# fileslice Documentation

***Release 0.0.2***

**Mohammad Alghafli**

**Oct 10, 2018**

# Contents

This tutorial gives an introduction to how to use fileslice python library and its features.

This is not a python tutorial. You are expected to have general knowledge in python before you start this tutorial.

CHAPTER 1

# The problem

Usually, programs open a file and look at it as a whole. However, it happens sometimes that your program needs to look at part of the file and never look at other parts of the file. For example, in a download manager, you may want to start multiple download threads, each thread downloads part of the file. In file servers, if the client requests part of the file, the server will open only the requested part of the file.

CHAPTER 2

# The solution

This library allows you to open part of a file. The part you open behaves like a regular independant file. Multiple parts of a file can be opened simultaneously. You can read or write to each part from multiple threads at the same time. Seeking to the beginning of a part is does not affect the seek position of other parts since each part has its own seek position and the library takes care of thread safety.

CHAPTER 3

# Content

## 3.1 Installation

### 3.1.1 Requirements

**The major requirement of fileslice is python 3. fileslice is a python 3 library and was never tested in python 2. So first make sure your version of python is 3.**

### 3.1.2 Installation

Make sure you have pip for python 3 installed.

On windows install using pip by running the command:

```
pip install fileslice
```

Or on linux:

```
pip3 install fileslice
```

Of course, pip command should be in your PATH environment variable. If you are using windows there is a good chance pip is not in your PATH. In this case you should specify the full pip path. Search about how to use pip on windows if you are having trouble.

Try to import fileslice to be sure it was installed successfully:

```
>>> import fileslice
>>>
```

If it is imported without errors, you are ready to use it. You may want to have a look at one or more of the following documents:

- *Quick Guide* For usage examples.

- *fileslice Reference* **This is the library reference. All classes and functions** documentation is here.

## 3.2 Quick Guide

### 3.2.1 Usage example

This is a typical usage example:

```
from fileslice import Slicer

#let's open a file for reading
r = open('example.png', 'br')

#create a slicer for the file
slicer = Slicer(r)

#the slicer behaves like a function. call it to create as many fileslices
#as you want
start = 5   #the beginning of our partial file is at 5
size = 95   #the size of the part is 95 bytes so our end is 99
fileslice = slicer (start, size)     #this is a file like object

#now we have a fileslice file from byte 5 to byte 99.
#the initial partial file seek position is 0.
print(fileslice.read())  #will print from byte 5 to 99.

#now our seek position is at the end of the fileslice file
#that is byte 100 of the full file
try:
    #if we seek to a position out of the fileslice file range (from 0 to 95)
    fileslice.seek(200)
except ValueError:
    #an exception will be thrown
    print('error while seeking to 200')

#we can seek from the end of the fileslice or from current fileslice
#seek position
#let's seek to fifth byte from the end of the fileslice. that is byte 95
fileslice.seek(-5, 2)
```

### 3.2.2 Equally sized slices

```
from fileslice import Slicer

#let's open a file for reading
r = open('example.png', 'bw')

#create a slicer for the file
slicer = Slicer(r)

#slicer.slices(<total size>, <number of slices>)
#returns a list of file slices
#each slice will be 250 bytes
```

(continues on next page)

(continued from previous page)

```
slice_list = slicer.slices(1000, 4)

#if you used:
#slice_list = slicer.slices(1000, 3)
#the first two slices will be 332. the last slice will be 334
#the last slice can be larger than the rest in some cases like this one
```

### 3.2.3 Further readings

In *fileslice Reference* you will find the library reference.

## 3.3 fileslice Reference

**Date** 2018-10-10

**Version** 0.0.2

**Authors**

- Mohammad Alghafli <thebsom@gmail.com>

File slice is a part of a file. This library allows you to open a binary file and see only part of it. You specify the start and end of the file part you want to see. Read and write operations will always start at the specified start position and will never exceed the specified end position. Here is an example of how to use it:

```
from fileslice import Slicer

#let's open a file for reading
r = open('example.png', 'br')

#create a slicer for the file
slicer = Slicer(r)

#the slicer behaves like a function. call it to create as many fileslices
#as you want
start = 5   #the beginning of our partial file is at 5
size = 95   #the size of the part is 95 bytes so our end is 99
fileslice = slicer (start, size)     #this is a file like object

#now we have a fileslice file from byte 5 to byte 99.
#the initial partial file seek position is 0.
print(fileslice.read())  #will print from byte 5 to 99.

#now our seek position is at the end of the fileslice file
#that is byte 100 of the full file
try:
    #if we seek to a position out of the fileslice file range (from 0 to 95)
    fileslice.seek(200)
except ValueError:
    #an exception will be thrown
    print('error while seeking to 200')

#we can seek from the end of the fileslice or from current fileslice
#seek position
```

(continues on next page)

(continued from previous page)

```
#let's seek to fifth byte from the end of the file. that is byte 95
fileslice.seek(-5, 2)
```

This library also works in writable files. Multiple threads can be used to read and write to different file slices from the same file. Just make sure you do not use the original opened file or the fileslices (and the original file object) will be confused.

**class** fileslice.**FileSlice**(*f*, *start*, *size*, *lock*)

---

**Note:** The *Slicer* class is a convinient way to create instances of this class.

---

A file slice. An instance of this class is a file-like object that only looks at a specific region in the original file object. Multiple slices can be created from the same file object and each slice will have its own seek position and you can read and write to each slice independantly. *FileSlice* instances generated from the same *Slicer* object are thread-safe.

In addition to the class methods, the following methods call the methods with the same name from the original file object:

- readable.
- writable.
- seekable.
- isatty.
- fileno.

**close**()
Same as *file.close()* but for the slice. All file access operations will raise ValueError after the file is closed. The original file object is not closed.

**end**
The end position of the slice. The slice will not read or write if the new position is ahead of this value.

**flush**()
Flushes the original file.

**read**(*size=-1*)
Same as *file.read()* but for the slice. Does not read beyond *self.end*.

**seek**(*offset*, *whence=0*)
Same as *file.seek()* but for the slice. Returns a value between *self.start* and *self.size* inclusive.

**raises:** ValueError if the new seek position is not between 0 and *self.size*.

**tell**()
Same as *file.tell()* but for the slice. Returns a value between *self.start* and *self.size* inclusive.

**write**(*b*)
Same as *file.write()* but for the slice.

**raises:** EOFError if the new seek position is > *self.size*.

**writelines**(*lines*)
Same as *file.writelines()* but for the slice.

**raises:** EOFError if the new seek position is > *self.size*.

**class** fileslice.**Slicer**(*f*)

File slicer. Use this class to slice a file into multiple file slices and use each slice as if it was an independant file.

**slices**(*size*, *n=3*)

Create equal sized slices of the file. The last slice may be larger than the others.

**args:**

- size (int): The full size to be sliced.
- n (int): The number of slices to return.

**returns:** A list of *FileSlice* objects of length (n).

## 3.4 Indices and tables

- genindex
- modindex
- search

# Python Module Index

## f

# Index