

## CSI31 Lecture 10

**Topics:** *Chapter 4. Computing with strings*  
4.6 File processing

**HW# 10** (due date: Wednesday, November 5th):

page 119/6

note: the name is stored in a file.

The user will input the file name when the program is run (i.e. your program should ask the user to input the file name)

## 4.6 File processing

We can *create, store, modify, erase* word documents (files).

Conceptually, a **file** is a sequence of data that is stored in secondary memory (hard drive, CDs, DVDs,...).

Files can contain any data type. In Python, text files can be very flexible, since it is easy to convert back and forth between strings and other data types (recall **int**, **float**, **long**, **str**, **eval** functions).

How to mark the end-of-the-line?

'\n' - single character *newline*

Another special character: '\t' for <Tab> (*tabulation*)

type the following in the interactive window:

```
>>> print 'Hello! \n It is a nice weather outside, isn't it? \n Good buy! \n'
>>> 'Hello! \n It is a nice weather outside, isn't it? \n Good buy! \n'
```

## File manipulation concepts:

- associate a certain file on a disk with a variable in the program ([opening a file](#))
  - manipulate the file variable ([read](#), [write](#),...)
  - release the file ([closing the file](#))
- (like we are using a word-processing application)

## Opening a file

open function:

```
<filevar> = open(<name>, <mode>)
```

**mode:** "r" – for reading from the file  
"w" – for writing to the file  
"a" – for appending

### examples:

```
infile = open('input-data.dat', 'r')
```

- opened file input-data.dat for reading

```
outfile = open('result.dat', 'w')
```

- opened file result.dat for writing
- if a file result.dat existed before, its old contents will be destroyed when we begin writing into it (if you'd like to add data to an existing file, use "a" mode).

## Reading from a file

`<filevar>.read()` - returns the entire *remaining contents* of the file as a single (potentially large, multi-line) string

`<filevar>.readline()` - returns the next line of the file  
(*all text up to and including the next newline character*)

`<filevar>.readlines()` - returns a list of the remaining lines in the file  
(each list item is a single line including the newline character at the end)

## Writing to a file

`<filevar>.write(<string>)` - writes the string to a file

! if a file that we opened for writing existed before, its old contents will be destroyed when we begin writing into it.

! when we write a string into a file, end-of-line is not added automatically.

`<filevar>.write(<string> + '\n')` adds end-of-line

## Closing a file

`<filevar>.close()` - closes the file

---

Type in the following in the interactive window:

```
>>> out=open('result.dat','w')
>>> out.write('Hello!')
```

Now, try to find the file **result.dat**, and see what's in it. Then type in the following:

```
>>> out.close()
```

Check what's in the file **result.dat** now. Then:

```
>>> out_next=open('result.dat','w')
>>> out_next.write('Good buy!')
>>> out_next.close()
```

See what's in the file **result.dat** now. Then:

```
>>> out=open('result.dat','a')
>>> out.write('\n And hello again!')
>>> out.close()
```

See what's in the file **result.dat** now.

Now. Let's try to write a program (with few modifications):

Write a program that opens a file numbers.dat, reads all the information from it, and displays it on the screen

see [read-from-file.py](#)  
[read-from-file2.py](#)  
[read-from-file\\_lists.py](#)

1<sup>st</sup> modification of program:

write the program so that the user inputs the file name

see [read-from-file-mod1.py](#)

2<sup>nd</sup> modification of the program:

your program should add up the numbers after displaying them on the screen.

don't forget to display the sum

see [read-from-file-mod2.py](#)

3<sup>rd</sup> another modification:

file may contain characters (not only numbers)

your program should add up only numbers

(don't forget to display the contents of the file on the screen)

see [read-from-file-mod3.py](#)

4<sup>th</sup> modification:

take care of the case when the user inputs a wrong file name

(the name of a file, that doesn't exist).

The program shouldn't crash. It should stop and inform the user about the problem.

see [read-from-file-mod4.py](#)