



Lesson 5: Writing/Reading Text Files, and working with Python Lists

Basic Programming in ArcGIS with Python
Workshop

RS/GIS Lab, Utah State University
With Material from ESRI

Learning Objectives:

- Review working with strings and lists.
- Learn how to read in and write to a text file.
- Learn more about working with lists.
- Learn about Python dictionaries.
- Understand what a dictionary *key* is.
- Understand the difference between *lists* and *dictionaries*.
- To be able to use lists and (possibly) dictionaries in simple scripts.

Lesson 5a: Working with strings

- Strings can be surrounded by double (“ “) or single (‘ ‘) quotes
- Paths are strings

```
gp.workspace = r "C:\john\lesson5"
```

- Strings are indexed
- Strings are zero-based from the left, and 1-based from the right

```
>>> x = "nests1990.shp"
```

```
>>> print x[0]
```

```
n
```

```
>>> print x[0:-4]
```

```
nests1990
```

Manipulating strings

- Must import String module for most string functions

```
import string
```

- Getting help for the split method of the string module

```
help(string.split)
```

```
Help on function split in module string:
```

```
split(s, sep=None, maxsplit=-1)    split(s [,sep [,maxsplit]]) ->  
list of strings
```

```
Return a list of the words in the string s, using sep as  
the delimiter string. If maxsplit is given, splits at no more  
than maxsplit places (resulting in at most maxsplit+1 words).  
If sep is not specified or is None, any whitespace string is a  
separator. (split and splitfields are synonymous)
```

- Example of the split method:

```
>>> # setting the var. files to a string of characters
>>> files = "nests1990.shp, nests1995.shp, nests2000.shp"
>>> print files
nests1990.shp, nests1995.shp, nests2000.shp
>>> files[1]
'e'
```

```
>>> # setting the var. fileList to a list of word objects
>>> fileList = files.split(",")
>>> print fileList
['nests1990.shp', ' nests1995.shp', ' nests2000.shp']
>>> fileList[1]
' nests1995.shp'
```

Other string manipulation methods

- Find () → returns start position of an object in a list, returns -1 if not found

```
if fileList.find("nests2000.shp") > -1:  
    print "nests2000.shp found in list"
```

- Upper () and lower ()

```
string.upper(fileList)  
'NESTS1990.SHP, NESTS1995.SHP, NESTS2000.SHP, NESTS2005.SHP'
```

- Replace ()

```
file1 = "nests1990.shp"  
file1 = file1.replace("1990", "1991")  
file1  
'nests1991.shp'
```

- Lstrip () and rstrip () → what do you think they do?

Writing to text file:

- Why learn it?
 - Log processes steps and track time to run script
 - Create reports
 - Helpful to understand reading text files
- Need to 'Open' the file first. Three modes to open:
 - Append mode ("a") → if not there, will create a file.
 - Write mode ("w") → if not there, will create, if there will overwrite!
 - Read mode ("r") → file must exist, or error message.
- Works for:
 - .txt file
 - .csv file

Writing to text file:

- Useful functions
 - `open()` → creates new file for opens existing file (“a”, “w”, or “r”)
 - `write()` → writes one line in an opened file
 - `writelines()` → writes many lines in an opened file
 - `close()` → releases the file from memory
- Newline character is: `\n`
- Basic 3-step process for writing a text file:
 - 1) Open in write or append mode, and assign to variable
 - 2) Write line-by-line or many lines (paragraph)
 - 3) Close file to release file from memory and save it to disk.

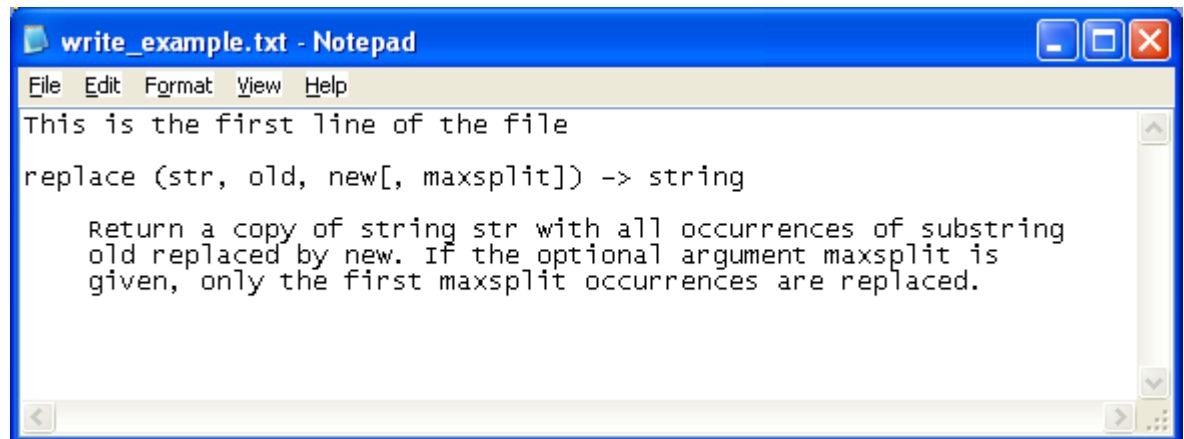
Writing to text file (example code):

```
# Open a new text file in write mode
outFile = open(r"C:\john\Lesson5write_example.txt", "w")

# Write one line to the file, and go to a new line
outFile.write("This is the first line of the file" + "\n")

# Write newline, many lines to the file, and go to a new line
outFile.writelines("\n" + string.replace.__doc__ + "\n")

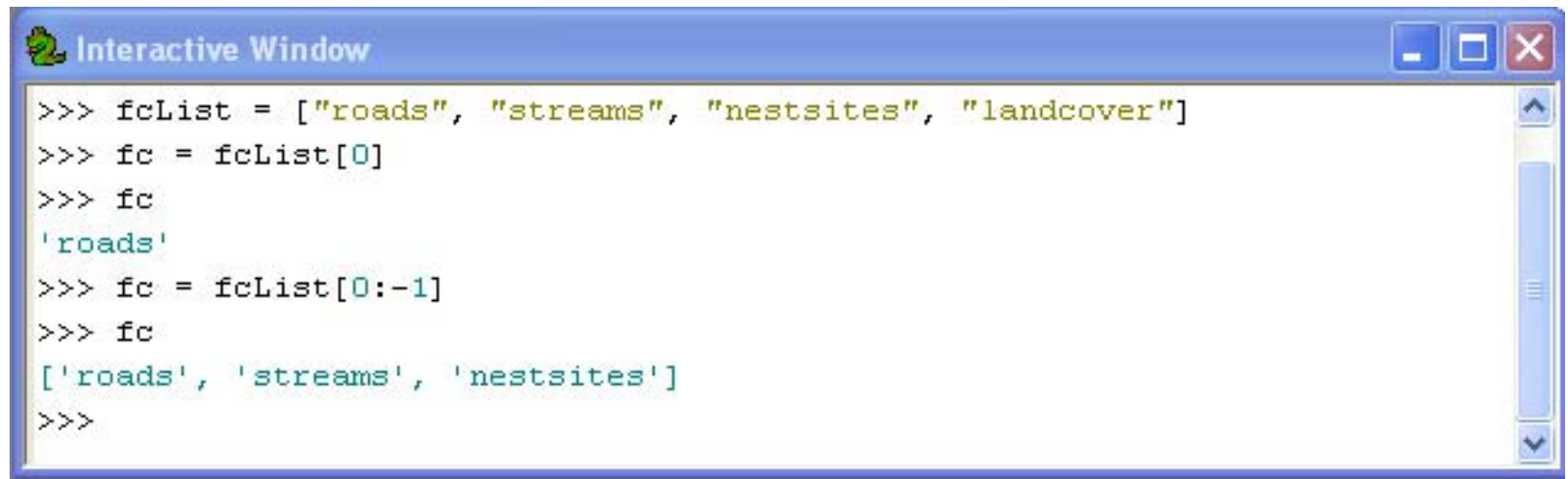
# Close the file
outFile.close()
```



```
write_example.txt - Notepad
File Edit Format View Help
This is the first line of the file
replace (str, old, new[, maxsplit]) -> string
    Return a copy of string str with all occurrences of substring
    old replaced by new. If the optional argument maxsplit is
    given, only the first maxsplit occurrences are replaced.
```

Using Lists in Python:

- Python Lists:
 - Are stored in variables
 - Are indexed and accessed by position
 - Can contain numbers and strings



```

Interactive Window
>>> fcList = ["roads", "streams", "nestsites", "landcover"]
>>> fc = fcList[0]
>>> fc
'roads'
>>> fc = fcList[0:-1]
>>> fc
['roads', 'streams', 'nestsites']
>>>
  
```

Example: Using a list index (with split)

```
# Open the text file in read mode
inFile = open(r"C:\john\Lesson5\nests2005_coords.csv", "r")

# Open a new text file to write to
outFile = open(r"C:\john\Lesson5\nests2005_format.txt", "w")

# Read entire file and print one line at a time
for line in inFile.readlines():
    nestList = line.split(",")
    id = nestList[0]
    cnd = nestList[1]
    x = nestList[2]
    y = nestList[3]
    outFile.write("Siteid: " + id + "\n")
    outFile.write("Condition: " + cnd + "\n")
    outFile.write("X Coordinate: " + x + "\n")
    outFile.write("Y Coordinate: " + y + "\n")
    outFile.write("\n")
```

Assignment 5a: Write a script that writes to a text file

- Complete the lesson5a.py script so that it does the following:
 - Opens a new text file in the write mode.
 - Makes a string variable from a list of characters (shapefiles listed)
 - Converts the string variable to a list variable, splitting on the “,” delimiter.
 - Writes each item in the list to the output file on separate lines.
 - Closes the file.

Reading a text file:

- The file must exist!
- Useful functions
 - `open()` → opens the text file
 - `readline()` → reads a single line
 - `readlines()` → reads in many lines
 - `close()` → releases the file from memory
- Basic 3-step process for writing a text file:
 - 1) Open in read mode and assign to variable (only possible way)
 - 2) Read line-by-line or many lines (paragraph)
 - 3) Close file release file from.

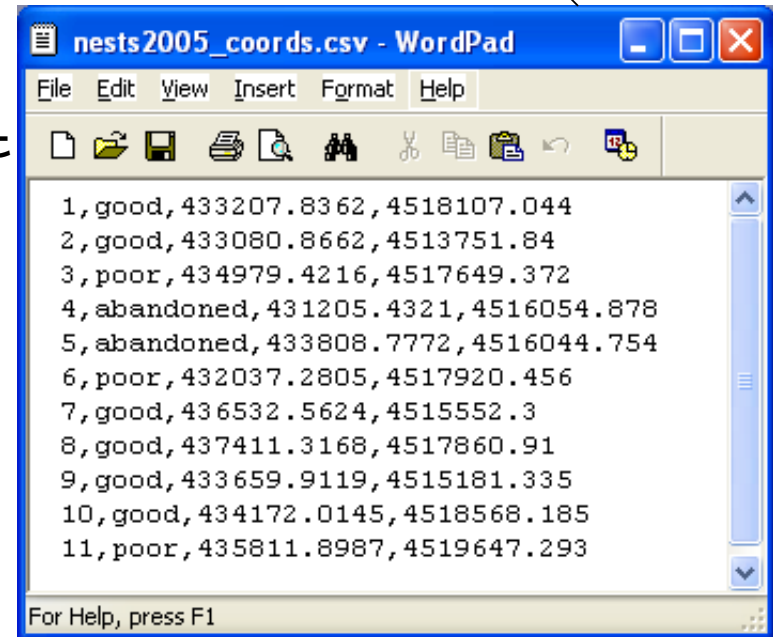
Read text file (example code):

```
# Open the text file in read mode
inFile = open(r"C:\john\Lesson5\nests2005_coords.csv", "r")

# Read entire file and print one line at a time
for line in inFile.readlines():
    print line

# Read one line in at time and print
singleLine = inFile.readline()
while singleLine <> "":
    print singleLine
    singleLine = inFile.readline()

# Close the file
inFile.close()
```



Things you can do with lists:

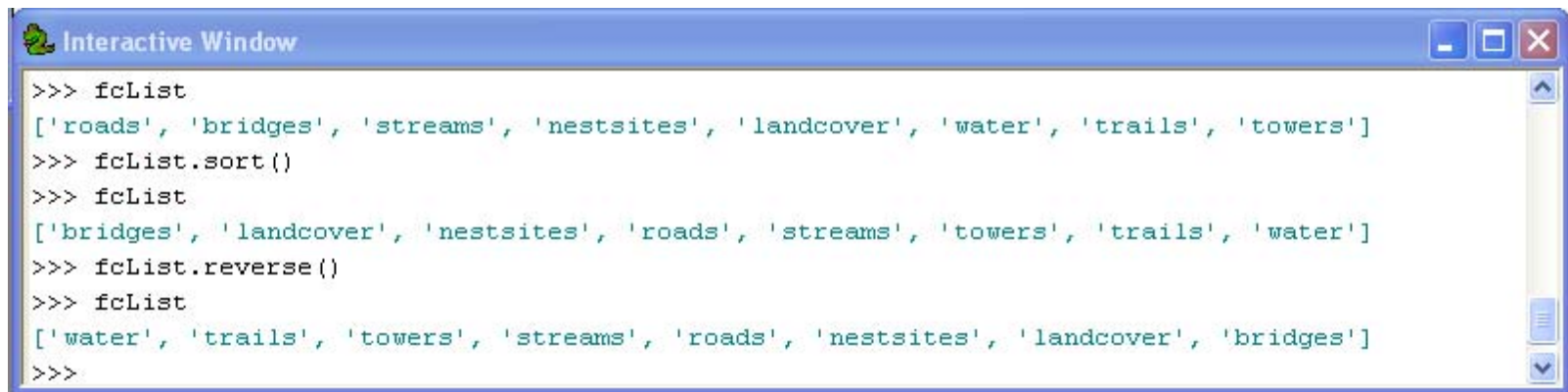
- Append () → add to the list by one object
- Extend () → add to the list with multiple objects
- Insert () → insert into the list at a given position

```

Interactive Window
>>> fcList = ["roads", "streams", "nestsites", "landcover"]
>>> fcList
['roads', 'streams', 'nestsites', 'landcover']
>>> fcList.append("water")
>>> fcList
['roads', 'streams', 'nestsites', 'landcover', 'water']
>>> fcList.extend(["trails", "towers"])
>>> fcList
['roads', 'streams', 'nestsites', 'landcover', 'water', 'trails', 'towers']
>>> fcList.insert(1, "bridges")
>>> fcList
['roads', 'bridges', 'streams', 'nestsites', 'landcover', 'water', 'trails', 'towers']
>>>
  
```

List are ordered:

- Sort () → sorts in sequence from low to high
- Reverse () → reverses the sequence, but does not sort

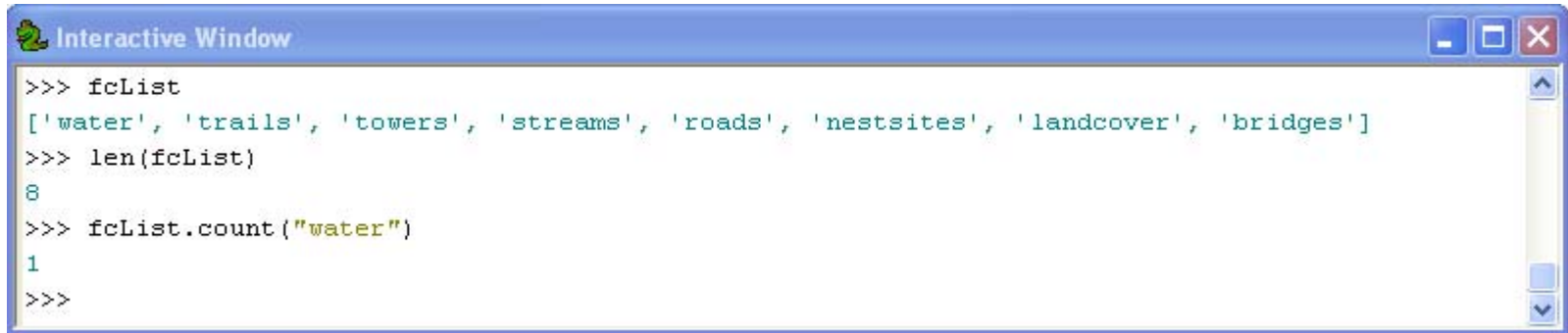


```

Interactive Window
>>> fcList
['roads', 'bridges', 'streams', 'nestsites', 'landcover', 'water', 'trails', 'towers']
>>> fcList.sort()
>>> fcList
['bridges', 'landcover', 'nestsites', 'roads', 'streams', 'towers', 'trails', 'water']
>>> fcList.reverse()
>>> fcList
['water', 'trails', 'towers', 'streams', 'roads', 'nestsites', 'landcover', 'bridges']
>>>
  
```

Lists have length and count:

- Len() returns the total number of items
- Count () returns the number of occurrences of an object



```

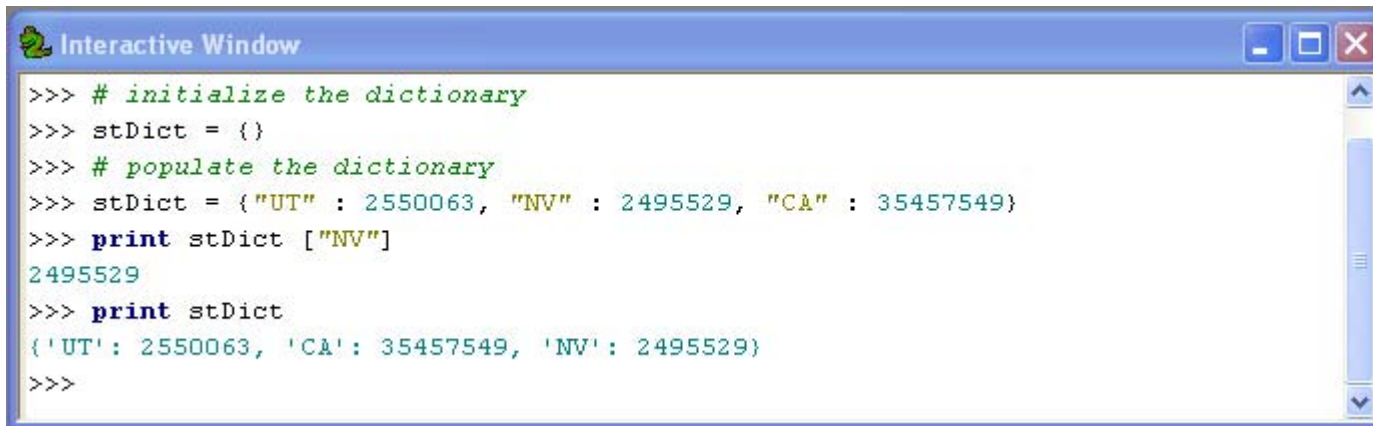
Interactive Window
>>> fcList
['water', 'trails', 'towers', 'streams', 'roads', 'nestsites', 'landcover', 'bridges']
>>> len(fcList)
8
>>> fcList.count("water")
1
>>>
  
```

Assignment 5b: Write a script that reads in a text file and writes to a text file

- Complete the lesson5b.py script so that it does the following:
 - Opens the nests2005_coords.csv file in read mode.
 - Opens a new text file to write to, called “nests2005_format.txt.”
 - Reads the entire inFile.
 - For each line in the inFile, parses it using the split method so that each word (separated by a comma) is assigned to a new variable.
 - Write out the value for each variable to the new textfile, so that it is formatted as depicted below:

Python Dictionaries:

- Unordered collections
- Stored in variables as pairs (key : value)
- Defined by { }
- Keys and values can be strings or numbers
- Access value by key
- Cannot have duplicate keys



```

Interactive Window
>>> # initialize the dictionary
>>> stDict = {}
>>> # populate the dictionary
>>> stDict = {"UT" : 2550063, "NV" : 2495529, "CA" : 35457549}
>>> print stDict ["NV"]
2495529
>>> print stDict
{'UT': 2550063, 'CA': 35457549, 'NV': 2495529}
>>>
  
```

Working with Python Dictionaries:

- Use a List to access the keys
- Check for existence of a key with `has_key()` method.

```

16 # Using has_key method to find out if a key exists
17 keyList = ["UT", "NV", "CA", "ID"]
18 -for inKey in keyList:
19 -    if stDict.has_key(inKey):
20         print inKey + " already exists"
21 -    else:
22 -        print inKey + " does not exist in the dictionary"

```



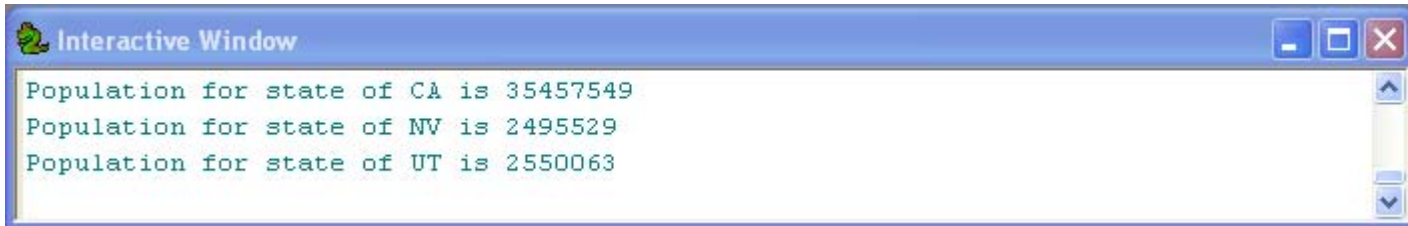
Working with Python Dictionaries:

- Use `keys ()` method to find all the keys in the Dict.
- Assign output from `keys ()` to a List, then loop thru List

```

24 # store the keys in the list
25 keyList = stDict.keys()
26 # sort the list (note the dict can't be sorted)
27 keyList.sort()
28 # loop through list, but access key & value from dict.
29 -for key in keyList:
30     print "Population for state of " + key + " is " + str(stDict[key])

```



```

Interactive Window
Population for state of CA is 35457549
Population for state of NV is 2495529
Population for state of UT is 2550063

```

Comparing Lists and Dictionaries:

	Lists	Dictionaries
Ordered Collection?	Yes	No
Access by?	Index (e.g. <code>fcList[0]</code>)	Key (e.g. <code>stDict["UT"]</code>)
How to determine number of items?	<code>Len()</code>	<code>Len ()</code>
How to initialize?	<code>myList = []</code>	<code>myDict = { }</code>
How to add data to?	<code>append, insert, extend</code>	<code>myDict = {key: value}</code>

Assignment 5c Challenge:

Assume that each of the points in the nests2005_coords.csv file is the center point of a square-shaped plot that is 20 meters by 20 meters. Write a script that takes the coordinate pairs from nests2005_coords.csv and calculates the ll, ul, ur, lr, coordinate pairs based on the center coordinate pair and outputs the 4 new coordinate pairs for every center coordinate pair to a new text file.

