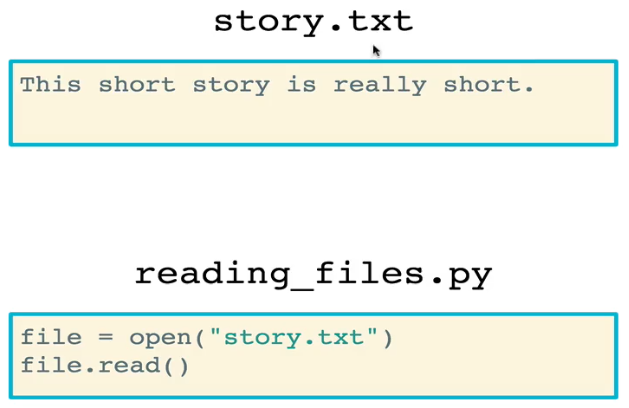
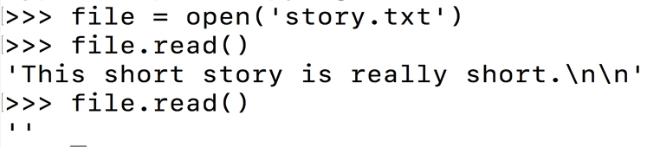
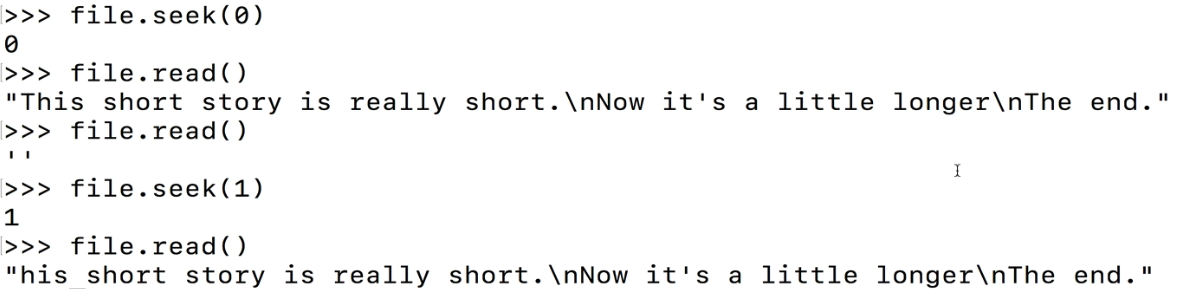
* File I/O stands for file input and output – it is the idea of working with files that are NOT Python files and do not necessarily contain code
  + This is a critical aspect of web development and data science
  + CSS files, CSV files, PDFs
* Reading basic files – read data in from existing files
  + Reading files can done with the built-in ***open*** function. *Open* takes as an argument (at a minimum) the file path and filename, and it returns to you a file object
  + A file object contains information such as the data itself, the information about the file (permissions, type of file, last modified, etc.)
  + <https://docs.python.org/3/library/functions.html#open>
  + Note: the open() method creates an active link (or “pipeline”) between Python and the file of interest. If update the file outside of Python, then any further manipulation of that file within Python will reflect the updated version of the file
    - This link remains open until you manually close the file in Python
  + You must then read in a file object using the *read* method
    - When given no arguments, the read method reads the entire file



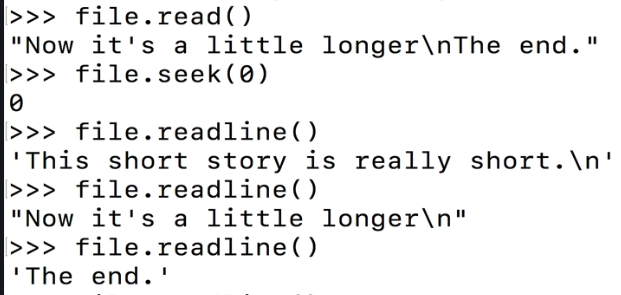
* Cursor Movement
  + If you try to read in a file a second time, Python will return an empty string. Why?



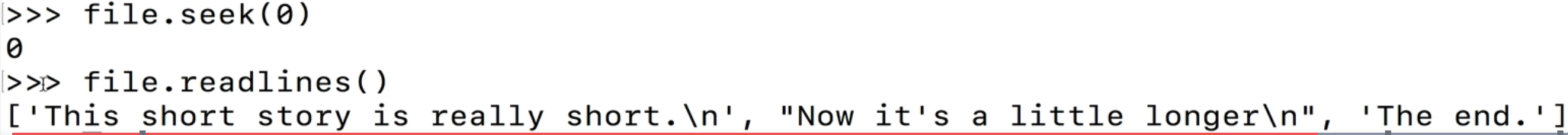
* + - Python reads files using a **cursor**. Think of the cursor of “where you are” in the file
    - After the file is read the first time, the cursor goes to the end of the file. Thus, if you try to read the file again, you get an empty string because there is no more file to read
  + The cursor is moved with the *seek* method
    - The **seek()** method takes as an argument the index location in the file that you want to move the cursor to.
      * The cursor is usually moved



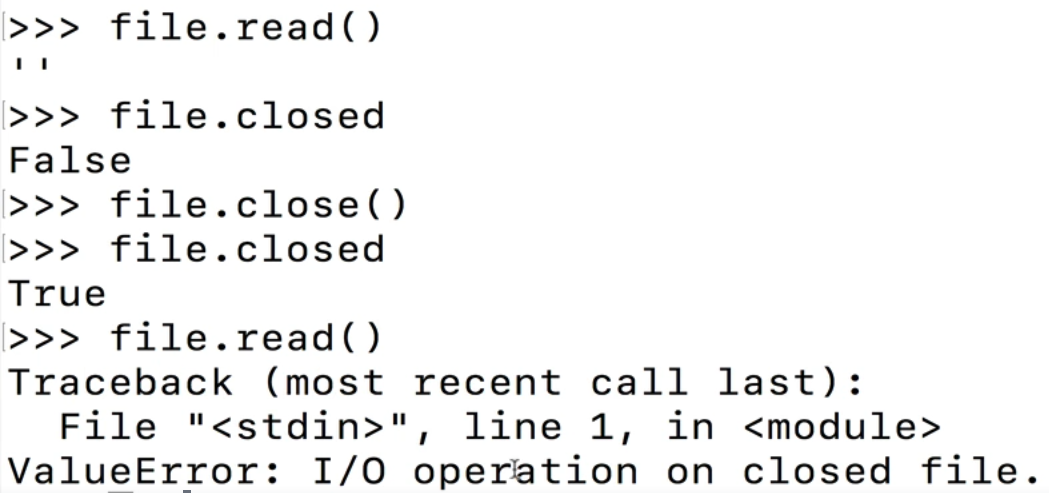
* + The **.readline()** method is useful when you don’t necessarily want to read the entire file at once.
    - readline() stops when a newline (\n) character is encountered



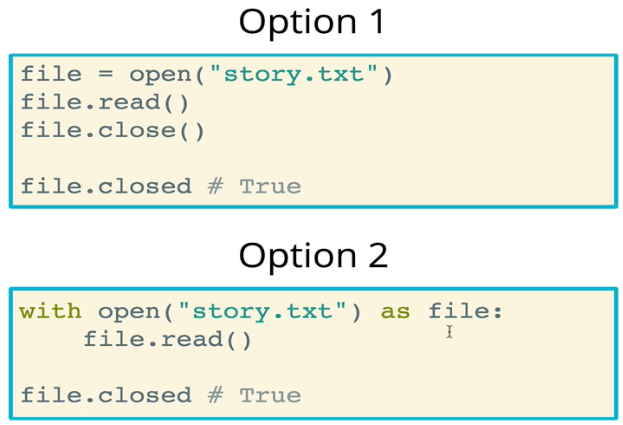
* + The **.readlines()** method returns a *list* of lines from the stream
    - Similar to .read(), it will put your cursor at the very end of the file



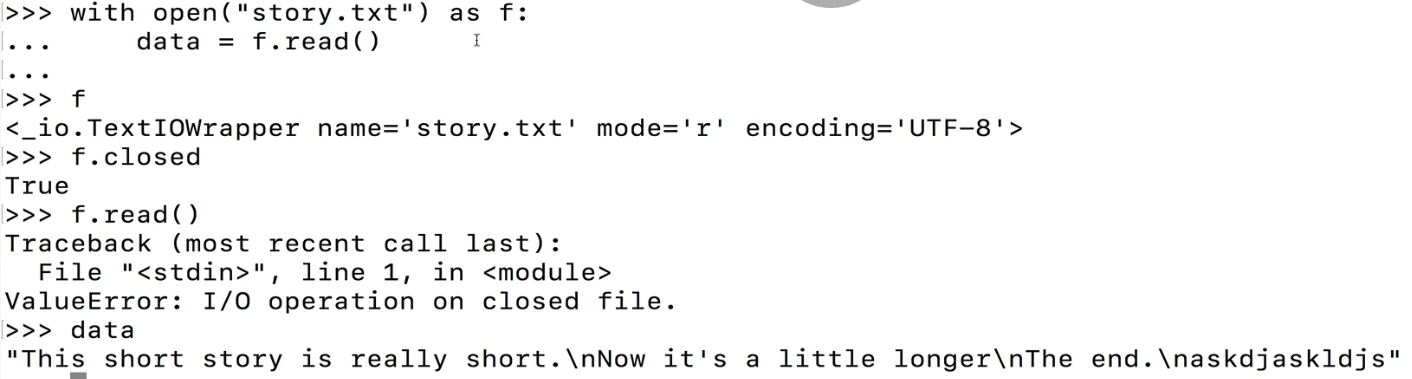
* **Closing files** is very important as it frees up system resources. This is done using the **.close()** method
  + You can check if a file is closed with the **.closed** attribute
  + Once closed, a file cannot be read again until you re-open it



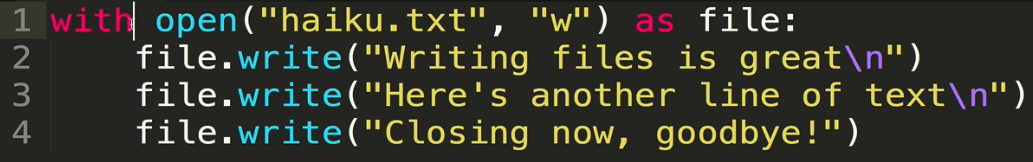
* The **with** block is an alternative and (usually) preferred method for reading files
  + Basic syntax is in the example below



* + The benefit of the *with* block is that, regardless of what happens when executing the code within the *with* block, the file will always be closed. It does not need to be closed manually

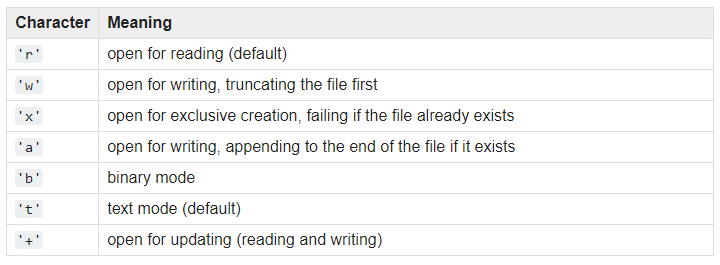


* + Behind the scenes, the \_\_enter\_\_ and \_\_exit\_\_ methods are called when using the *with* statement on a file object
    - For file objects, \_\_enter\_\_ simply returns the file object, while \_\_exit\_\_ closes the file object
  + *With* is not limited to files – we can create custom objects that work with *with* statements as long as they have \_\_enter\_\_ and \_\_exit\_\_ statements
* Writing to files
  + You must still use the open function to write to a file
  + In order to actually write, you have to pass in the “w” flag as a second argument to the open() method
    - By default, with no flags, files are opened in read mode (“r”)
  + In the example below (written in a *with* statement), a file is opened and three new lines are written



* + IMPORTANT: When working with the standard “w” flag, the write method completely overwrites the existing content of the file
  + Also, the file you are writing to need not exist before opening and writing to it. If it does not exist, then it will be created
* All about **file modes –** the open() method can be used to open files in different modes
  + Recall that *mode* is one of the keyword arguments for the open() method





* + Flags
    - r – read to a file (no writing); this is the default mode
    - w – write to a file (previous contents of the file are removed)
    - a – append to a file (previous contents not removed)
      * Important drawback: append always appends to the end of the file – you cannot control where the cursor goes with the .seek() method
    - r+ - read and write to a file (writing based on cursor location)
      * Very commonly used when you need to make changes to a file with existing data and those changes are not at the end of the file
      * It only works with an existing file, unlike “a” and “w” which will create new files
      * By default, the cursor will begin writing at the beginning of the file and will overwrite anything that is already there. It’s not an insert, it’s an overwrite
      * However, you can use seek() method within the *with open* block to move the cursor around
      * You cannot write directly into the middle or beginning of a file without overwriting the contents. In order to do that, you must (for example) save the existing content as a variable,