TATA ELXSI

Python Scripting

Learning & Development Team

Monday, November 2, 2020

Module – 8 : I/O Files

Disclaimer

- This material is developed for in house training at **TATA ELXSI**.
- *TATA ELXSI is not responsible for any errors or omissions, or for the results obtained from the use of this information. All information in this material is provided "as is", with no guarantee of completeness, timeliness or of the results obtained from the use of this information...*
- *This is a training material and the information contained herein is not intended to be a source of advice or research outcome with respect to the material presented, and the information and/or documents/examples contained in this material do not constitute authors or TATA ELXSI's advice.*
- The training material therefore was solely developed for educational purposes. Currently, commercialization of the prototype does not exist, nor is the prototype available for general usage.
- **TATA ELXSI**, its staff, its students or any other participants can use it as a part of training. This material should not be found on any website.
- For the preparation of this material we have referred from the below mentioned links or books.
- Excerpts of these material has been taken where there is no copy right infringements.

Agenda

- Open – close the files
- Reading files
- Writing to the files
- Usage of pickle
- Storing and parsing packed binary data in files

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

- You can open and use files for reading or writing by creating an object of the file class.

- use read, readline or write methods appropriately to read from or write to the file.

- The ability to read or write to the file depends on the mode you have specified for the file opening.

- The close method to tell Python that we are done using the file.

Opening and Closing a File in Python

- When you want to work with a file, the first thing to do is to open it.
- This is done by invoking the open() built-in function.

- Syantax:
- open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)

- file = open(‘details.txt')

Opening and Closing a File in Python

- "a": Appends all output to the end of the file; does not overwrite information currently present. If the indicated file does not exist, it is created.

- "r": Opens a file for input (reading). If the file does not exist, an IOError exception is raised.
- "r+": Opens a file for input and output. If the file does not exist, causes an IOError exception.

- "w": Opens a file for output (writing). If the file exists, it is overwritten. If the file does not exist, one is created.
- "w+": Opens a file for input and output. If the file exists, it is overwritten; otherwise one is created.

- "ab", "rb", "r+b", "wb", "w+b": Opens a file for binary (i.e.,non-text) input or output. [Note: These modes are supported only on the Windows and Macintosh platforms. Unix-like systems don’t care about the data type.]

close

- Warning: You should always make sure that an open file is properly closed.

- In most cases, upon termination of an application or script, a file will be closed eventually.
- However, there is no guarantee when exactly that will happen.
- This can lead to unwanted behavior including resource leaks. It’s also a best practice within Python (Pythonic) to make sure that your code behaves in a way that is well defined and reduces any unwanted behavior.

- reader = open('details.txt')
- try:
-     # Further file processing goes here
- finally:
-     reader.close()

Open-close

- The second way to close a file is to use the with statement:
- >>> with open('workfile') as f:
- ...   read_data = f.read()

- >>> # We can check that the file has been automatically closed.
- >>> f.closed
- True

- The ‘with’ statement automatically takes care of closing the file once it leaves the with block, even in cases of error.
- Even if an exception is raised at some point. Using with is also much shorter than writing equivalent try-finally blocks:

Open-close

- If you're not using the with keyword, then you should call f.close() to close the file and immediately free up any system resources used by it.

- If you don't explicitly close a file, Python's garbage collector will eventually destroy the object and close the open file for you, but the file may stay open for a while.

- Another risk is that different Python implementations will do this clean-up at different times.

reading

- >>> with open('poem.txt', 'r') as reader:
- >>>   # Read & print the entire file
- >>>   print(reader.read())

- To read a file’s contents, call f.read(size), which reads some quantity of data and returns it as a string (in text mode) or bytes object (in binary mode).
- size is an optional numeric argument.

- When size is omitted or negative, the entire contents of the file will be read and returned;

- Otherwise, at most size characters (in text mode) or size bytes (in binary mode) are read and returned.

- If the end of the file has been reached, f.read() will return an empty string ('').

readline

- readline() reads a single line from the file; a newline character (\n) is left at the end of the string, and is only omitted on the last line of the file if the file doesn’t end in a newline.
- readline() returns an empty string, the end of the file has been reached, while a blank line is represented by '\n', a string containing only a single newline.

- Here’s an example of how to read 5 bytes of a line each time using the Python .readline() method:
- >>> with open('poem.txt', 'r') as reader: # Read & print the first 5 characters of the line 5 times
- >>>   print(reader.readline(5))
- >>>   # Notice that line is greater than the 5 chars and continues
- >>>   # down the line, reading 5 chars each time until the end of the
- >>>   # line and then "wraps" around
- >>>   print(reader.readline(5))
- >>>   print(reader.readline(5))

reading

- Here’s an example of how to read the entire file as a list using the Python .readlines() method:

- >>> f = open('poem.txt')
- >>> f.readlines() # Returns a list object

Write function

- f.write(string) writes the contents of string to the file, returning the number of characters written.

- >>> f.write('This is a test\n')
- 15
- Other types of objects need to be converted – either to a string (in text mode) or a bytes object (in binary mode) – before writing them:

- >>>
- >>> value = ('the answer', 42)
- >>> s = str(value) # convert the tuple to string
- >>> f.write(s)
- 18

Tell() and seek()

- f.tell() returns an integer giving the file object’s current position in the file represented as number of bytes from the beginning of the file when in binary mode and an opaque number when in text mode.

- To change the file object’s position, use f.seek(offset, whence).

- The position is computed from adding offset to a reference point; the reference point is selected by the whence argument.

- A whence value of
  - 0 measures from the beginning of the file,
  - 1 uses the current file position, and
  - 2 uses the end of the file as the reference point.

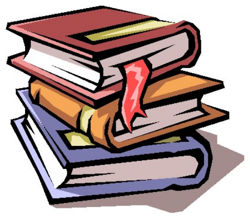
seek

- whence can be omitted and defaults to 0, using the beginning of the file as the reference point.
- >>> f = open('workfile', 'rb+')
- >>> f.write(b'0123456789abcdef')
- 16
- >>> f.seek(5)   # Go to the 6th byte in the file
- 5
- >>> f.read(1)
- b'5'
- >>> f.seek(-3, 2) # Go to the 3rd byte before the end
- 13
- >>> f.read(1)
- b'd'

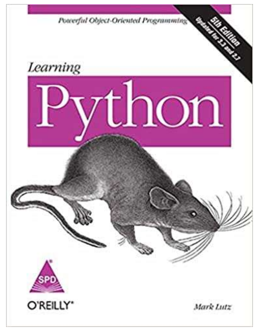
NOTE

- In text files (those opened without a b in the mode string), only seeks relative to the beginning of the file are allowed.
- Exception being seeking to the very file end with seek(0, 2)) and the only valid offset values are those returned from the f.tell(), or zero.
- Any other offset value produces undefined behavior.

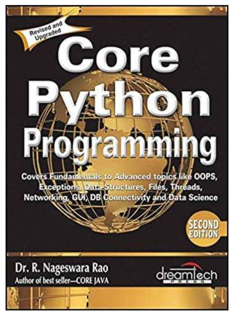
References



- **Python 3.x.x documentation**: https://docs.python.org/3/



- **Learning Python: Powerful Object-Oriented Programming: 5th Edition**



- **Core Python Programming [**by R. Nageswara Rao (Author)]

TATA ELXSI

Thank you

Learning & Development

Tata Elxsi

Bangalore

www.tataelxsi.com

TATA ELXSI