How to iterate over the file in python

Asked 13 years, 8 months ago Modified 5 years, 2 months ago Viewed 141k times

52

I have a text file with some hexadecimal numbers and i am trying to convert it to decimal. I could successfully convert it, but it seems before the loop exist it reads some unwanted character and so i am getting the following error.

```
Traceback (most recent call last):
  File "convert.py", line 7, in <module>
    print >>g, int(x.rstrip(),16)
ValueError: invalid literal for int() with base 16: ''
```

My code is as follows

```
f=open('test.txt','r')
g=open('test1.txt','w')
#for line in enumerate(f):
while True:
    x=f.readline()
    if x is None: break
    print >>g, int(x.rstrip(),16)
```

Each hexadecimal number comes in a new line for input

python

Share

Improve this question

Follow

edited Apr 20, 2011 at 16:22

eldarerathis

36.2k • 10 • 92 • 94

asked Apr 20, 2011 at 16:15

user567879

5,289 • 21 • 74 • 108

Well, take the debugger and figure out what the value of 'x' is causing the problem. Perhaps you have a file with a BOM? – user2665694 Apr 20, 2011 at 16:20

@RestRisiko: If there was a BOM, the error message would show that. No, it just tried to feed the empty string `''` to `int`. – user395760 Apr 20, 2011 at 16:21

5 Answers

Sorted by: Highest score (default)

The traceback indicates that probably you have an empty line at the end of the file. You can fix it like this:

78

```
f = open('test.txt','r')
g = open('test1.txt','w')
while True:
    x = f.readline()
    x = x.rstrip()
    if not x: break
    print >> g, int(x, 16)
```

On the other hand it would be better to use `for x in f` instead of `readline`. Do not forget to close your files or better to use `with` that close them for you:

```
with open('test.txt','r') as f:
    with open('test1.txt','w') as g:
        for x in f:
            x = x.rstrip()
            if not x: continue
            print >> g, int(x, 16)
```

Share

Improve this answer

Follow

edited Apr 20, 2011 at 17:04

answered Apr 20, 2011 at 16:20

joaquin

85.4k ● 31 ● 144 ● 155

24 The indentation-heaviness of the latter can be reduced in more recent versions: `with open('test.txt','r') as f, open('test1.txt','w') as g` – user395760 Apr 20, 2011 at 16:32

5 @delnan, Nice new thing! When indentation is not a problem I still prefer the two-lines form. I read it better... – joaquin Apr 20, 2011 at 16:38

1 oops! why the downvote? downvotes are not useful without explanation especially when the OP already accepted the answer. – joaquin Apr 20, 2011 at 16:42

1 The first program gives the same error of we have a blank line at the end – user567879 Apr 20, 2011 at 16:44

4 This is a great example of the "cleanness" of Python idiom -- no need for EOF, readline(), writeline() or any guts -- just concept: iterate over lines in one file, transform, and export to another file. Very nice! (+1) – Assad Ebrahim Feb 11, 2014 at 18:58

14

Just use `for x in f: ...`, this gives you line after line, is much shorter and readable (partly because it automatically stops when the file ends) and also saves you the `rstrip` call because the trailing newline is already stipped.

The error is caused by the exit condition, which can never be true: Even if the file is exhausted, `readline` will return an empty string, not `None`. Also note that you could still run into trouble with empty lines, e.g. at the end of the file. Adding `if line.strip() == "": continue` makes the code ignore blank lines, which is propably a good thing anyway.

Share Improve this answer Follow

answered Apr 20, 2011 at 16:20

user395760

4 This does NOT strip the trailing newlines: `python -c 'with open("file.txt") as f: print(repr([l[-1] for l in f]))'` returns many instances of `\n` on Python 2.7.12 and 3.4.5 – JamesTheAwesomeDude May 2, 2017 at 4:02

7

```
with open('test.txt', 'r') as inf, open('test1.txt', 'w') as outf:
    for line in inf:
        line = line.strip()
        if line:
            try:
                outf.write(str(int(line, 16)))
                outf.write('\n')
            except ValueError:
                print("Could not parse '{0}'".format(line))
```

Share

Improve this answer

Follow

edited Oct 8, 2019 at 11:33

taras

6,914 ● 10 ● 44 ● 53

answered Apr 20, 2011 at 16:40

Hugh Bothwell

56.5k ● 9 ● 90 ● 102

Hugh - does outf.write(int(line, 16)) work for you? My code is the exact same as yours except that I'm trying to write a dictionary d. I get a ValueError. See below: Traceback (most recent call last): File "./test.py", line 22, in <module> outf.write(d) ValueError: I/O operation on closed file – blehman Aug 18, 2013 at 6:22

2

You should learn about EAFP vs LBYL.

```
from sys import stdin, stdout
def main(infile=stdin, outfile=stdout):
    if isinstance(infile, basestring):
        infile=open(infile,'r')
    if isinstance(outfile, basestring):
        outfile=open(outfile,'w')
    for lineno, line in enumerate(infile, 1):
        line = line.strip()
         try:
             print >>outfile, int(line,16)
         except ValueError:
             return "Bad value at line %i: %r" % (lineno, line)

if __name__ == "__main__":
    from sys import argv, exit
    exit(main(*argv[1:]))
```

Share

edited Apr 23, 2011 at 17:25

answered Apr 20, 2011 at 16:22

Improve this answer

Follow

bukzor

38.4k ● 12 ● 82 ● 114

1

This is probably because an empty line at the end of your input file.

Try this:

```
for x in f:
    try:
        print int(x.strip(),16)
    except ValueError:
        print "Invalid input:", x
```

Share Improve this answer Follow

answered Apr 20, 2011 at 16:21

Rumple Stiltskin

10.3k ● 2 ● 24 ● 25