




Python and Files

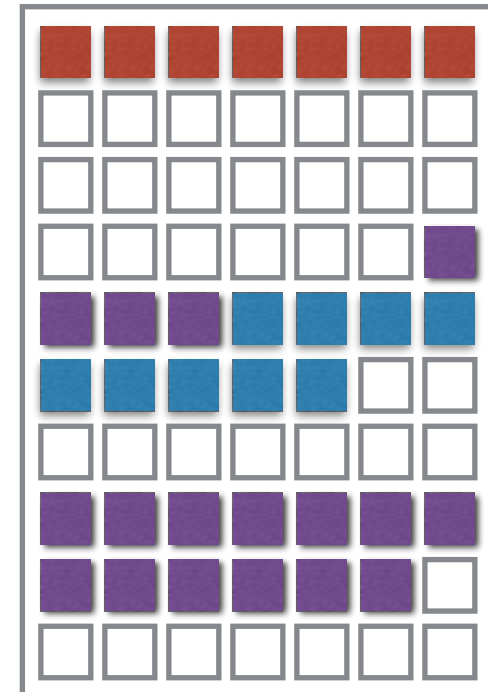
February 28, 2016

What are files?

Files are locations in secondary and persistent storage where we keep data that is used by our programs. Secondary storage is a fancy way to say hard drive, SSD, SD flash card, or thumb drive. A file system is used to organize files on storage.

File System

-  Table - Identifies location on the drive, the name of the file and the directory.
-  Files - Take up space on the drive, have a start and length and may be broken up
-  Empty - The file system keeps track of empty space



Types of files

Files fall into a couple of groups, **binary** and **text**. Text files are normally human readable, whereas binary files are not human readable. Binary files are used by programs to efficiently store data and instructions. Text files contain readable text, but can sometimes contain special text that requires special tools to read.

Text Files

`<html>hi</html>` → HTML

name, age, location → CSV

`def func():
 print('Hi')` → Python

The Old Brown Shoe → Text

我爱学校 → Text

Binary Files

```
01000010 01101100 01100001 01100011 01101011 01100010 01101001 01110010 01100100 00100000
01110011 01101001 01101110 01100111 01101001 01101110 01100111 00100000 01101001 01101110
00100000 01110100 01101000 01100101 00100000 01100100 01100101 01100001 01100100 00100000
01101111 01100110 00100000 01101110 01101001 01100111 01101000 01110100 00001101 00001010
01010100 01100001 01101011 01100101 00100000 01110100 01101000 01100101 01110011 01100101
00100000 01100010 01110010 01101111 01101011 01100101 01101110 00100000 01110111 01101001
01101110 01100111 01110011 00100000 01100001 01101110 01100100 00100000 01101100 01100101
01100001 01110010 01101110 00100000 01110100 01101111 00100000 01100110 01101100 01111001
00001101 00001010 01000001 01101100 01101100 00100000 01111001 01101111 01110101 01110010
00100000 01101100 01101001 01100110 01100101 00001101 00001010 01011001 01101111 01110101
00100000 01110111 01100101 01110010 01100101 00100000 01101111 01101110 01101100 01111001
00100000 01110111 01100001 01101001 01110100 01101001 01101110 01100111 00100000 01100110
01101111 01110010 00100000 01110100 01101000 01101001 01110011 00100000 01101101 01101111
01101101 01100101 01101110 01110100 00100000 01110100 01101111 00100000 01100001 01110010
01101001 01110011 01100101 00001101 00001010
```

.exe → Programs

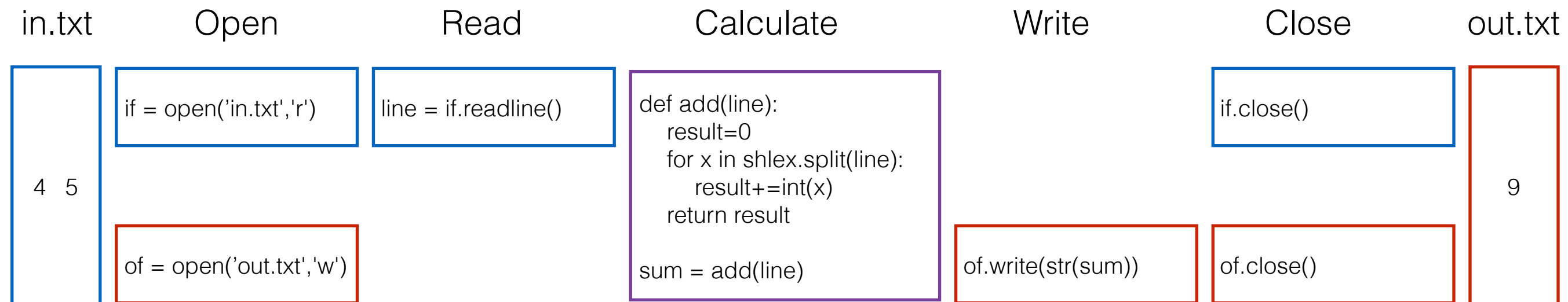
.jpg, gif, .png → Images

.pyc → Python byte code

Why do I care about files?

or why do we write code?

We **enjoy** taking inputs, making some observations, making some calculations and turning out something useful. A lot of times our program input and output are stored in files. Always our code is stored in files.



To use files we need to understand a basic set of **file operations**

Open

To open a file means to tell the filesystem that you are ready to read from or write to a file. If the file you are writing to doesn't exist the file system will create it. The filesystem will prevent any other program from writing to a file that you've opened.

Text Files

```
# read text
f = open('file name','r')

# write text
f = open('file name','w')

# read and write text
f = open('file name','r+')
```

Binary Files

```
# read binary
f = open('file name','rb')

# write binary
f = open('file name','wb')

# read and write binary
f = open('file name','r+b')
```

Read or Write

Read a file to pull the file contents from secondary storage into memory, like into a variable. Write to a file to store data in memory on to secondary storage.

Read

```
# read a single line of text  
line = f.readline()
```

```
# read all contents of a file, text or binary  
all_of_file = f.read()
```

Write

```
# write a line of text to a file  
f.write('this is a line\n')
```

```
# write a block of text or binary data to a file  
f.write(data)
```

Close

Finally you must close the file after you finish reading and writing to it. You do this with the close function. If you do not close a file, it may not be available for other programs to use.

`f.close()`

The easier way, with

The with statement takes care of closing an open file for you, even if an error occurs while working with the file. All operations on the file must be in a block of code beneath the with statement.

```
with open('sample','r') as f:  
    for line in f:  
        print('line:',line)
```

f only has scope beneath the with statement

Python File Operations

Open

```
# read text
f = open('file name','r')

# write text
f = open('file name','w')

# read and write text
f = open('file name','r+')

# read binary
f = open('file name','rb')

# write binary
f = open('file name','wb')

# read and write binary
f = open('file name','r+b')
```

Read

```
# read a single line of text
line = f.readline()

# read all contents of a file, text or binary
all_of_file = f.read()
```

Close

```
f.close()
```

Write

```
# write a line of text to a file
f.write('this is a line\n')

# write a block of text or binary data to a file
f.write(data)
```

With

```
with open('sample','r') as f:
    for line in f:
        print('line:',line)
```

Exercise 1

Create a new file named sample.py in your project. Then add the following blocks of code to it.

Read one line of a text file.

```
f = open('sample.txt','r')
first_line = f.readline()
print('first line:', first_line)
f.close()
```

Iterate through all lines of a text file.

```
f = open('sample.txt','r')
for line in f:
    print('line:', line)
f.close()
```

Read the entire file in a single text call.

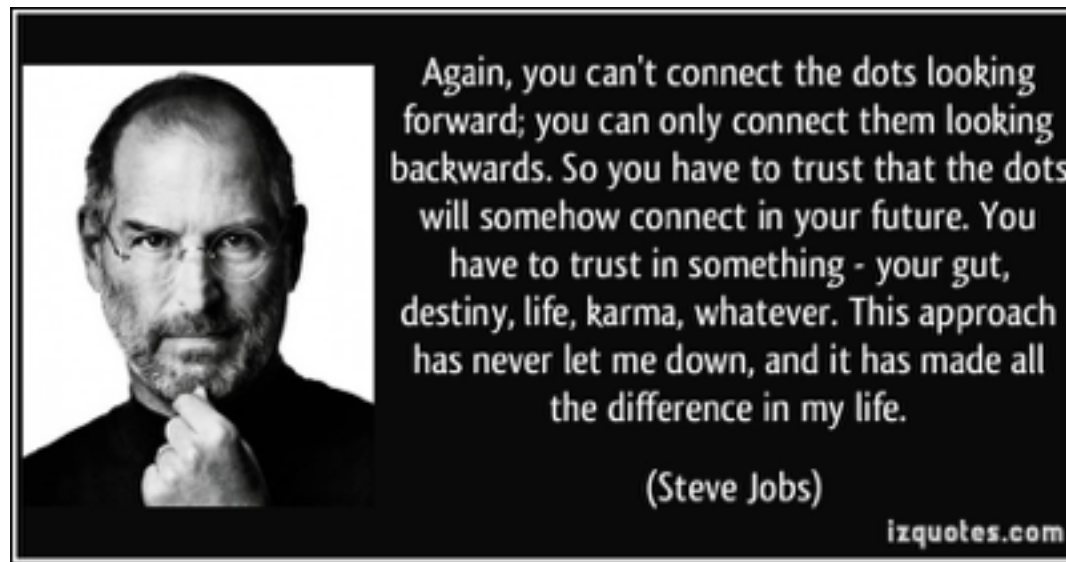
```
f = open('sample.txt','r')
entire_file = f.read()
print('entire file:', entire_file)
f.close()
```

Exercise 2

- create another file in your project and name it add.py. Add the following to add.py.
- create a variable named result and initialize it to zero
- open file numbers.dat for text read
- open sum.dat for text write
- iterate through all lines of numbers.data
 - in your loop convert each line to an int, int(), and add the int to result
- after you've iterated through all lines, convert result to a string, str(result)
- and write result string to sum.dat
- close both numbers.dat and sum.dat
- open sum.dat to check the result

Exercise 3

- create a new file in your project called image.py
- in image.py add the following
- import Blocks, this will let you use functions found in blocks
- create variable and assign an existing image file name to it, we'll read this file
- create variable and assign an output file name to it
- using with statement open the existing file for read as binary as in field
 - read all file contents into a variable file_contents
 - call Blocks.start() passing file contents in as a parameter, store the result of this call into a new variable new_contents
- using with statement open your output file for write as binary
 - write new_contents to your output file



- Innovation <https://vimeo.com/77911159>
- Boston dynamics <https://youtu.be/rVlhMGQgDkY>

