File Processing in Python

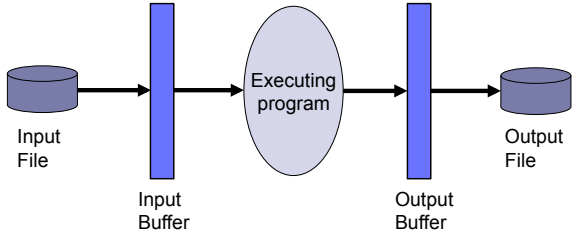
A **file** is a collection of data stored on a secondary storage device, e.g., a disk drive or thumb drive

The operating system manages the file system,

Python program interacts with OS through:

- ***file objects***, which represent files
- builtin functions, e.g., `open()`, `print()`, ...
- methods of a file object,e.g., `read()`, `write()`, `close()`, …

Streams and Buffers



Each file object has a *buffer* (large string) which is the interface between the program and the file. The OS copies large blocks to and from buffers; Python I/O operations copy smaller items.

To open a text file in Python

```
my_file = open( "datafile.txt", "w" )
```

To open a text file in Python

creates and returns a file object containing a buffer, connected to the named file, which is open for input or output

```
my_file = open( "datafile.txt", "w" )
```

variable

name of file (on disk)

access mode (read or write)

- `"r"` (read, the default)
  - file exists ==> open operation succeeds
  - file doesn't exist ==> open operation fails (raises exception)
- `"w"` (write)
  - file exists ==> succeeds; file contents deleted
  - file doesn't exist ==> succeeds; file created

To close a text file in Python

```
my_file.close()
```


To close a text file in Python

writes the contents of the buffer to disk and breaks the connection between the disk file and the file object.

```
my_file.close()
```

reference to a file object

Reading from a file

Most common Python approach:

```
my_file = open("data.txt", "r")
# process one line at a time
for line_str in my_file:
    suite    # process this line
my_file.close()
```

- Treats each line of the file as a string; and the file as a collection of line strings.
- Assigns `line_str` the string of characters in consecutive lines of the file, executing `suite` for each.

Reading from a file

Alternative – use readline():

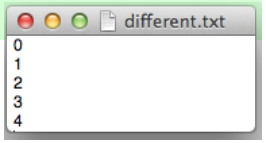
```
my_file = open("data.txt", "r")
line_str = my_file.readline()
while len( line_str ) > 0:
    . . .     # process this line
    line_str = my_file.readline()
my_file.close()
```

- first time: reads up through the first newline (`'\n'`)
- subsequent times: read from where the last call left off, through the next newline
- returns the string read or, if at end of file, the empty string

To write into a text file in Python:

Most common: Use **`print()`** function, but assign the file object to named parameter **`file`**

```
file_obj = open( "different.txt", "w" )
for num in range(5):
    print( num, file=file_obj )
file_obj.close()
```



To write into a text file in Python:

Alternative: Use **`write()`** method, giving it a string to write

```
file_obj = open( "different.txt", "w" )
for num in range(5):
    file_obj.write( str(num) + "\n" )
file_obj.close()
```