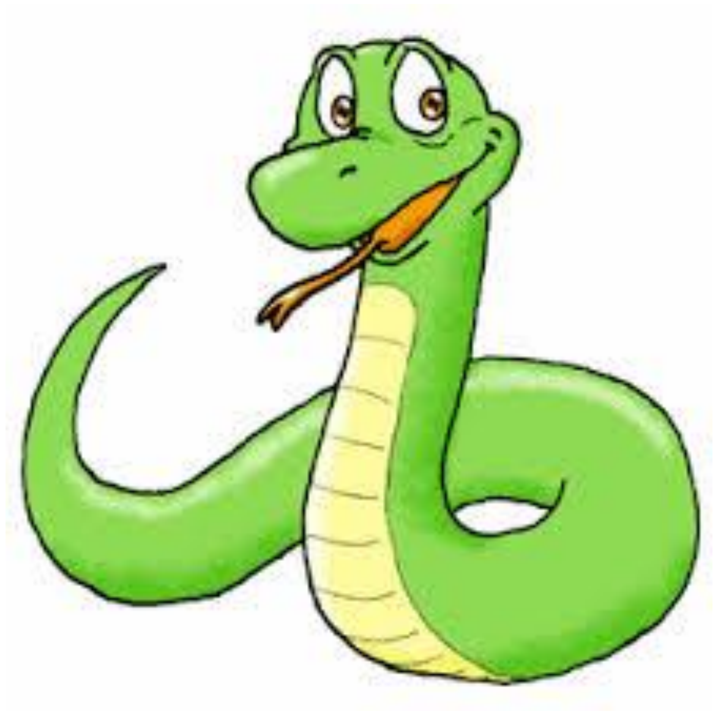
Application of GIS with Python

Chapter 7: Files



https://www.python.org/
http://www.tutorialspoint.com/python/

Table of Content

Files

- There are several ways to present the output of a program; data can be printed in a human-readable form, or written to a file for future use.
- One of the simplest ways for programs to maintain their data is by reading and writing text files.
- A text file is a sequence of characters stored on a permanent medium like a hard drive, flash memory, or CD-ROM.

- Files are organized into **directories** (also called “folders”).
- Every running program has a “current directory,” ;the default directory for most operations.
- For example, when open a file for reading, Python looks for it in the current directory

- The OS module provides functions for working with files and directories (“os” stands for “operating system”).
- os.getcwd returns the name of the current directory:

```
>>> import os
>>> cwd = os.getcwd()
>>> print cwd
```

cwd stands for “current working directory.”

A string like cwd that identifies a file is called a **path**.

- A **relative path** starts from the current directory
- An **absolute path** starts from the topmost directory in the file system

The paths with simple filenames are relative to the current directory.

- os.path.exists checks whether a file or directory exists:
- If it exists, os.path.isdir checks whether it’s a directory:
- Similarly, os.path.isfile checks whether it’s a file.
- os.listdir returns a list of the files (and other directories) in the given directory:

```
>>> os.path.exists('E:\\WRC\\2015winter\\Gis With Python')
True

  >>> os.path.isdir('E:\\WRC\\2015winter\\Gis With Python')
  True

  >>> os.path.isdir('music.txt')
  False


  >>> os.listdir(cwd)
  ['DLLs', 'Doc', 'include',…….']
```

Opening and Closing files

- The file in plain text can be opened with a text editor, but can also read it from Python.
- The built-in function open() take the name of the file as a parameter and returns a **file object,** that creates a new file or open an existing file

```
>>> fin = open('words.txt’)
 >>> print fin
<open file 'words.txt', mode 'r' at 0xb7f4b380>
```

- fin is a common name for a file object used for input. Mode 'r' indicates that this file is open for reading (as opposed to 'w' for writing). **object** that can use to read the file.

- When done with file need to close the file.
- **The with Statement**: The **with** statement is used to be absolutely sure that we have closed a file (or other resource) when we're done using it.

```
>>>with fin as source:
        for line in source:
            print line
```

>>> fin.close() #method that close the file object, flushing all data. The closed flag is set.

```
>>> fin.closed   #checks whether file closed
True
```

Opening and Closing files

- **File Mode Strings**. The mode string specifies how the file will be accessed by the program. There are three separate issues addressed by the mode string**: opening, text handling and operations**.
- **Opening**. For the opening part of the mode string, there are three alternatives:
- **'r'** Open for reading. Start at the beginning of the file. If the file does not exist, raise an IOError exception. This is implied if nothing else is specified.
- **'w'** Open for writing. Start at the beginning of the file, overwriting an existing file. If the file does not exist, create the file.
- **'a'** Open for appending. Start at the end of the file. If the file does not exist, create the file.

- **Text Handling**. For the text handling part of the mode string, there are two alternatives:
  - **‘b’** Interpret the file as bytes, not text.
  - **(nothing)** The default, if nothing is specified is to interpret the content as text: a sequence of characters with newlines at the end of each line.
- **Operations**. For the additional operations part of the mode string, there are two alternatives:
  - **‘+’** Allow both read and write operations.
  - **(nothing)** If nothing is specified, allow only reads for files opened with “r”; allow only writes for files opened with “w” or “a”.

Reading and Writing files

- Normally, files are opened in text mode, that means, read and write strings from and to the file, which are encoded in a specific encoding (character or byte sequence)
- The resulting/created/ opened file object has a number of operations that change the state of the file, read or write data, or return information about the file

Reading and Writing files

Read Methods:

- The following methods read from a file.
- As data is read, the file position is advanced from the beginning to the end of the file.
- The file must be opened with a mode that includes or implies 'r' for these methods to work.

read(*[size]*)
Read as many as size characters or bytes from the file. If size is negative or omitted, the rest of the file is read.
readline(*[size]*)
Read the next line. If size is negative or omitted, the next complete line is read. If the size is given and positive, read as many as size characters from the file; an incomplete line can be read. If a complete line is read, it includes the trailing newline character, \n. If the file is at the end, this will return a zero length string.
readlines(*[hint]*)
Read the next lines or as many lines from the next hint characters from file. The value of hint may be rounded up to match an internal buffer size. If hint is negative or omitted, the rest of the file is read. All lines will include the trailing newline character, \n. If the file is at the end, this returns a zero length list.

- **Write Methods**. The following methods write to a file. As data is written, the file position is advanced, possibly growing the file.
- If the file is opened for write, the position begins at the beginning of the file.
- If the file is opened for append, the position begins at the end of the file.
- If the file does not already exist, both writing and appending are equivalent.
- The file must be opened with a mode that includes 'a' or 'w‘ for these methods to work.

write(*string*)
Write the given string to the file.
writelines(*list*)
Write the list of strings to the file. Buffering may mean that the strings do not appear on any console until a close() or flush() operation is used.
truncate(*[size]*)
Truncate the file. If size is not given, the file is truncated at the current position. If size is given, the
file will be truncated at size.

Copying files

The **shutil** module offers a number of high-level operations on files and collections of files. In particular, functions are provided which support file copying and removal.

shutil.copyfile(src, dst)
Copy the contents (no metadata) of the file named src to a file named dst and return dst. src and dst are path names given as strings. dst must be the complete target file name. If src and dst specify the same file, SameFileError is raised.

shutil.copy(src, dst)
Copies the file src to the file or directory dst. src and dst should be strings. If dst specifies a directory, the file will be copied into dst using the base filename from src. Returns the path to the newly created file.

shutil.copy2(src, dst)
Identical to copy() except that copy2() also attempts to preserve all file metadata.

```
>>> import shutil

>>> help(shutil.copyfile)

>>> shutil.copyfile('E:\\WRC\\2015winter\\work.txt', 'E:\\WRC\\2015winter\\workcopy.txt‘)

>>> shutil.copy('E:\\WRC\\2015winter\\work.txt', 'E:\\WRC\\2015winter\\workcopy.txt‘)

>>> shutil.copy2('E:\\WRC\\2015winter\\work.txt', 'E:\\WRC\\2015winter\\workcopy.txt‘)
```

#returns path to the newly created file.