NED University of Engineering and Technology
Department of Computer and Information Systems Engineering

Fall Semester 2020

FE Batch 2020

CS-115 Computer Programming

Online Lecture 11 (Week 10)

Files

Mr. Kashif Asrar

Files

- ✓ A file is a sequence of bytes stored on a secondary memory device.
- ✓ Files are of various types:
  - ❖ Text Files
  - ❖ Spreadsheets
  - ❖ Binary Files
  - ❖ Executable Files
- ✓ Files are managed by File system that operating system supports.
- ✓ Processing a file is based on three steps:
  - ❖ Operating a file for reading or writing
  - ❖ Reading from or writing to the file
  - ❖ Closing the file

Files

- ✓ Python Standard library includes a module names 'io' which contains class for handling files called 'TextIoWrapper'.

```
>>> io.TextIOWrapper
<class '_io.TextIOWrapper'>
```

- ✓ Since filing is a very common activity , no import is required to access these functions.

Opening a File

- ✓ The function open() is used to open the files (text or binary).
- ✓ This function is defined in built-in modules.
- ✓ The function takes:
  - ❖ A file name (with or without path)
  - ❖ '\' is used for path but since it may coincide with escape sequence so python accepts '/' (forward slash).
  - ❖ A path could be absolute or relative:
    - ▪ Raw / absolute path starts from root directory:
      e.g.: c:\office\classes\CP\Text.txt
    - ▪ Relative path starts the sequence from current directory:
      e.g: CP\Text.txt

Opening a File

- The function takes:
  - A file name (with or without path).
  - Mode specifies how to interact with opened file.
    - r=>reading mode (default)
    - w=>writing mode, if the file already exists otherwise its content is wiped out
    - a=>append mode, the data will append to the end of file.
    - t=>text mode (default)
    - b=>binary mode

Examples

```
>>> f=open('myfile.txt') ≡ f=open('myfile.txt', 'r')
```

- Opens myfile.txt if it exist ; returns an object of io.TextIOWrapper type simply called File Type.
- Generates error if the file does not exist.
- The file is opened for reading only.

```
>>> f=open('myfile.txt', 'w')
```

- Opens myfile.txt for writing.
- Creates a new file if it doesn’t exist.
- Overwrites the existing file.

Examples

```
>>> f=open('myfile.txt', 'a')
```

- Opens myfile.txt for writing.
- Creates a new file if it doesn’t exist.
- Appends at the end of existing file.

Closing a File

- ✓ A file must be closed after use.
- ✓ Closing a file releases the file system resources that keeps track of information about the opened file.
- ✓ Try deleting an opened file without closing it.
- ✓ close() closes the file.

```
>>>f.close()
```

with/as statement

- ✓ Automatically closes a file after block of code is executed.
- ✓ Syntax:

```
with open <file name> as f:
        <block>
```

- ✓ Opens <file name> and assigns it handler.
- ✓ Closes f after <block> is executed.

Reading a File

- ✓ `f.read(n)`

  reads and returns as string 'n' characters from file 'f' or until the end of file is reached.

- ✓ `f.read()`

  reads and returns as string characters from file f until the end of file

Reading a File

- ✓ `f.readline()`
  reads and returns as string characters from file f until (including) new line character or end of file.

- ✓ `f.readlines()`
  reads and returns as list.

Reading a File

- With every opened file, the system will associate a cursor that points to the character in the file.
- When the file is first opened, the cursor typically points to the start of the file.
- Using different types of read operations consecutively, second read commences from where first read ended

Example-1

Store the following file as myfile.text

myfile.txt - Notepad

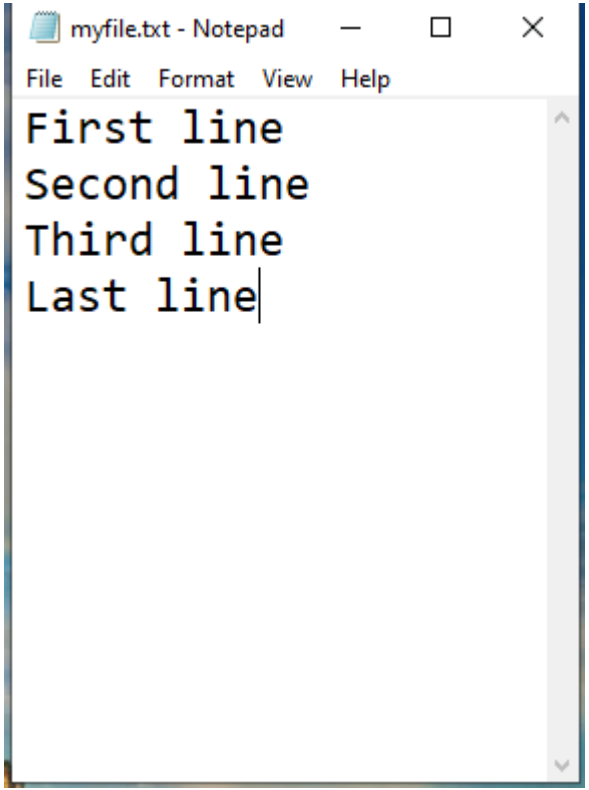
File Edit Format View Help

```
First line
Second line
Third line
Last line
```

```
>>> f=open('myfile.txt')
>>> f.read()
'First line\nSecond line\nThird line\nLast line'
>>> f.read()
''
>>> f.close()
>>> f=open('myfile.txt')
>>> print(f.read())
First line
Second line
Third line
Last line
>>> f.close()
```

Example-2

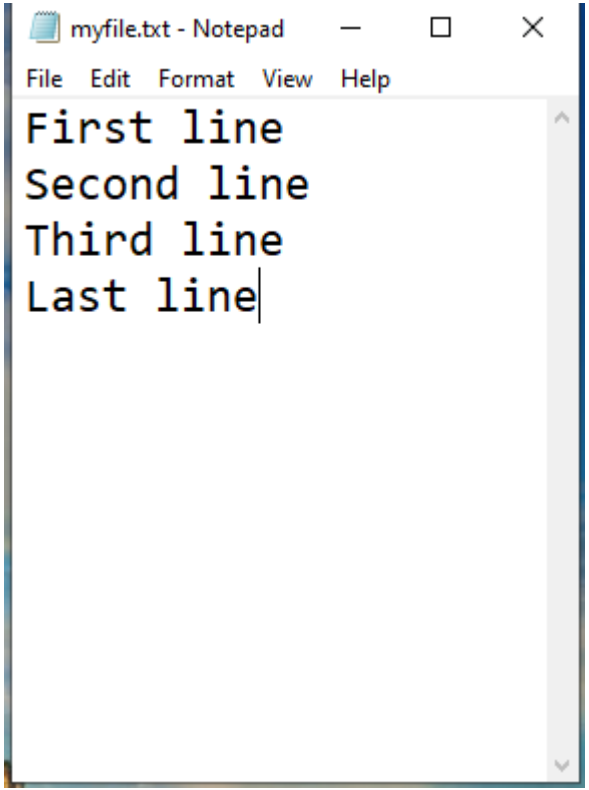
Reading the files 10 characters at a time




```
>>> f=open('myfile.txt')
>>> f.read(10)
'First line'
>>> f.read(10)
'\nSecond li'
>>> f.read(10)
'ne\nThird l'
>>> f.close()
```

Example-3

Reading one line at a time




```
>>> f=open('myfile.txt')
>>> f.readline()
'First line\n'
>>> f.readline()
'Second line\n'
>>> f.readline()
'Third line\n'
>>> f.close()
```

Example-4

Reading all lines

myfile.txt - Notepad

File Edit Format View Help

```
First line
Second line
Third line
Last line
```

```
>>> f=open('myfile.txt')
>>> f.readlines()
['First line\n', 'Second line\n', 'Third line\n', 'Last line']
>>> f=open('myfile.txt')
>>> p=f.readlines()
>>> print(p[1])
Second line

>>> f.close()
```

Other Useful Functions on Files

- ✓ f.name

  Contains name of file. It's an attribute, not a method.
- ✓ f.seek(offset, from_what)
  - ❖ Changes file object position.
  - ❖ Position is computed from adding offset to a reference point
  - ❖ Reference point is selected by from_what argument.
    - ▪ From_what=0, offset measured from start of file.
    - ▪ From_what=1, offset measured from current position.
    - ▪ From_what=2, offset measured from EoF
    - ▪ Default value is 0
- ✓ f.tell(): returns an integer giving file objects current position in the file as number of bytes from the beginning of the file.

Example-5

```
>>> f=open('myfile.txt')
>>> f.read()
'First line\nSecond line\nThird line\nLast line'
>>> f.read()
''
>>> f.seek(0)
0
>>> f.read()
'First line\nSecond line\nThird line\nLast line'
>>> f.seek(3)
3
>>> f.read(5)
'st li'
>>> f.tell()
8
```

Practice Problem

Read file "myfile.txt" created in examples and print the following information about it.

- Name of the file
- Total characters in the file
- Total words in file
- Total lines in the file.

Solution to Practice Problem

```
f=open('myfile.txt')
print('Name of the file:', f.name)
content=f.read()
print('Number of characters:', len(content))
l=content.split()
print('Number of words:', len(l))
f.seek(0)
lines=f.readlines()
print('Number of lines:', len(lines))
f.close()
```

```
Name of the file: myfile.txt
Number of characters: 43
Number of words: 8
Number of lines: 4
```

Practice Problem-2

Find the number of alphabets in ”myfile.txt”

```
f=open('myfile.txt')
content=f.read()
count=0
for i in content:
    if i.isalpha():
        count+=1
print('total alphabets=', count)
f.close()
```

IDLE

```
total alphabets= 36
```

Practice Problem-3

Print large files line wise in order to avoid large memory consumption simultaneously.

```
with open('myfile.txt') as f:
    for i in f:
        print(i)
```

Writing a File

- Write <text string> writes string to the file.
- Write starts from cursor position.
- It returns the number of bytes/characters written.

Example

```
>>> f=open('newfile.txt','w')
>>> f.write('this is a new file')
18
>>> f.write('Adding another line')
19
>>> f.close()
>>> f=open('newfile.txt')
>>> f.read()
'this is a new fileAdding another line'
```

Read / Write modes: r+, w+, a+

- All these modes allow read and write.
- r+ can not create a file, sets cursor at start.
- w+ always overwrites a file. Sets cursor at start, Opens a file in write mode and erases all previous content.
- a+ can create a file and sets cursor at EoF.

Mode	Create File	Cursor	Overwrite File
r, r+	No	Start	No
w, w+	Yes	Start	Yes
a, a+	Yes	End	No

Example Using r+

Create a file having content 'This is another file'.

```
>>> f=open('f.txt','r+')
>>> f.read()
'This is another line'
>>> f.write('\n Adding another line')
21
>>> f.read()
''
>>> f.flush()
>>> f.read()
''
>>> f.seek(0)
0
>>> f.write('overwriting')
11
>>> f.seek(0)
0
>>> f.read()
'overwritingther line\n Adding another line'
>>> f.close()
```

Thank you!