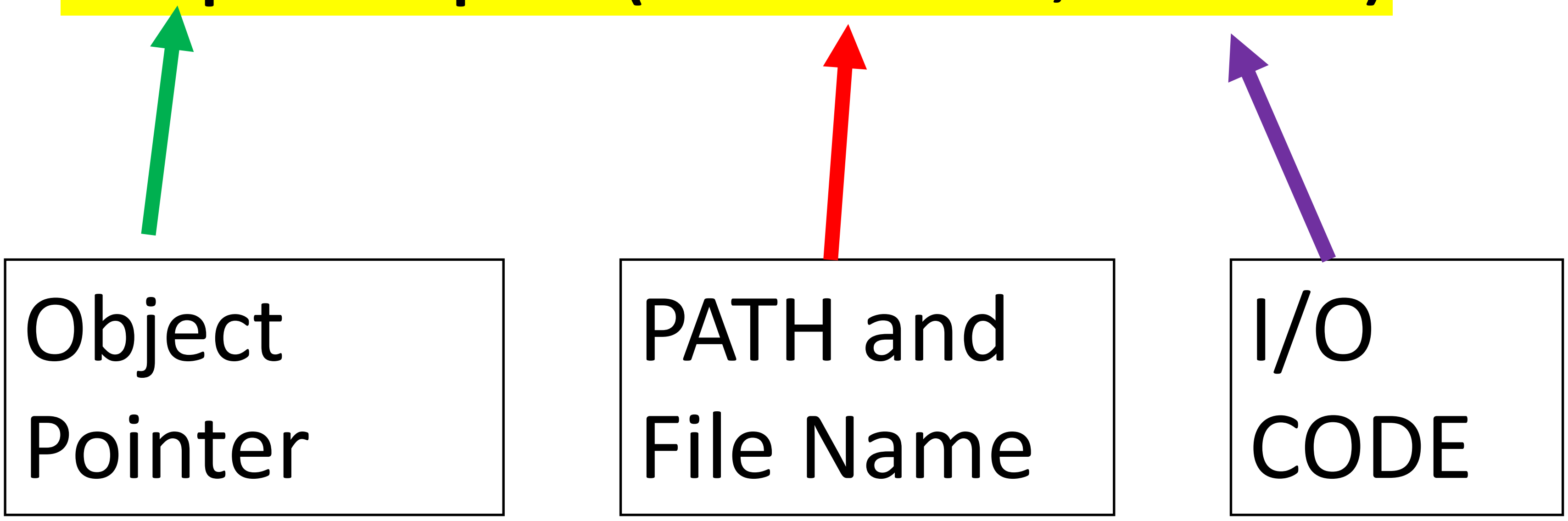
c:/temp/filelist.txt

NOTE:
String (path & file) cannot have the backslash.

Open

FORM:

```
filept = open('workfile', ‘code')
```



Object Pointer

PATH and File Name

I/O CODE

CODE List

r - read

w - write

a - append

Open Example

```
inputfile=‘c:/temp/testfile1.txt’

filein = open(inputfile, ‘r')
```

Codes

r - Opens a file for reading. (default)

w - Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.

x - Opens a file for exclusive creation. If the file already exists, the operation fails.

a - Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.

t - Opens in text mode. (default)

b - Opens in binary mode.

+ - Opens a file for updating (reading and writing

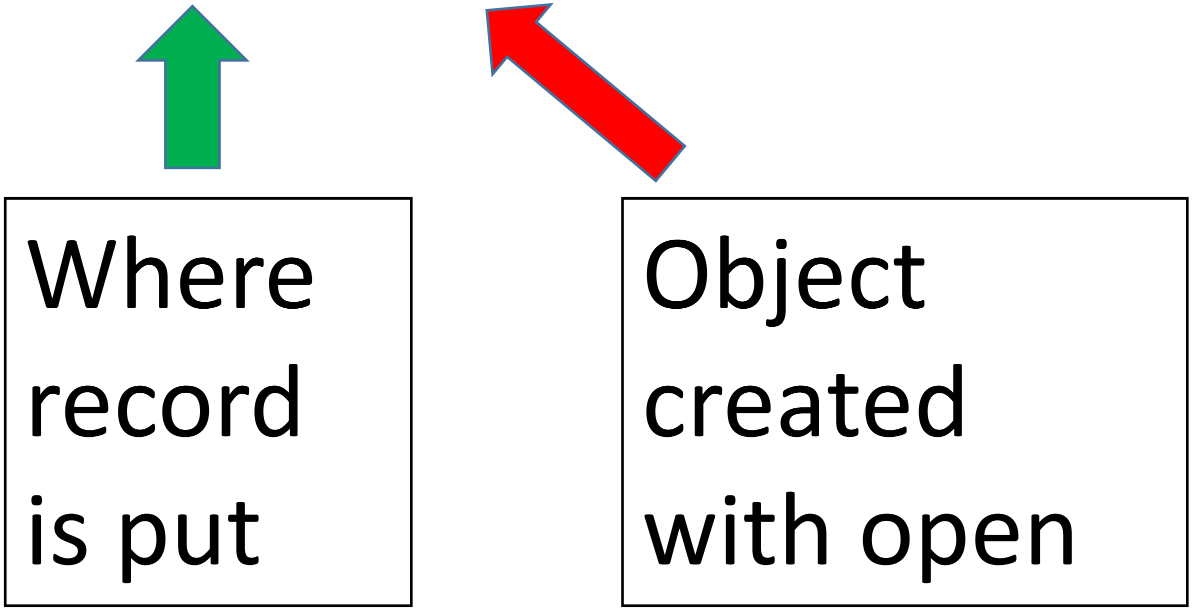
read

Read one record

Read - read a record

Form:

```
rline = filein.readline()
```



Where record is put

Object created with open

Read execution

```
rline = filein.readline()
```

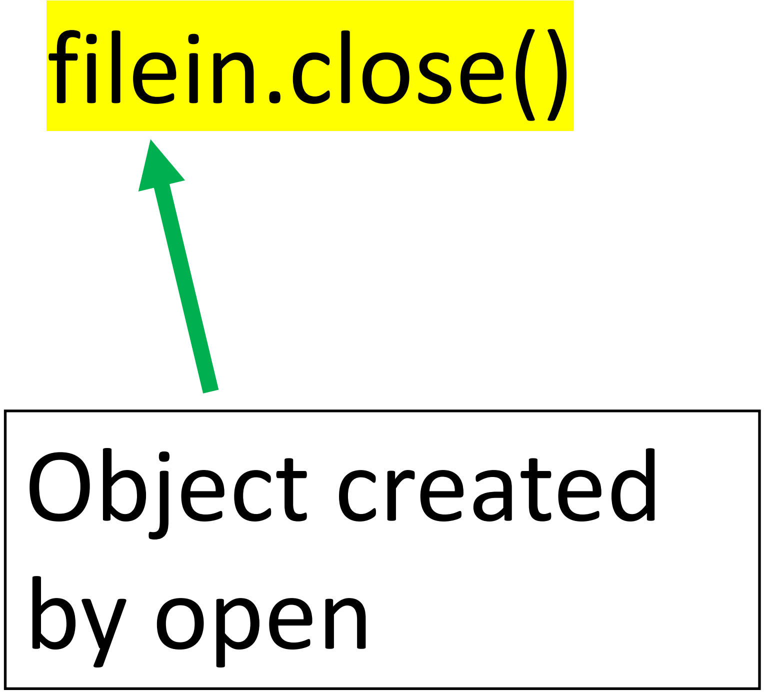
rline will contain either a record or a null character

close

Close the file

Form:

```
filein.close()
```



Object created by open

- It puts your program in the garbage collectors hands - though the file in theory will be auto closed, it may not be closed. Python 3 and Cpython generally do a pretty good job at garbage collecting, but not always, and other variants generally not good at it.

- It can slow down your program. Too many things open, and thus more used space in the RAM, will impact performance.

- For the most part, many changes to files in python do not go into effect until after the file is closed, so if your script edits, leaves open, and reads a file, it won't see the edits.

- You could, theoretically, run in to limits of how many files you can have open.

- Windows treats open files as locked, so legit things like AV scanners or other python scripts can't read the file.

- Memory get released back to the O/S

- It is sloppy programming

examples

Create File

Open Notepad

Type in the following:

```
  Record 1
  Record 2
  Record 3
```

Save as: testfile1.txt   in   C:\temp

```
def main():
    filenamein ='c:/temp/testfile1.txt'
    infile = open(filenamein,'r')
    line1 = infile.readline()
    line2 = infile.readline()
    line3 = infile.readline()
    infile.close()
    print('Contents of file')
    print(line1)
    print(line2)
    print(line3)
main()
```

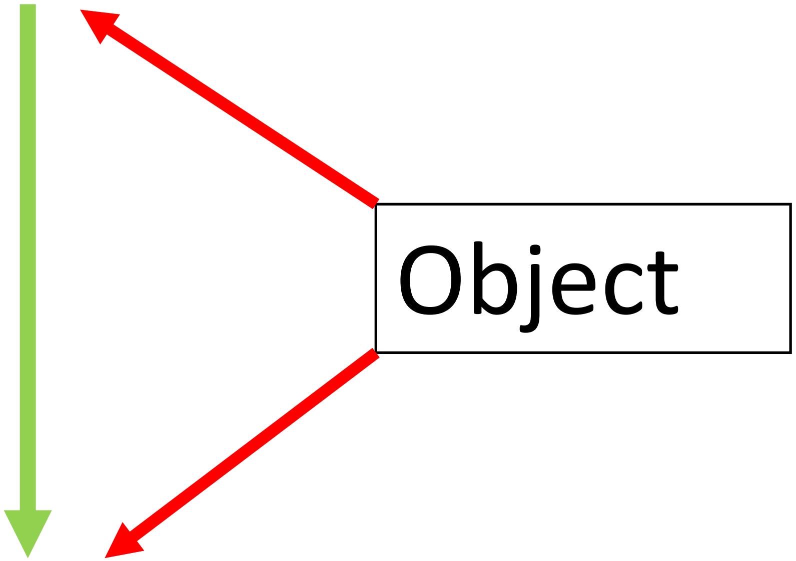
Try
This

write

Write

Open with code w

```
outfile = open(filenameout,'w')
```



Form:

```
outfile.write(' the record 1\n')
```

Try this

```
def main():
    filenameout ='c:/temp/ph.txt'
    outfile = open(filenameout,'w')
    outfile.write(' john locke')
    outfile.write('david hume')
    outfile.write('ed burke')
    outfile.close()
main()
```

Examine file with notepad

file02

```
def main():
    filenameout ='c:/temp/ph2.txt'
    outfile = open(filenameout,'w')
    outfile.write(' john locke\n')
    outfile.write('david hume\n')
    outfile.write('ed burke\n')
    outfile.close()
main()
```

Fix
and
Try

Add: \n

Examine with notepad

The Null character

Hex 00

line = ''

or

line = ""

Sets a NULL string

EOF – End of file

Read a record

Get a record or get the Null character

```
    line1 = infile.readline()
```

Condition Test for Null

```
  (line == "")  or (line != “”)
```

Use in while loop

While – detect end of file

```
open file
line =“”
line = infile.readline()
while (line != “”):
      statements
      line = infile.readline()
statements
close file
```

DONE ONCE

Detect End of File

ONE ONCE

NOTE

```
def main():
    filenamein ='c:/temp/testfile1.txt'
    rcdcnt = 0
    infile = open(filenamein,'r')
    line=infile.readline()
    while (line !=""):
        print (line)
        line=infile.readline()
        rcdcnt = rcdcnt+1
    infile.close()
    print('\n’, 'Number of records read: ‘ , rcdcnt)
    print (' ---done')
main()
```

DONE ONCE

Processing loop

Done once

file04

Try This

```
def main():
    filenamein ='c:/temp/testfile1.txt'
    rcdcnt = 0
    infile = open(filenamein,'r')
    line=infile.readline()
    while (line !=""):
        rcdcnt = rcdcnt+1
        print (line)
        line=infile.readline()
    infile.close()
    print('\n'+ 'Number of records read: ' + str(rcdcnt))
    print (' ---done')
main()
```

file04

File error

Detect with try & except

```
infile = open(filenamein,'r’)
```

```
FileNotFoundError:
[Errno 2] No such file or directory:
'c:/temp/testfile1.txt'
```

Example code with try/except

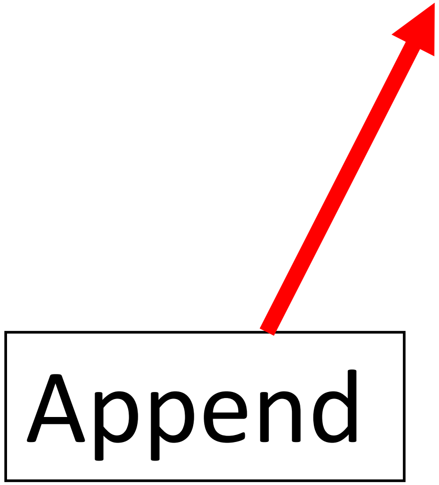
```
good = 0
try:
    filein=open(filenamein,’r’)
except FileNotFoundError:
     good = 1
     break
except:
     good=2
```

append

Append

Set open to append

```
fileapp = open(filenameap,’a’)
```



Append

Try This

```
def main():
    eofm = “\n”
    filename ='c:/temp/ph.txt'
    appfile = open(filename, 'a')
    print (' enter three names: ')
    name1 = input(‘Name #1: ')
    name2 = input('Name #2: ')
    name3 = input('Name #3: ')
    appfile.write(name1 + eofm)
    appfile.write(name2 + eofm)
    appfile.write(name3 + eofm)
    appfile.close()
main()
```

file05

Other

Remove end of record marker (\n)

```
rstrip(‘\n’)

rcd = rcd.rstrip(‘\n’)
```

Concatenation

Joining two or more string together.

Useful in building output records

Use the plus (+) to join strings

Fname = ‘John’

Lname = ‘Five’

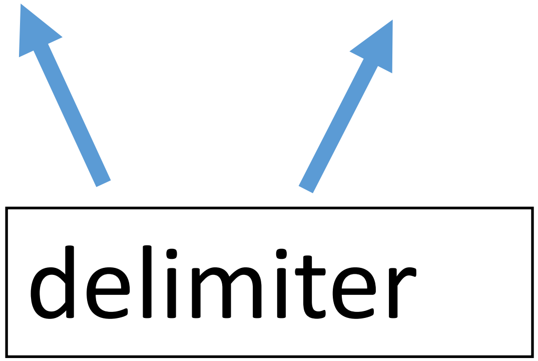
thename = Fname + ‘ ‘ + Lname

results: John Five

Multiple fields - output

Use concatenation to build output record.

```
ol = a + ',' + b + ',' + c + '\n'
```



delimiter

Delimiter

Used to separate fields

Comma (,) most common

Also the pipe (|)

Any character can be a delimiter

The delimiter cannot be in the data

Try This

```
def main():
    filename ='c:/temp/testfile2.txt'
    appfile = open(filename, 'w’)
    ans = 'Y'
    while (ans.upper() == 'Y'):
        name1 = input('enter name: ')
        addr = input('enter address: ')
        city = input('Enter city name: ')
        state = input('enter state code: ')
        rcd = name1 +',' + addr + ',' + city + ',' + state + '\n'
        appfile.write(rcd)
        ans = input('More input (y/n): ')
    appfile.close()
main()
```

file06

Multiple fields - input

Read a record with fields that have a delimiter

Use the readline command

split

Process input

Split –separate the fields

Form:

(list of variables) = line.split(‘delimiter’)

Example:

Data: 20,40

(a,b) = line.split(',’)

A = 20 and b = 40

Create test file

Open notepad

Type in:

```
25,50,75
10,20,30
33,44,55
```

Save as testfile5.txt in c:\temp folder

Try This

```
def main():
    filename ='c:/temp/testfile5.txt'
    rcdcnt = 0
    datafile = open(filename,'r')
    line=datafile.readline()
    print('here are the sales amount: ')
    while line != '':
        print(' line before split: ' + line)
        (a,b,c) =line.split(',')
        print(' A is: ' + a)
        print(' B is: ' + b)
        print(' C is: ' + c)
        line=datafile.readline()
        rcdcnt = rcdcnt+1
    datafile.close()
    print('\n'+ 'Number of records read: ' + str(rcdcnt))
    print (' ---done')
main()
```

file07

Reminder

Before splitting, check delimiters count

```
        line.count(‘,’)
```

Example:

```
 if line.count(‘,’) == 5:
     processrecord(line)
 else:
     processbadrecord(line)
```

Remember

For every open

there should be a close

Both open and close are done **ONCE**

Read and writing records are done one or more times

Multiple files

```
filenamein1 = ‘c:/temp/in1.txt’
filenamein2 = ‘c/temp/in2.txt’
filenameout = ‘c/temp/out.txt’
filein1 = open(filenamein1,’r’)
filein2 = open(filenamein2,’r’)
fileout = open(filenameout,’w’)
    .    .     .
filein1.close()
filein2.close()
fileout.close()
```

Try This Part 1

```
def closefiles(datafile):
    datafile.close()
def openfiles(filename, how):
    datafile=""
    try:
        datafile = open(filename, how)
    except FileNotFoundError:
        print(' ERROR - File not found: ', filename)
    except:
        print('unknown error')
    return datafile
```

file08

Try This Part 2

```
def main():
    filename ='c:/temp/testfile5.txt'
    rcdcnt = 0
    infile = openfiles(filename,'r')
```

Try This Part 3

```
if (infile!=""):
    line=infile.readline()
    print('here are the sales amount: ')
    while line != '':
        print(' line before split: ' + line)
        a,b,c =line.split(',')
        print(' A is: ' + a)
        print(' B is: ' + b)
        print(' C is: ' + c)
        line=infile.readline()
        rcdcnt = rcdcnt+1
```

Try This Part 4

Same indent As the if

Same indent as while

```
        closefiles(infile)
        print('\n'+ 'Number of records read: ' + str(rcdcnt))
    else:
        print(' program ending error in file')
    print (' ---done')
main()
```

Whole Program

```
def closefiles(datafile):
    datafile.close()
def openfiles(filename, how):
    datafile=""
    try:
        datafile = open(filename, how)
    except FileNotFoundError:
        print(' ERROR - File not found: ', filename)
    except:
        print('unknown error')
   return datafile
```

```
def main():
    filename ='c:/temp/testfile5.txt'
    rcdcnt = 0
    infile = openfiles(filename,'r')
    if (infile!=""):
        line=infile.readline()
        print('here are the sales amoount: ')
        while line != '':
            print(' line before split: ' + line)
            a,b,c =line.split(',')
            print(' A is: ' + a)
            print(' B is: ' + b)
            print(' C is: ' + c)
            line=infile.readline()
            rcdcnt = rcdcnt+1
        closefiles(infile)
        print('\n'+ 'Number of records read: ' + str(rcdcnt))
    else:
         print(' program ending error in file')
    print (' ---done')
main()
```

Multiple files example 1 of 5

```
from datetime import date
from datetime import time
from datetime import datetime
def logpgm(logfile,cond):
    dt = datetime.today()
    dts = str(dt) + '\n'
    if cond == 0:
        ps = 'program started '+ '\n'
    else:
        ps = 'program ended '+ '\n'
    logfile.write(ps)
    logfile.write(dts)
    return
```

file09

Multiple 2 of 5

```
def openfiles():
    filein ='C:/data/ptest/testfile7.txt'
    datafilein = open(filein,'r')

    fileout ='C:/data/ptest/testfile7out.txt'
    datafileout1 = open(fileout,'w')

    filelg ='C:/data/ptest/testfile7log.txt'
    datafileout2 = open(filelg,'w')

    return(datafilein,datafileout1,datafileout2)
```

Multiple 3 of 5

```
def main():
    (infile,outfile,logfile)=openfiles()
    logpgm(logfile,0)
    line = ''
    newline = ''
    rcdcnt = 0
    line=infile.readline()
```

Multiple 4 of 5

```
while line != '':
    print(line)
    a,b,c,d =line.split(',')
    print(' A is: ' + a)
    print(' B is: ' + b)
    print(' C is: ' + c)
    print(' D is: ' + d)
    line=infile.readline()
    rcdcnt = rcdcnt+1
    newline = a + ',' + d
    outfile.write(newline)
```

Multiple 5 of 5

```
 print('number of records: ',rcdcnt)
   logpgm(logfile,0)


   closefiles(infile,outfile,logfile)


main()
```

Whole program

```
from datetime import date
from datetime import time
from datetime import datetime
def logpgm(logfile,cond):
    dt = datetime.today()
    dts = str(dt) + '\n'
    if cond == 0:
        ps = 'program started '+ '\n'
    else:
        ps = 'program ended '+ '\n'
    logfile.write(ps)
    logfile.write(dts)
    return
def openfiles():
    filein ='C:/data/ptest/testfile7.txt'
    datafilein = open(filein,'r')
    fileout ='C:/data/ptest/testfile7out.txt'
    datafileout1 = open(fileout,'w')
    filelg ='C:/data/ptest/testfile7log.txt'
    datafileout2 = open(filelg,'w')
    return(datafilein,datafileout1,datafileout2)
def closefiles(fi,fo,fg):
    fi.close()
    fo.close()
    fg.close()
    return
```

```
def main():
    (infile,outfile,logfile)=openfiles()
    logpgm(logfile,0)
    line = ''
    newline = ''
    rcdcnt = 0
    line=infile.readline()
    while line != '':
        print(line)
        a,b,c,d =line.split(',')
        print(' A is: ' + a)
        print(' B is: ' + b)
        print(' C is: ' + c)
        print(' D is: ' + d)
        line=infile.readline()
        rcdcnt = rcdcnt+1
        newline = a + ',' + d
        outfile.write(newline)
    print('number of records: ',rcdcnt)
    logpgm(logfile,0)
    closefiles(infile,outfile,logfile)
main()
```

File - error handling

File processing errors

Most common:

file not found

read past end of file

read or write = file not open

File error processing

File error are system level errors

Use

```
try:
    statements  (when good)
except:
    statements ( when there is an error
```

Example program
file I/O
with try & except

```
from datetime import date
from datetime import time
from datetime import datetime
def logpgm(logfile,cond):
    dt = datetime.today()
    dts = str(dt) + '\n'
    if cond == 0:
        ps = 'program started '+ '\n'
    else:
        ps = 'program ended '+ '\n'
    logfile.write(ps)
    logfile.write(dts)
    return
```

file10

```
def openfiles():
    code = 0
    try:
        filein ='C:/temp/testfile7.txt'
        datafilein = open(filein,'r')
    except:
        code = 1
    try:
        fileout ='C:/temp/testfile7out.txt'
        datafileout1 = open(fileout,'w')
    except:
        code = 3
    try:
        filelg ='C:/temp/testfile7log.txt'
        datafileout2 = open(filelg,'w')
    except:
        code = 3
    return(datafilein,datafileout1,datafileout2, code)
```

```
def closefiles(fi,fo,fg):
    fi.close()
    fo.close()
    fg.close()
    return
def errorchk(code):
    msg='unknown file processing error'
    if code == 1:
        msg = ' error - input file'
    elif code == 2:
        msg = 'error - output file 1'
    else:
        msg = 'error - output file 2'
    return msg
def validline(line):
    msg = ''
    if line.count(',') != 3:
        msg = 'Recrd format invalid'
    return msg
```

```
def main():
    msg = ''
    rcdcnt = 0
    (infile,outfile,logfile,code)=openfiles()
    if code == 0:
        line = ''
        newline = ''
        line=infile.readline()
        while line != '':
            print(line)
            msg = validline(line)
            if msg =='':
                (a,b,c,d) =line.split(',')
                print(' A is: ' + a)
                print(' B is: ' + b)
                print(' C is: ' + c)
                print(' D is: ' + d)
                line=infile.readline()
                rcdcnt = rcdcnt+1
                newline = a + ',' + d
                outfile.write(newline)
            else:
                print(msg)
                print('record error: ',line)
                print(' Program terminated due to errors')
                line=''
    else:
        errorchk(code)
        msg = logpgm(logfile,0)
        print(msg)
    print('number of records: ',rcdcnt)
    logpgm(logfile,1)
    closefiles(infile,outfile,logfile)
main()
```

File Processing Statistics

Date and time stamp

At beginning of program, display the date and time program started

At end of program, display the date and time the program stoped

File Processing Statistics

Count record read, written to file

Display file counts at end of program

Other counts,

Count individual errors (how many of each error)

Example program

File20

Date and time stamp

Record count

Report at end of program

With / While /For

COMPARE

```
with open("/tmp/foo.txt") as file:
    data = file.read()
```

```
file = open("/tmp/foo.txt")
data = file.read()
file.close()
-----OR--------
file = open("/tmp/foo.txt")
try:
   data = file.read()
finally:
   file.close()
```

With/for – file processing loop

Simplifies File I/O

(both input and output)

With the open, python will detect End of File and Close the file

All you have to do is read and process the file records

TRY THIS

```
def main():
    print('pgm started')
    filenamein ='c:/data6/pgmdata/testfile1.txt'
    rcdcnt = 0
    with open(filenamein,'r') as infile:
        for line in infile:
            rcdcnt = rcdcnt+1
            line = line.strip()
            print (line)
    print('\n'+ 'Number of records read: ' + str(rcdcnt))
    print (' ---done')
main()
```

file30

With/while example

```
def main():
    print('pgm started')
    filenamein ='c:/data6/pgmdata/testfile1.txt'
    rcdcnt = 0
    with open(filenamein,'r') as infile:
        line=infile.readline()
        while line:
            rcdcnt = rcdcnt+1
            line = line.strip()
            print (line)
            line=infile.readline()
    print('\n'+ 'Number of records read: ' + str(rcdcnt))
    print (' ---done')
main()
```

file32

For loop example

```
def main():
    print('pgm started')
    filenamein ='c:/data6/pgmdata/testfile1.txt'
    rcdcnt = 0
    infile = open(filenamein,'r')
    for line in infile:
        rcdcnt = rcdcnt+1
        line = line.strip()
        print (line)
    infile.close()
    print('\n'+ 'Number of records read: ' + str(rcdcnt))
    print (' ---done')
main()
```

file34

While loop example

```
def main():
    print('pgm started')
    filenamein ='c:/data6/pgmdata/testfile1.txt'
    rcdcnt = 0
    infile = open(filenamein,'r')
    line = infile.readline()
    while line:
        rcdcnt = rcdcnt+1
        line = line.strip()
        print (line)
        line = infile.readline()
    infile.close()
    print('\n'+ 'Number of records read: ' + str(rcdcnt))
    print (' ---done')
main()
```

file36

Command reminder

read – input the whole file (string)

readline – input one record at a time (string)

write – write a record (string)
note you have to supply the end of record marker ‘\n’

open – open files for processing

close – close file

done