

Files

(permanent data collections)

1

This video will discuss the use of files in Python.

Files

- Permanent data collections stored on the hard drive...
 - files remain on hard drive after Python is closed or restarted.
- Amount of data that can be stored is limited only by hard drive space.
- Slower for data retrieval than lists and dictionaries.
- Only certain types of files can be read directly in Python – e.g. text files, kml file.
- Reading GIS files requires modules (e.g. arcpy).

2

Files are data collections that stored on the hard drive – they exist independently of Python.

Files can store large amounts of data because they are not limited to the computers internal memory.

Retrieving data from files is slower and less convenient than for lists, dictionaries, and other Python data collections.

Python can only read certain types of files directly – these types of files include text files and kml files.

Python requires modules, like arcpy, to work with GIS files.

Reading from a text file

```
txtFile = r"C:\Python_workshop\data\points.txt"
```


1. Open the file...

```
open_File = open(txtFile)
```

2. Get a line from the file...

```
line = open_File.readline()[:-1]
```

Remove return character ("\n")



3. Split the line to separate values...

```
line = line.split(",")
```

4. Repeat steps 2 and 3 for successive lines.
 - Files are read in order from the top down
 - Use while loop (slide 34) to read through file.

3

To read from text, kml, or similar files:

- 1) First, open the file using the **open** function - this creates a **file object**.
- 2) Next, get a line from the file using the **readline** method of the file object. The last character in a line is "\n" (which is the command to start a new line). The "\n" should typically be removed.
- 3) The line retrieved from the text file will always be a string. To extract individual values from the line, you will typically need to use the **split** method on the string.
- 4) The readline and split steps can be repeated until the entire file has been processed. Python bookmarks the last line that was retrieved with the readline method so each time the method is used, a new line is retrieved from the file. The repetition of steps 2 and 3 can be done most efficiently using a loop (which will be discussed in a later video).

Reading from a line from a text file - a closer look

```
points.txt - Notepad
File Edit Format View Help
0,white oak,1136486.276080,856551.833411,50
1,silver maple,1136363.768700,856522.818504
2,red oak,1136020.425620,855898.997990,85

line = open_File.readline()[:-1]
'0white oak,1136486.276080,856551.833411,50'

comma separates values
line = line.split(",")
['0', 'white oak', '1136486.276080', '856551.833411']
```

Let's look at a demonstration of the process for reading a text file:

`open_File` refers to a file object. Using the **readline** method gets the first line from the text file.

The “\n” can be removed from the end of the string by retrieving all but the last character in the line.

The line is a string so we use the **split** method to divide the string anywhere there is a comma.

In this case, we select a comma as the split character because the values in the line are separated by commas.

The split method puts the string segments in a list for easy retrieval.

Writing to a text file

```
new_file = r"C:\Python_workshop\newFile.txt"
```

- Open a text file in write mode...

```
o_new_file = open(new_file, "w")
```

Write mode; will create a new file if it does not exist, will overwrite file if it does exist

- Write line to file...

```
line = "This is only a test\n"  
o_new_file.write(line)
```

End line

5

To write to a text file:

- 1) Use the **open** function to open a file in **write mode**. It is important to note that opening a file in write mode will erase a file if it already exists.
- 2) Next, define a string to add to the file. Add "\n" to the end of the string to end the line in the text file.
- 3) Use the **write** method of the file object to add the line to the file.

Closing a file

- Always close a file when finished reading or writing it.
 - Removes file lock
- Close file with...

```
o_new_file.close()
```



open file object

6

Files should be closed when you're finished working with them. This will remove the file lock and allow you to view the file using Notepad or other text editor.

Use the file object's **close** method to close an open file.

Getting help on data types and data collections

- Use the **help** function to list all properties and methods for a particular object:
 - help (int) # integers
 - help (float) # decimal numbers
 - help (str) # strings
 - help (list) # lists
 - help (dict) # dictionaries
 - help (file) # files

7

The **help** function can be used to get further information on any objects in Python. The object names for the data types and collections that we've previously discussed are provided here.