How to modify a text file?

Asked 16 years, 3 months ago Modified 2 years, 10 months ago

Viewed 592k times

234

I'm using Python, and would like to insert a string into a text file without deleting or copying the file. How can I do that?

python file text

Share

Improve this question

Follow

edited Jul 28, 2017 at 19:12

martineau

123k ● 29 ● 177 ● 311

asked Sep 24, 2008 at 6:30

Oscar

1 You can refer to this answer by Alex Martelli. – Alok Apr 2, 2014 at 4:29

1 stackoverflow.com/a/4358169/538284 – Omid Raha Apr 2, 2014 at 18:04

Possible duplicate of Writing on the topmost row of a csv file in python – Ani Menon Jun 15, 2016 at 5:52

@Ani the other post *is* a duplicate of Inserting Line at Specified Position of a Text File anyway and certainly there are clear composed answers here, Why not add your answer here instead of the other way? Accepted answer is *not* a requirement for a good question. – Bhargav Rao Jun 15, 2016 at 7:49

@BhargavRao Vote retracted. I should have found that duplicate though! – Ani Menon Jun 15, 2016 at 7:55

8 Answers

Sorted by: Highest score (default)

166

Unfortunately there is no way to insert into the middle of a file without re-writing it. As previous posters have indicated, you can append to a file or overwrite part of it using seek but if you want to add stuff at the beginning or the middle, you'll have to rewrite it.

This is an operating system thing, not a Python thing. It is the same in all languages.

What I usually do is read from the file, make the modifications and write it out to a new file called myfile.txt.tmp or something like that. This is better than reading the whole file into memory because the file may be too large for that. Once the temporary file is completed, I rename it the same as the original file.

This is a good, safe way to do it because if the file write crashes or aborts for any reason, you still have your untouched original file.

Share Improve this answer Follow

edited Oct 9, 2012 at 12:11
user229044 ♦
239k ● 41 ● 344 ● 346

answered Sep 24, 2008 at 6:57
Adam Pierce
34.3k ● 23 ● 71 ● 89

---

4 Do unix tools like awk/sed do something similar in their code?
– Manish Gill Mar 22, 2013 at 19:16

---

1 It's not true that this is the same in all languages. In ActionScript: fileStream.openAsync(filename,FileMode.UPDATE); Then I can go anywhere in the file I want and change anything.
– AndrewBenjamin Jul 11, 2014 at 3:58

---

3 @AndrewBenjamin Do you know what system calls ActionScript is making? Is there a possibility that openAsync reads the file and writes a new one after the call?
– AlexLordThorsen Dec 2, 2014 at 22:23

---

1 @Rawrgulmuffins I do not. However, I do know that it isn't reading the entire file into memory, as I have used it to handle filesizes of several GB. I suspect it's the same as writing with C# streamwriter. I view python as a tool for doing small things quickly, rather than large scale development and file manipulation. – AndrewBenjamin Mar 26, 2015 at 21:48

---

6 @AndrewBenjamin, the user isn't asking about seeking around in the file and changing it (every language I know of can do that); he is asking about inserting text, which is different than simply changing/overwriting what is already in the file. Maybe in practical application it is different, but nothing I can find in the ActionScript API indicates that it behaves any different from any other language in this regard.
– eestrada Jul 22, 2015 at 14:44

146

Depends on what you want to do. To append you can open it with "a":

```
with open("foo.txt", "a") as f:
    f.write("new line\n")
```

If you want to preprend something you have to read from the file first:

```
with open("foo.txt", "r+") as f:
     old = f.read() # read everything in the file
     f.seek(0) # rewind
     f.write("new line\n" + old) # write the new line
```

Share Improve this answer Follow

answered Sep 24, 2008 at 6:35

Armin Ronacher
32.5k ● 14 ● 67 ● 69

11 Just a small addition, to use the `with` statement in Python 2.5 you need to add "from **future** import with_statement". Other than that, opening files with the `with` statement is definitely more readable and less error-prone than manual closing. – Alexander Kojevnikov Sep 24, 2008 at 6:48

2 You might consider the `fileinput` helper lib with handles the dirty open/read/modify/write/replace routine nicely when using the `inline=True` arg. Example here: stackoverflow.com/a/2363893/47390 – mikegreenberg Feb 1, 2012 at 21:14

7 It's not a style I use, D.Rosado, but when using the with style, I don't think you need to manually close. The with

keeps track of the resource it creates. – Chris May 14, 2012 at 17:47

10 You *do not* need to manually close the file. That's the whole point of using "with" here. (Well, actually, Python does this as soon as the file object is garbage collected, which in CPython happens when the name bound to it goes out of scope... but other implementations don't, and CPython might stop doing it some day, so "with" is recommended)
– Jürgen A. Erhard Jun 22, 2013 at 11:16

Can we replace some characters in it? – alper Feb 2, 2022 at 15:26

84

The `fileinput` module of the Python standard library will rewrite a file inplace if you use the inplace=1 parameter:

```
import sys
import fileinput

# replace all occurrences of 'sit' with 'SIT' and inse
for i, line in enumerate(fileinput.input('lorem_ipsum.
    sys.stdout.write(line.replace('sit', 'SIT'))  # re
    if i == 4: sys.stdout.write('\n')  # write a blank
```

Share Improve this answer Follow

edited Jan 26, 2017 at 15:49
Open AI - Opting Out
24.1k ● 7 ● 65 ● 102

answered Nov 28, 2009 at 7:25
Dave
2,073 ● 2 ● 18 ● 19

2 How is this expected to work in python3? I just ported an app that had some code like this from python to python3 and I

just could not get this to work right at all. The 'line' variable is a bytes type, I tried decoding it into unicode and then modifying it and then encoding it back to bytes but it just would not work right. It raised some exception I can't remember off the top of my head. Are people using fileinput inplace=1 in python3 with any success? – robru Feb 21, 2015 at 5:08

4 @Robru: here's Python 3 code – jfs Dec 19, 2016 at 8:02

14 But its no problem cos you tested it first on an unimportant file right? – Paula Livingstone Nov 18, 2017 at 13:55

34

Rewriting a file in place is often done by saving the old copy with a modified name. Unix folks add a `~` to mark the old one. Windows folks do all kinds of things -- add .bak or .old -- or rename the file entirely or put the ~ on the front of the name.

```
import shutil
shutil.move(afile, afile + "~")

destination= open(aFile, "w")
source= open(aFile + "~", "r")
for line in source:
    destination.write(line)
    if <some condition>:
        destination.write(<some additional line> + "\n

source.close()
destination.close()
```

Instead of `shutil`, you can use the following.

```
import os
```

```
os.rename(aFile, aFile + "~")
```

Share Improve this answer Follow

edited Feb 2, 2022 at 15:28
alper
3,361 ● 11 ● 61 ● 116

answered Sep 24, 2008 at 10:27
S.Lott
391k ● 82 ● 517 ● 788

1 Looks good. Wondering if .readlines() is better than iterating the source? – bozdoz Apr 10, 2013 at 15:48

2 @bozdoz: iterating is better since readlines reads the whole file. Not good for big files. Of course, this presumes that you can do your modifications in such a localized way. Sometimes you can't, or your code gets a lot more complicated. – Jürgen A. Erhard Jun 22, 2013 at 11:20

@S.Lott: `os.rename(aFile, aFile + "~")` will modify the name of the source file, not creating a copy. – Patapoom Mar 12, 2020 at 9:39

15

Python's mmap module will allow you to insert into a file. The following sample shows how it can be done in Unix (Windows mmap may be different). Note that this does not handle all error conditions and you might corrupt or lose the original file. Also, this won't handle unicode strings.

```
import os
from mmap import mmap

def insert(filename, str, pos):
    if len(str) < 1:
        # nothing to insert
        return

    f = open(filename, 'r+')
    m = mmap(f.fileno(), os.path.getsize(filename))
    origSize = m.size()

    # or this could be an error
    if pos > origSize:
        pos = origSize
    elif pos < 0:
        pos = 0

    m.resize(origSize + len(str))
    m[pos+len(str):] = m[pos:origSize]
    m[pos:pos+len(str)] = str
    m.close()
    f.close()
```

It is also possible to do this without mmap with files opened in 'r+' mode, but it is less convenient and less efficient as you'd have to read and temporarily store the contents of the file from the insertion position to EOF - which might be huge.

Share Improve this answer Follow

answered Sep 25, 2008 at 0:41

mhawke

86.9k ● 10 ● 121 ● 142

14

As mentioned by Adam you have to take your system limitations into consideration before you can decide on approach whether you have enough memory to read it all into memory replace parts of it and re-write it.

If you're dealing with a small file or have no memory issues this might help:

Option 1) Read entire file into memory, do a regex substitution on the entire or part of the line and replace it with that line plus the extra line. You will need to make sure that the 'middle line' is unique in the file or if you have timestamps on each line this should be pretty reliable.

```
# open file with r+b (allow write and binary mode)
f = open("file.log", 'r+b')
# read entire content of file into memory
f_content = f.read()
# basically match middle line and replace it with itse
f_content = re.sub(r'(middle line)', r'\1\nnew line',
# return pointer to top of file so we can re-write the
string
f.seek(0)
# clear file content
f.truncate()
# re-write the content with the updated content
f.write(f_content)
# close file
f.close()
```

Option 2) Figure out middle line, and replace it with that line plus the extra line.

```
# open file with r+b (allow write and binary mode)
f = open("file.log" , 'r+b')
# get array of lines
f_content = f.readlines()
# get middle line
middle_line = len(f_content)/2
# overwrite middle line
f_content[middle_line] += "\nnew line"
```

```
# return pointer to top of file so we can re-write the
string
f.seek(0)
# clear file content
f.truncate()
# re-write the content with the updated content
f.write(''.join(f_content))
# close file
f.close()
```

Share Improve this answer

Follow

answered Nov 19, 2012 at 23:24

Maxime R.

171 ●1 ●6

1

Wrote a small class for doing this cleanly.

```
import tempfile

class FileModifierError(Exception):
    pass

class FileModifier(object):

    def __init__(self, fname):
        self.__write_dict = {}
        self.__filename = fname
        self.__tempfile = tempfile.TemporaryFile()
        with open(fname, 'rb') as fp:
            for line in fp:
                self.__tempfile.write(line)
        self.__tempfile.seek(0)

    def write(self, s, line_number = 'END'):
        if line_number != 'END' and not isinstance(lin
            raise FileModifierError("Line number %s is
line_number)
        try:
            self.__write_dict[line_number].append(s)
        except KeyError:
                self.__write_dict[line_number] = [s]

    def writeline(self, s, line_number = 'END'):
        self.write('%s\n' % s, line_number)

    def writelines(self, s, line_number = 'END'):
        for ln in s:
            self.writeline(s, line_number)

    def __popline(self, index, fp):
        try:
            ilines = self.__write_dict.pop(index)
            for line in ilines:
                fp.write(line)
        except KeyError:
            pass

    def close(self):
        self.__exit__(None, None, None)

    def __enter__(self):
        return self

    def __exit__(self, type, value, traceback):
        with open(self.__filename,'w') as fp:
            for index, line in enumerate(self.__tempfi
                self.__popline(index, fp)
                fp.write(line)
            for index in sorted(self.__write_dict):
                for line in self.__write_dict[index]:
                    fp.write(line)
        self.__tempfile.close()
```

Then you can use it this way:

```
with FileModifier(filename) as fp:
    fp.writeline("String 1", 0)
    fp.writeline("String 2", 20)
    fp.writeline("String 3")  # To write at the end of
```

Share Improve this answer Follow

answered Feb 2, 2016 at 9:42

ananth krishnan
259 ● 2 ● 5

This doesn't work for me personally, it does add text to the file but it removes everything first! – Bret Hawker Jan 12, 2019 at 19:47

Indeed, this doesn't work at all. Shame, because it seemed like a good idea. – Mario Krušelj Jun 14, 2019 at 9:39

-1

If you know some unix you could try the following:

Notes: $ means the command prompt

Say you have a file my_data.txt with content as such:

```
$ cat my_data.txt
This is a data file
with all of my data in it.
```

Then using the `os` module you can use the usual `sed` commands

```
import os

# Identifiers used are:
my_data_file = "my_data.txt"
command = "sed -i 's/all/none/' my_data.txt"

# Execute the command
os.system(command)
```

If you aren't aware of sed, check it out, it is extremely useful.

Share Improve this answer Follow

answered Nov 2, 2018 at 18:43

G. LC
824 ● 2 ● 10 ● 27

3 It's not Pythonic at all – DarkSuniuM May 27, 2019 at 1:02