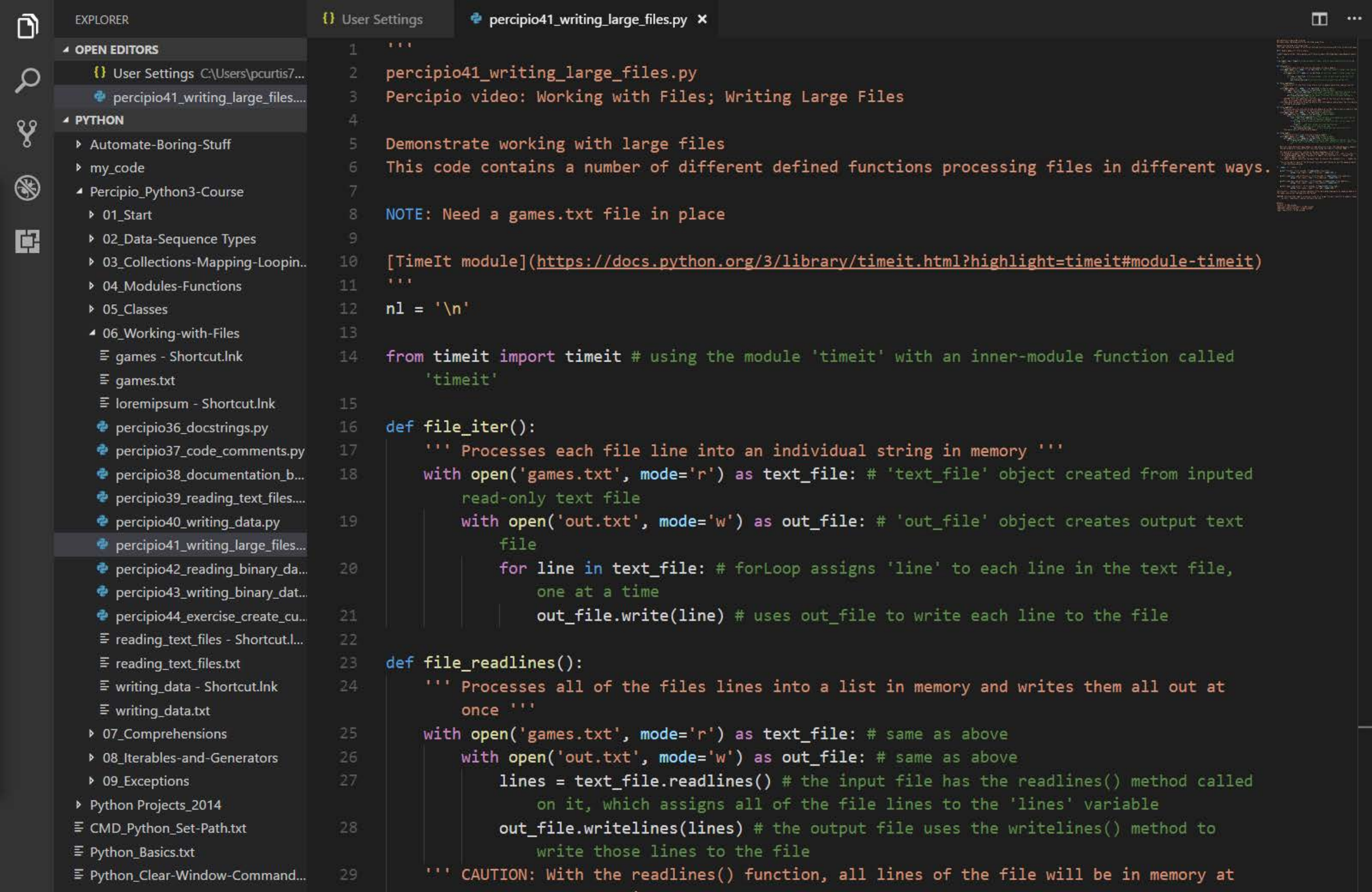
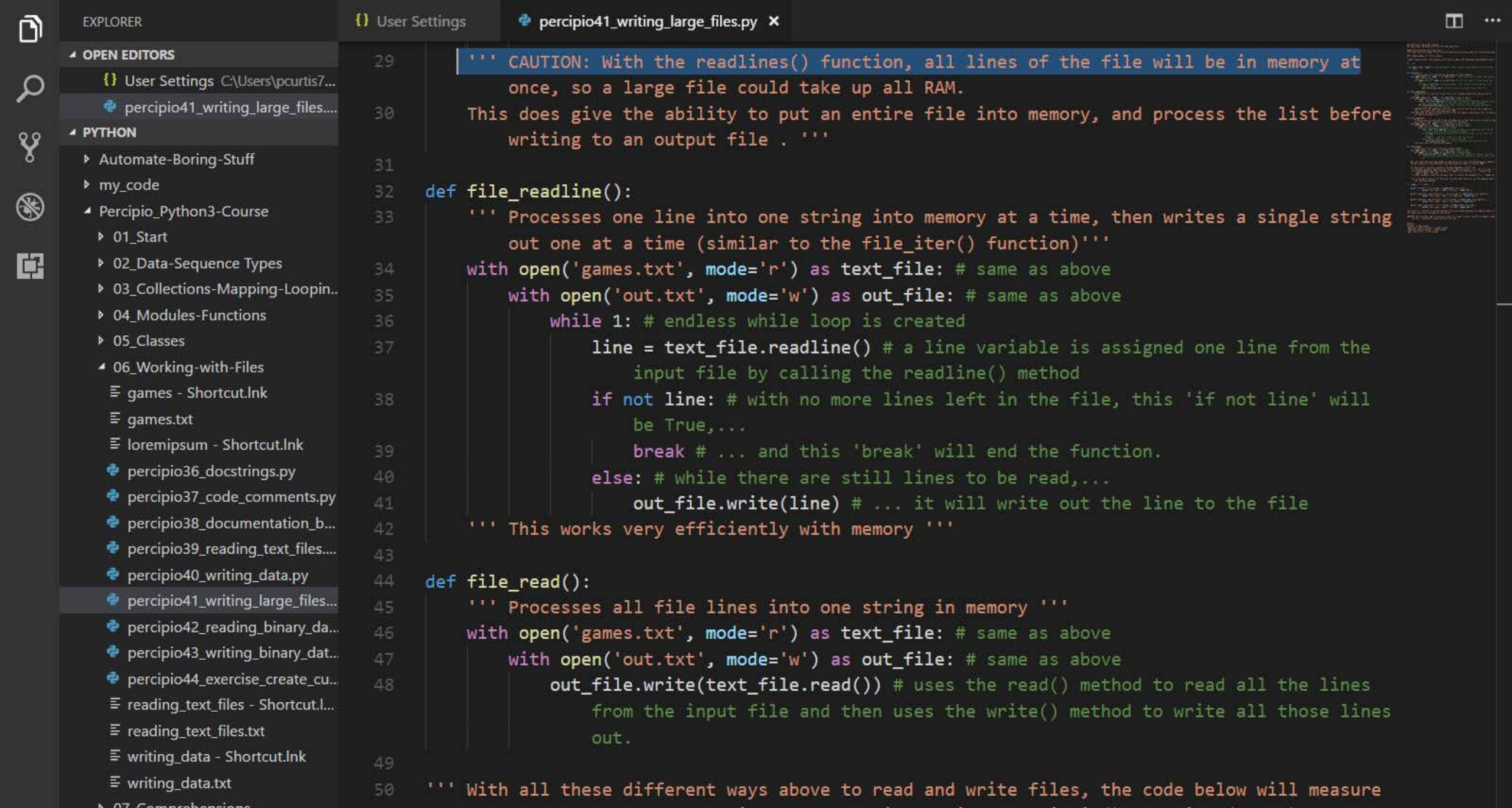

EXPLORER
User Settings
percipio41_writing_large_files.py
OPEN EDITORS
User Settings C:\Users\pcurtis7...
percipio41_writing_large_files....
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mapping-Loopin..
04_Modules-Functions
05_Classes
06_Working-with-Files
games - Shortcut.lnk
games.txt
loremipsum - Shortcut.lnk
percipio36_docstrings.py
percipio37_code_comments.py
percipio38_documentation_b...
percipio39_reading_text_files....
percipio40_writing_data.py
percipio41_writing_large_files...
percipio42_reading_binary_da...
percipio43_writing_binary_dat...
percipio44_exercise_create_cu...
reading_text_files - Shortcut.l...
reading_text_files.txt
writing_data - Shortcut.lnk
writing_data.txt
07_Comprehensions
08_Iterables-and-Generators
09_Exceptions
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Basics.txt
Python_Clear-Window-Command...
'''
percipio41_writing_large_files.py
Percipio video: Working with Files; Writing Large Files
Demonstrate working with large files
This code contains a number of different defined functions processing files in different ways.
NOTE: Need a games.txt file in place
[TimeIt module](https://docs.python.org/3/library/timeit.html?highlight=timeit#module-timeit)
'''
nl = '\n'
from timeit import timeit # using the module 'timeit' with an inner-module function called 'timeit'
def file_iter():
''' Processes each file line into an individual string in memory '''
with open('games.txt', mode='r') as text_file: # 'text_file' object created from inputed read-only text file
with open('out.txt', mode='w') as out_file: # 'out_file' object creates output text file
for line in text_file: # forLoop assigns 'line' to each line in the text file, one at a time
out_file.write(line) # uses out_file to write each line to the file
def file_readlines():
''' Processes all of the files lines into a list in memory and writes them all out at once '''
with open('games.txt', mode='r') as text_file: # same as above
with open('out.txt', mode='w') as out_file: # same as above
lines = text_file.readlines() # the input file has the readlines() method called on it, which assigns all of the file lines to the 'lines' variable
out_file.writelines(lines) # the output file uses the writelines() method to write those lines to the file
''' CAUTION: With the readlines() function, all lines of the file will be in memory at


EXPLORER
User Settings
percipio41_writing_large_files.py
OPEN EDITORS
User Settings C:\Users\pcurtis7...
percipio41_writing_large_files....
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mapping-Loopin...
04_Modules-Functions
05_Classes
06_Working-with-Files
games - Shortcut.lnk
games.txt
loremipsum - Shortcut.lnk
percipio36_docstrings.py
percipio37_code_comments.py
percipio38_documentation_b...
percipio39_reading_text_files....
percipio40_writing_data.py
percipio41_writing_large_files...
percipio42_reading_binary_da...
percipio43_writing_binary_dat...
percipio44_exercise_create_cu...
reading_text_files - Shortcut.l...
reading_text_files.txt
writing_data - Shortcut.lnk
writing_data.txt
''' CAUTION: With the readlines() function, all lines of the file will be in memory at once, so a large file could take up all RAM.
This does give the ability to put an entire file into memory, and process the list before writing to an output file . '''
def file_readline():
''' Processes one line into one string into memory at a time, then writes a single string out one at a time (similar to the file_iter() function)'''
with open('games.txt', mode='r') as text_file: # same as above
with open('out.txt', mode='w') as out_file: # same as above
while 1: # endless while loop is created
line = text_file.readline() # a line variable is assigned one line from the input file by calling the readline() method
if not line: # with no more lines left in the file, this 'if not line' will be True,...
break # ... and this 'break' will end the function.
else: # while there are still lines to be read,...
out_file.write(line) # ... it will write out the line to the file
''' This works very efficiently with memory '''
def file_read():
''' Processes all file lines into one string in memory '''
with open('games.txt', mode='r') as text_file: # same as above
with open('out.txt', mode='w') as out_file: # same as above
out_file.write(text_file.read()) # uses the read() method to read all the lines from the input file and then uses the write() method to write all those lines out.
''' With all these different ways above to read and write files, the code below will measure




EXPLORER

{} User Settings | percipio41_writing_large_files.py ×

OPEN EDITORS
- {} User Settings C:\Users\pcurtis7...
- percipio41_writing_large_files....

PYTHON
- Automate-Boring-Stuff
- my_code
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mapping-Loopin..
  - 04_Modules-Functions
  - 05_Classes
  - 06_Working-with-Files
    - games - Shortcut.lnk
    - games.txt
    - loremipsum - Shortcut.lnk
    - percipio36_docstrings.py
    - percipio37_code_comments.py
    - percipio38_documentation_b...
    - percipio39_reading_text_files....
    - percipio40_writing_data.py
    - percipio41_writing_large_files...
    - percipio42_reading_binary_da...
    - percipio43_writing_binary_dat...
    - percipio44_exercise_create_cu...
    - reading_text_files - Shortcut.l...
    - reading_text_files.txt
    - writing_data - Shortcut.lnk
    - writing_data.txt
  - 07_Comprehensions
  - 08_Iterables-and-Generators
  - 09_Exceptions
- Python Projects_2014
- CMD_Python_Set-Path.txt
- Python_Basics.txt
- Python_Clear-Window-Command...
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-Scripts....
- Python_Tutorials.md

```
50  ''' With all these different ways above to read and write files, the code below will measure
        the performance of these different functions using the timeit() function (WOW!!)
51
52      For each print() function running in the main namespace, it will use:
53      * a statement parameter specifies the function name to execute (i.e: 'stmt='file_iter()').
54      * a setup parameter specifies from where to import this function (i.e.: 'setup='from
            __main__ import file iter')
55      * a number parameter specifies how many times to execute the statement (i.e.: 'number=10')
56
57      This is called for each of the different functions and then print out the speed at which
            the functions executed.
58  '''
59  if __name__ == '__main__': #
60
61      print('iterator: %5.3f seconds' % timeit(stmt='file_iter()',
62                  setup='from __main__ import file_iter', number=10)) #
63
64      print('readlines() and writelines(): %5.3f seconds' % timeit(stmt='file_readlines()',
65                  setup='from __main__ import file_readlines', number=10)) #
66
67      print('readline() and writeline(): %5.3f seconds' % timeit(stmt='file_readline()',
68                  setup='from __main__ import file_readline', number=10)) #
69
70      print('read() and write(): %5.3f seconds' % timeit(stmt='file_read()',
71                  setup='from __main__ import file_read', number=10)) #
72  '''
73  The iterator() function is the most pythonic offering a good compromise of speed an memory use
74  The read() and write() methods are the fastest
75
76  CAUTION: Function that read or write all lines of a large file will use alot of memory (read()
        , write(), readlines(), and writelines() do this.)
77  '''
78  '''
79  RESULTS:
80  iterator: 0.196 seconds
81  readlines() and writelines(): 0.196 seconds
82  readline() and writeline(): 0.200 seconds
83  read() and write(): 0.136 seconds
84  '''
```