Working with Text Files and Rasters

1

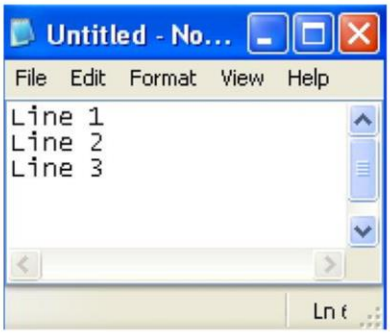
This video will discuss how to work with rasters that have been converted to a text file format. This topic is an application that uses many of the topics that we have covered previously in the course.

Reading text files – a review

- Use the readline method to get current line.
- Repeat the readline method to get successive lines.
- readline returns null value (‘’) at end of file

```
>>> oTxtFile = open(txtFile)
>>> oTxtFile.readline()
'Line 1\n'
>>> oTxtFile.readline()
'Line 2\n'
>>> oTxtFile.readline()
'Line 3\n'
>>> oTxtFile.readline()
''
```

null value

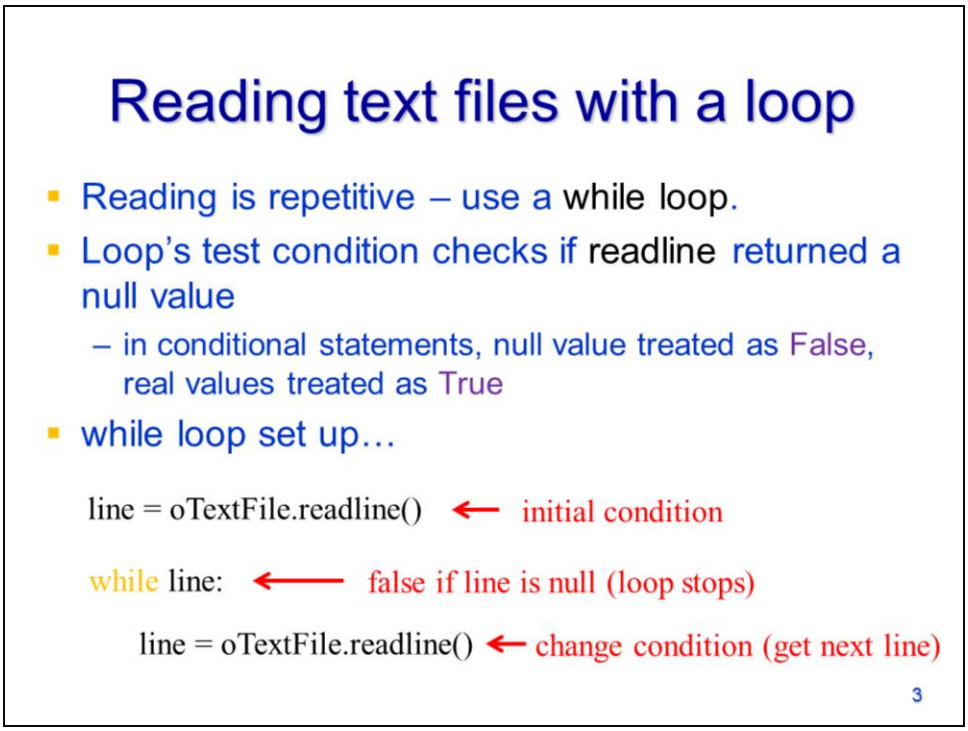


2

Recall that text files are read by iteratively using the **readline** method of the file object.

Each time the readline method is used, a new line of the text file is retrieved.

When the end of the file is reached, the readline method will return a null value.



The repetition of reading a file makes it an ideal task for a loop. A while loop is preferred since we generally don't know in advance how many lines are contained in a text file.

The while loop will run until the readline method returns a null value when the end of the file is reached.

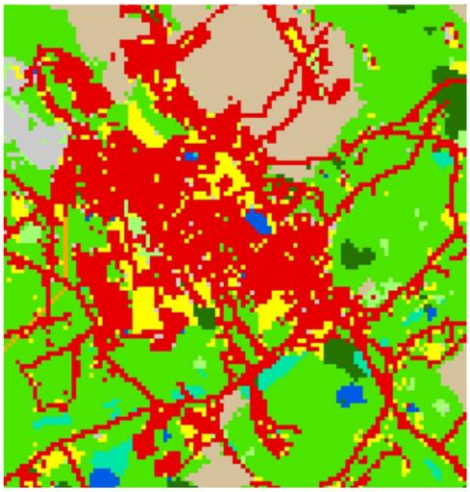
To initialize the while loop condition, get the 1st line from the file.

In the while loop header line, the result from the readline method becomes the test condition. As long as the readline method returns a non-null value, then the loop will continue.

Inside the loop, the last line should repeat the readline method – this will get the next line from the text file and allow the script to progress.

Raster data

- Data represented as pixels
- Each pixel is assigned a single number…
  - integer for categorical data (i.e. land cover)
  - decimal for continuous data (i.e. elevation)



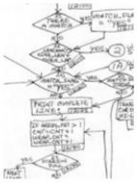
4

Rasters are a common data format in GIS. These data are represent as pixels.

Each pixel has a single number value assigned to it. This number may be an integer as in the case of categorical data such as a land cover dataset. Pixel values may also be decimal numbers as in the case of continuous data such as a digital elevation model.

Working with raster data

- Out-of-the-box raster tools require the Spatial Analyst extension…
  - sometimes requires convoluted workflows



  - not always available

- Python provides other options…
  - convert raster to array
  - rasters can be converted to a text file format
  - text files can be converted back to a raster

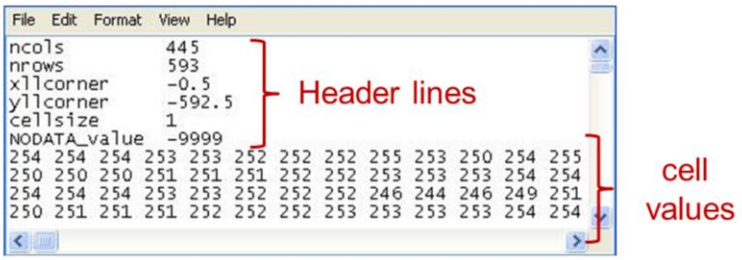
In ArcGIS, the tools for working with raster data are contained within the Spatial Analyst extension which requires an additional license.

Python provides a number of other options for working with raster data that are freely available. The option available to us, given our current Python knowledge, is to work with the raster as a text file. The text file can be modified and then converted back to a raster.

Converting a raster to a text file

arcpy.RasterToASCII_conversion(input_raster, output_textFile)

- Conversion does not require spatial analyst
- Works on all raster formats (ESRI grids, .img, .jpeg, etc.)

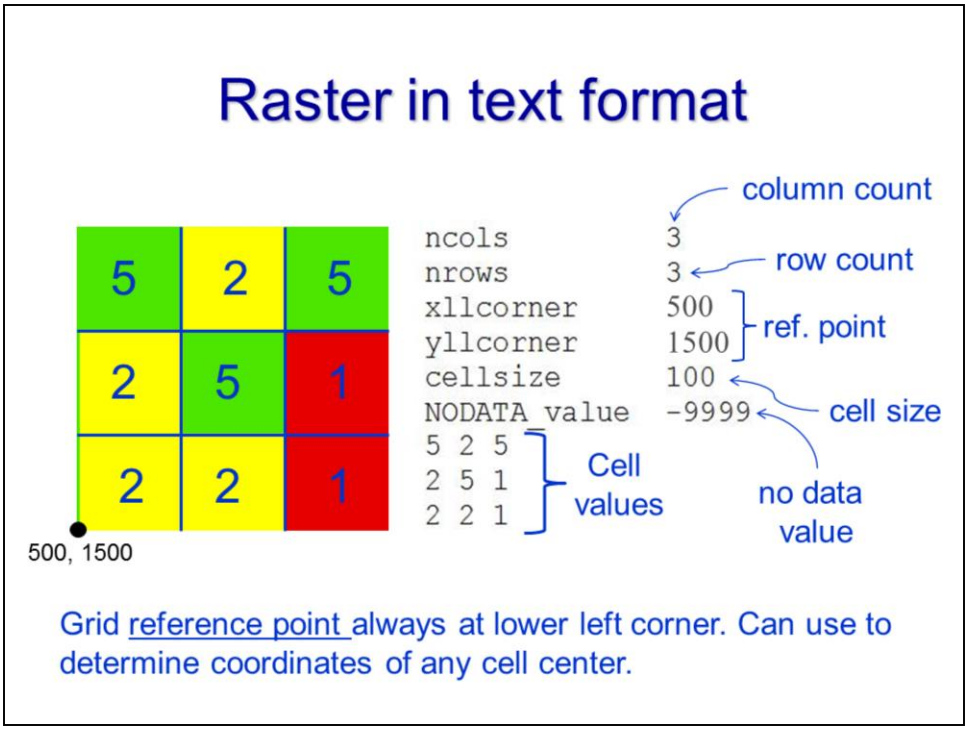


6

An ArcTool can convert the raster into a text file format.

This tool will work on all rasters and requires only the basic ArcGIS license.

The resulting text file will contain some basic raster properties in the header section. The header is followed by the main body of the text file which contains the actual pixel values.



Let's look at an example of a small raster in a text format.

The 1st line of the text file specifies the number of columns in the raster.

The 2nd line specifies the number of rows

The 3rd and 4th lines specify the coordinates of the lower left corner of the raster – this coordinate georeferences the raster so that it can be linked to real world coordinates.

The 5th line indicates the width of the pixels

The 6th line indicates the value used for null pixels which contain no real data.

Starting with line 7, the value for each pixel is listed with a space separating each value. There is one line of text file that corresponds to each row in the raster.

Modifying a raster in text format

- Not practical to modify the text files directly
- Easier to write a new text file with modified values

original

```
ncols          3
nrows          3
xllcorner      500
yllcorner      1500
cellsize       100
NODATA_value   -9999
5 2 5
2 5 1
2 2 1
```

new

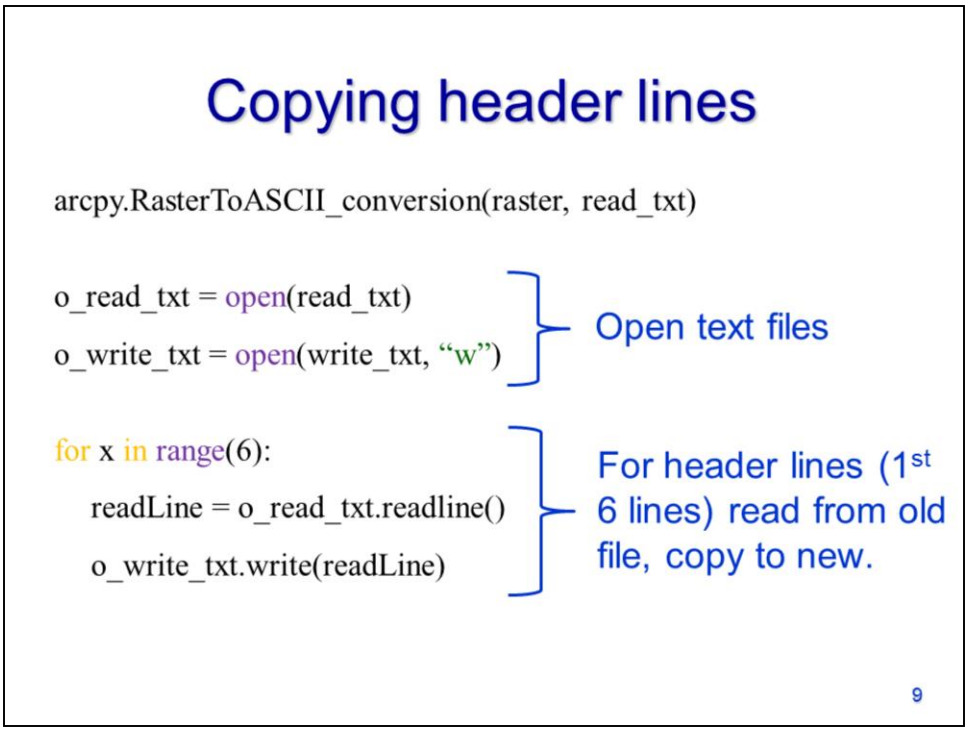
```
ncols          3
nrows          3
xllcorner      500
yllcorner      1500
cellsize       100
NODATA value   -9999
1 0 1
0 1 0
0 0 0
```

8

When modifying a raster in a text file format, it is more efficient to create an entirely new text file than to modify the original.

The approach that I'll present will be to read the original raster text file

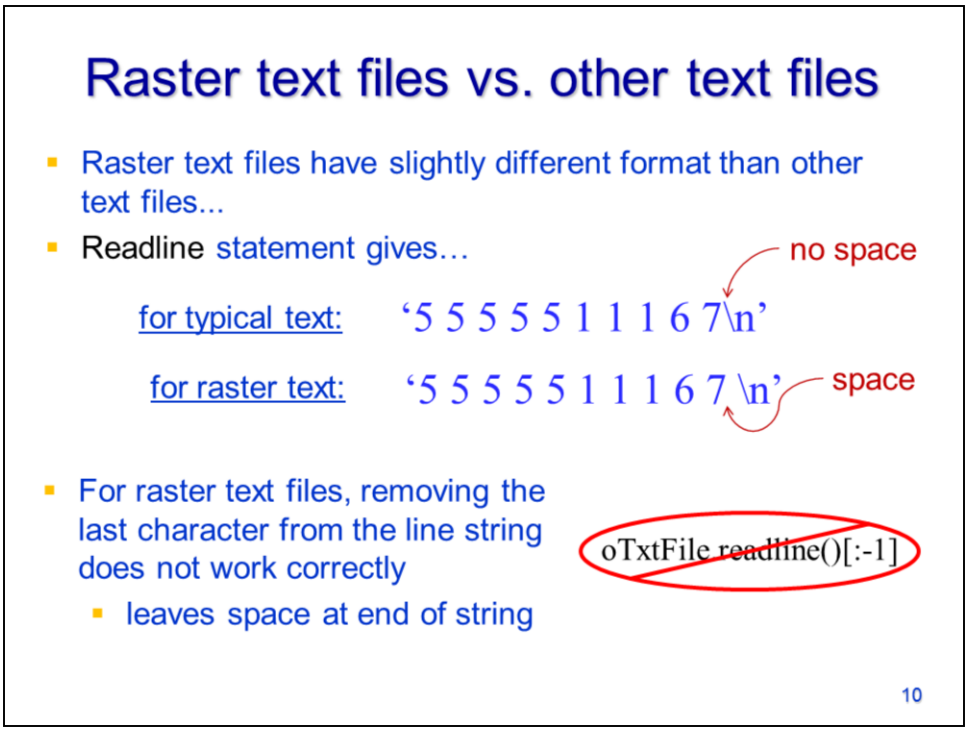
and create a new text file with the new pixel values.



The first step is in modifying a raster is to convert it to a text file using arcpy's RasterToASCII tool.

Then open the raster text file in read mode. Open a second text file in write mode to store the new raster.

Next, copy the header lines from the original text file to the new text file – this will give the new raster the same spatial reference information as the original. Use a for loop to iterate through the first 6 lines of the original raster text file. Within the loop, read a line from the original file and write it to the new file.

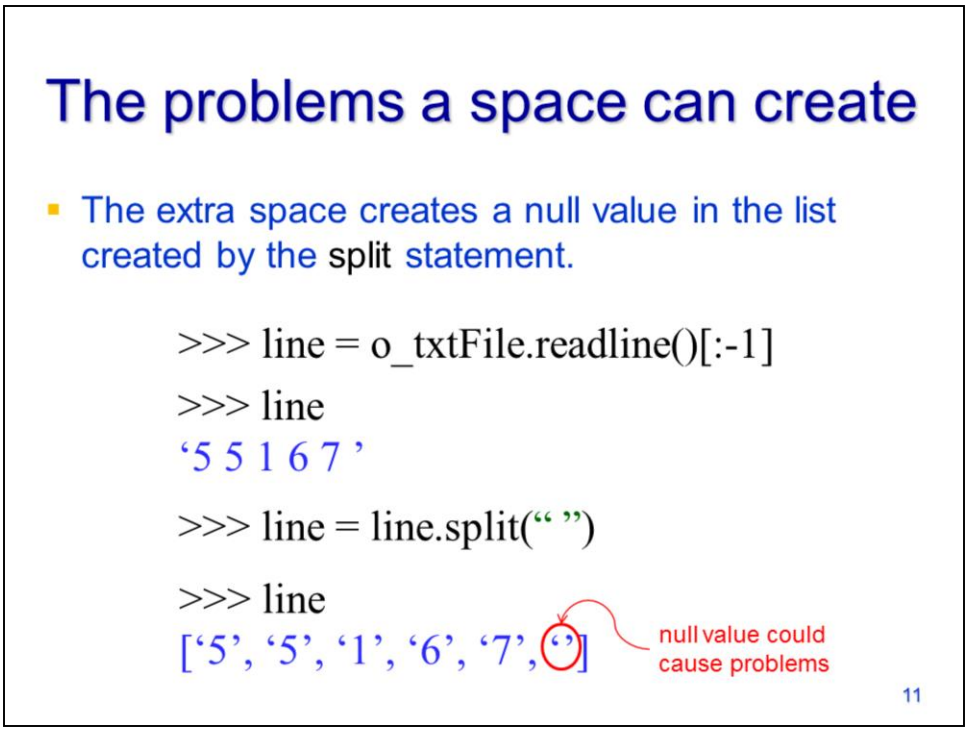


Raster files are slightly different from other text files…

In a typical text file, the "\n" character usually occurs immediately after the last value.

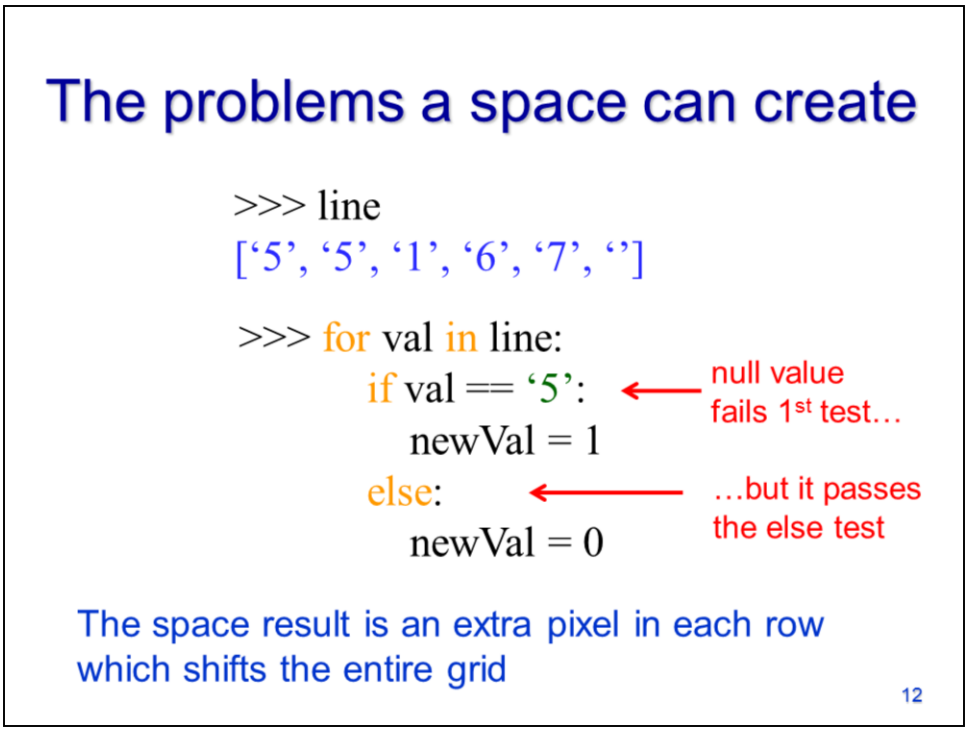
However, in a raster text file, there is a space between the last value and the "\n".

If we remove the "\n" character from the string using the procedure shown in the "Working with files" lecture, then we'll be left with a space after the final value for in the line.



The extra space in the line causes a problem when we split the line to extract the values.

The resulting list will contain a null value at the end.

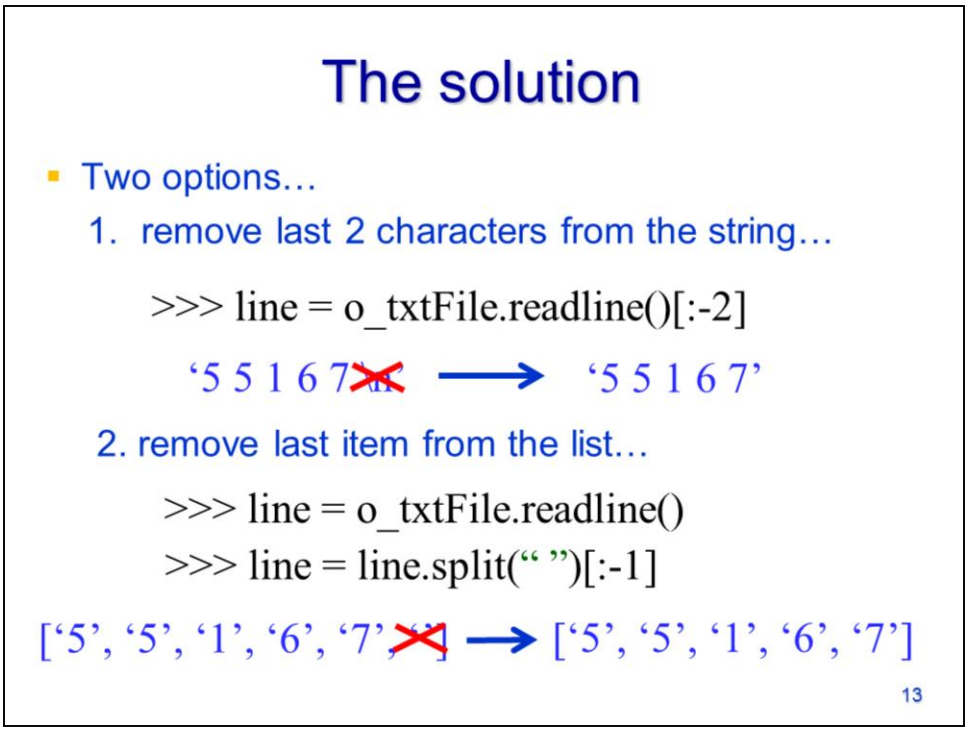


To see what could happen if we don't account for the null value in the list, let's look at a simple example.

In this case, we want to change all the pixels with an original value of 5 to a new value of zero. Pixels with any other values will not be changed.

When the null value at the end of the list is run through the if statement, it will fail the if test…

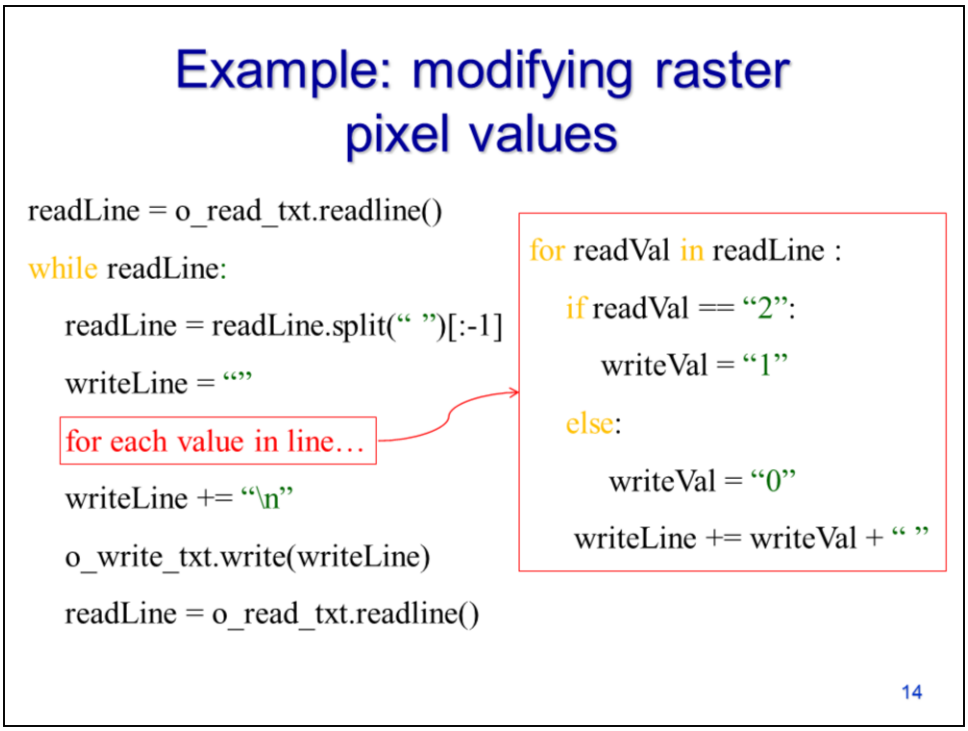
but it will pass the else test. Since the else test passes, the null pixel would be assigned a new value of zero. This would create an extra pixel at the end of each row – which will cause an error when the text file is converted to a raster.



There are two easy solutions to the problem caused by the space between the last value in the line and the "\n" character.

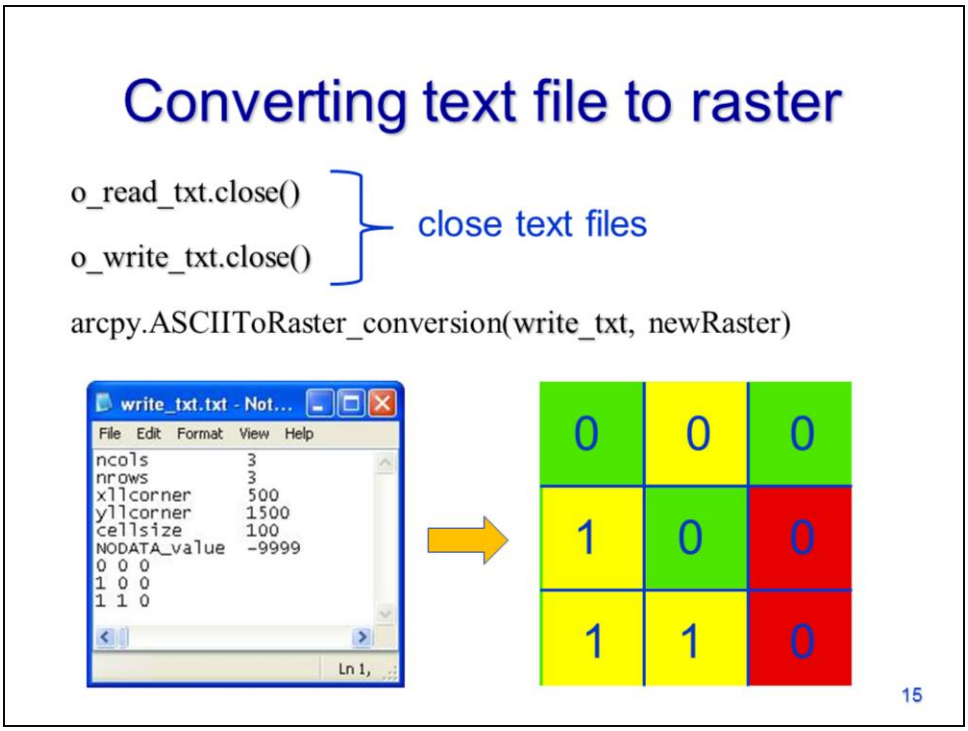
The 1st option is to remove the last two characters from the end of each line in the **readline** statement.

The 2nd option is to remove the last item from the list in the **split** statement.



Here we see an example of the code that can be used to create a new raster text file that is a modified version of the original. Note that prior to these lines of code, are lines that 1) open both the original text file and the new text file and 2) copy the header lines from the original text file to the new text file.

The “for each value in line” section would change depending on the specific objective. In this example, it simply changes all original pixels with a value of 2 to a new value of 1; all other pixels are changed to a new value of 0.



The final steps are to close both the original and the new text file…

And then convert the new text file back to a raster format using arcpy's **ASCIIToRaster** tool.