NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 5, Lecture 3

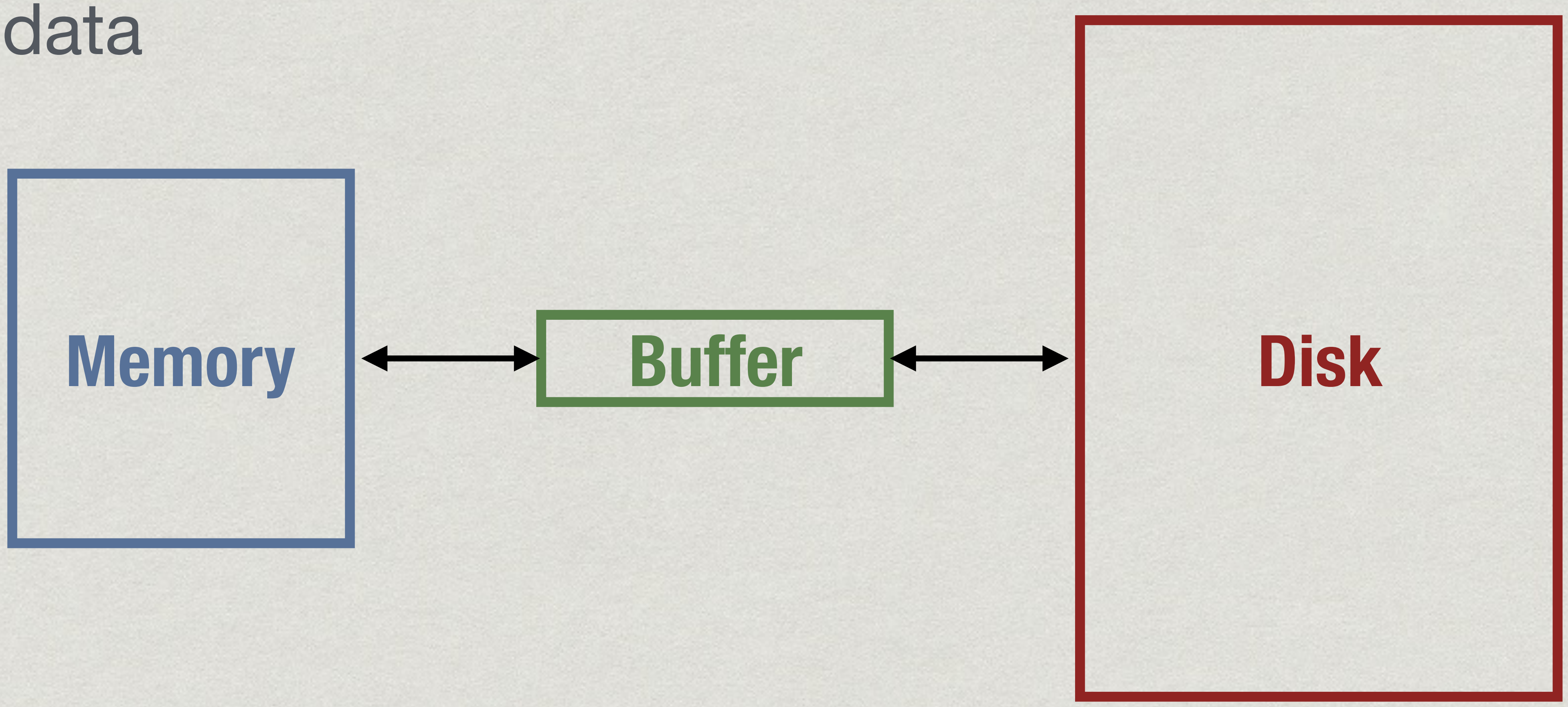
Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Dealing with files

- Standard input and output is not convenient for large volumes of data
- Instead, read and write files on the disk
- Disk read/write is much slower than memory

Disk buffers

- Disk data is read/written in large blocks
- “Buffer” is a temporary parking place for disk data



Reading/writing disk data

Reading/writing disk data

- Open a file — create file handle to file on disk
  - Like setting up a buffer for the file

Reading/writing disk data

- Open a file — create file handle to file on disk
    - Like setting up a buffer for the file
- Read and write operations are to file handle

Reading/writing disk data

- Open a file — create file handle to file on disk
  - Like setting up a buffer for the file
- Read and write operations are to file handle
- Close a file
  - Write out buffer to disk (flush)
  - Disconnect file handle

Opening a file

Opening a file

```
fh = open("gcd.py", "r")
```

Opening a file

```
fh = open("gcd.py", "r")
```

- First argument to `open` is file name
  - Can give a full path

Opening a file

```
fh = open("gcd.py", "r")
```

- First argument to `open` is file name
  - Can give a full path
- Second argument is mode for opening file
  - Read, `"r"`: opens a file for reading only
  - Write, `"w"`: creates an empty file to write to
  - Append, `"a"`: append to an existing file

Read through file handle

Read through file handle

```
contents = fh.read()
```

- Reads entire file into name as a single string

Read through file handle

```
contents = fh.read()
```

- Reads entire file into name as a single string

```
contents = fh.readline()
```

- Reads one line into name—lines end with `'\n'`
    - String includes the `'\n'`, unlike `input()`

Read through file handle

```
contents = fh.read()
```

- Reads entire file into name as a single string

```
contents = fh.readline()
```

- Reads one line into name—lines end with `'\n'`
  - String includes the `'\n'`, unlike `input()`

```
contents = fh.readlines()
```

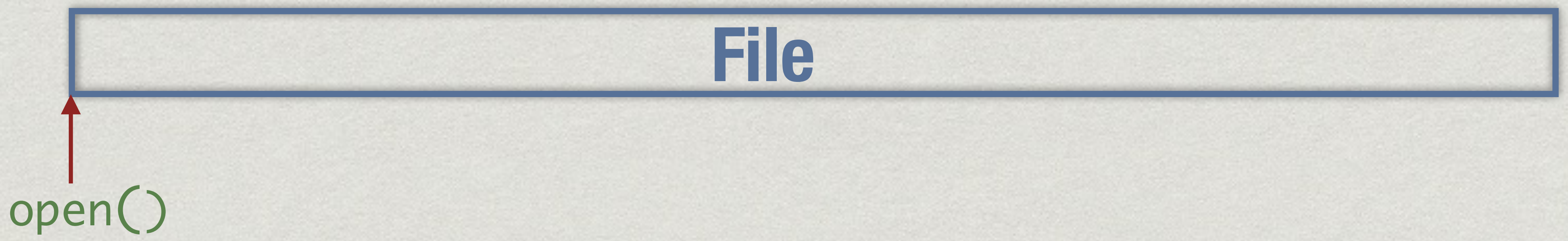
- Reads entire file as list of strings
  - Each string is one line, ending with `'\n'`

Reading files

Reading files

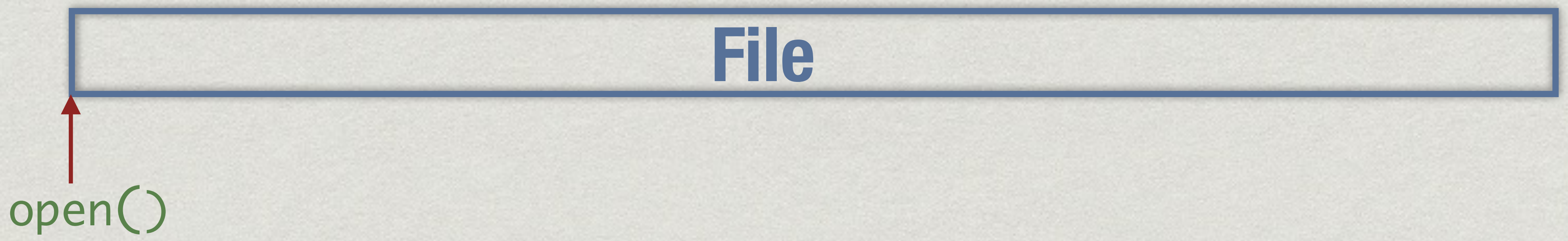
- Reading is a sequential operation

Reading files



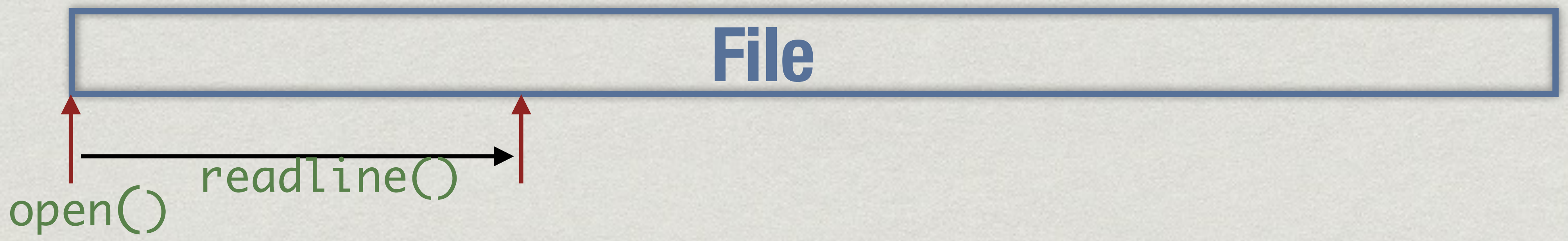
- Reading is a sequential operation
    - When file is opened, point to position 0, the start

Reading files



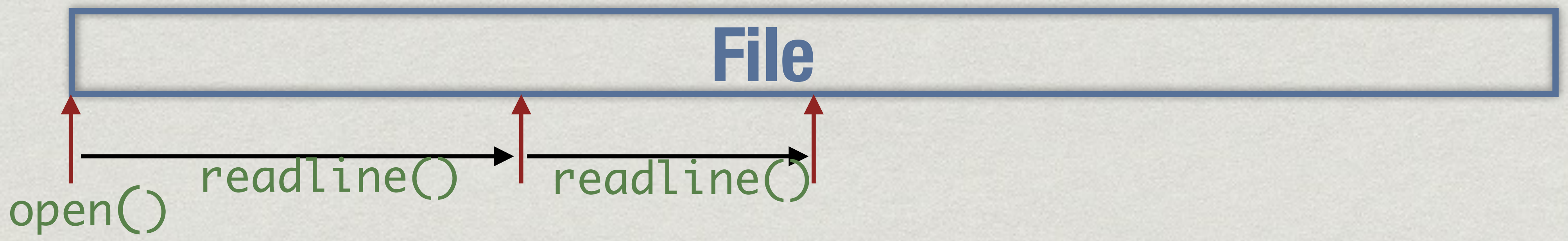
- Reading is a sequential operation
    - When file is opened, point to position 0, the start
    - Each successive `readline()` moves forward

Reading files



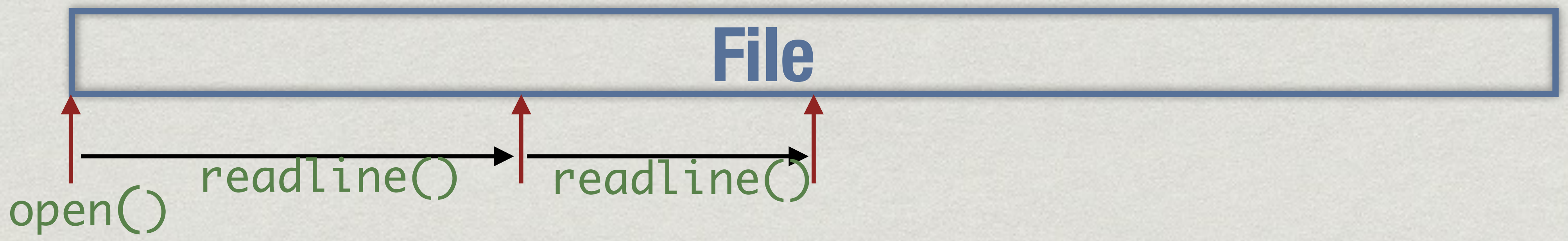
- Reading is a sequential operation
  - When file is opened, point to position 0, the start
  - Each successive `readline()` moves forward

Reading files



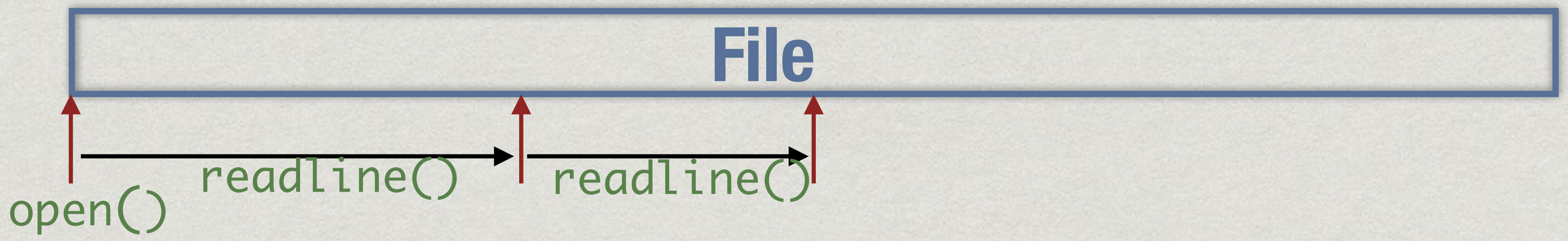
- Reading is a sequential operation
  - When file is opened, point to position 0, the start
  - Each successive `readline()` moves forward

Reading files



- Reading is a sequential operation
    - When file is opened, point to position 0, the start
    - Each successive `readline()` moves forward
- `fh.seek(n)` — moves pointer to position `n`

Reading files



- Reading is a sequential operation
  - When file is opened, point to position 0, the start
  - Each successive `readline()` moves forward
- `fh.seek(n)` — moves pointer to position `n`
- `block = fh.read(12)` — read a fixed number of characters

End of file

End of file

* When reading incrementally, important to know when file has ended

End of file

- When reading incrementally, important to know when file has ended
- The following both signal end of file
  - `fh.read()` returns empty string `""`
  - `fh.readline()` returns empty string `""`

Writing to a file

Writing to a file

```
fh.write(s)
```

- Write string `s` to file
  - Returns number of characters written
  - Include `'\n'` explicitly to go to a new line

Writing to a file

```
fh.write(s)
```

- Write string `s` to file
  - Returns number of characters written
  - Include `'\n'` explicitly to go to a new line

```
fh.writelines(l)
```

- Write a list of lines `l` to file
  - Must includes `'\n'` explicitly for each string

Closing a file

Closing a file

```
fh.close()
```

- Flushes output buffer and decouples file handle
  - All pending writes copied to disk

Closing a file

```
fh.close()
```

- Flushes output buffer and decouples file handle
    - All pending writes copied to disk

```
fh.flush()
```

- Manually forces write to disk

Processing file line by line

Processing file line by line

```
contents = fh.readlines()
for l in contents:
  . . .
```

Processing file line by line

```
contents = fh.readlines()
for l in contents:
    . . .
```

- Even better

```
for l in fh.readlines():
    . . .
```

Copying a file

```
infile = open("input.txt", "r")
outfile = open("output.txt", "w")
for line in infile.readlines():
  outfile.write(line)
infile.close()
outfile.close()
```

Copying a file

```
infile = open("input.txt", "r")

outfile = open("output.txt", "w")

contents = infile.readlines()

outfile.writelines(contents)

infile.close()

outfile.close()
```

Strip new line character

Strip new line character

- Get rid of trailing `'\n'`

```
contents = fh.readlines()
for line in contents:
  s = line[:-1]
```

Strip new line character

- Get rid of trailing `'\n'`

```
contents = fh.readlines()
for line in contents:
  s = line[:-1]
```

- Instead, use `rstrip()` to remove trailing whitespace

```
for line in contents:
  s = line.rstrip()
```

Strip new line character

- Get rid of trailing `'\n'`

```
contents = fh.readlines()
for line in contents:
  s = line[:-1]
```

- Instead, use `rstrip()` to remove trailing whitespace

```
for line in contents:
  s = line.rstrip()
```

- Also `strip()` — both sides, `lstrip()` — from left
  - String manipulation functions — coming up

Summary

- Interact with files through file handles
- Open a file in one of three modes — read, write, append
- Read entire file as a string, or line by line
- Write a string, or a list of strings to a file
- Close handle, flush buffer
- String operations to strip white space