Files

Open a file $\rightsquigarrow$ associate file on disk with a "file handle" in program

Read/write file via file handle

Close file $\rightsquigarrow$ disconnects file handle
complete pending writes

Opening a file

"Mode" of operation – read / write / append
| overwrite

fh = open("myfile.txt", "r")

FileNotFoundError ⟷ "r" ⟶ or "w" or "a"

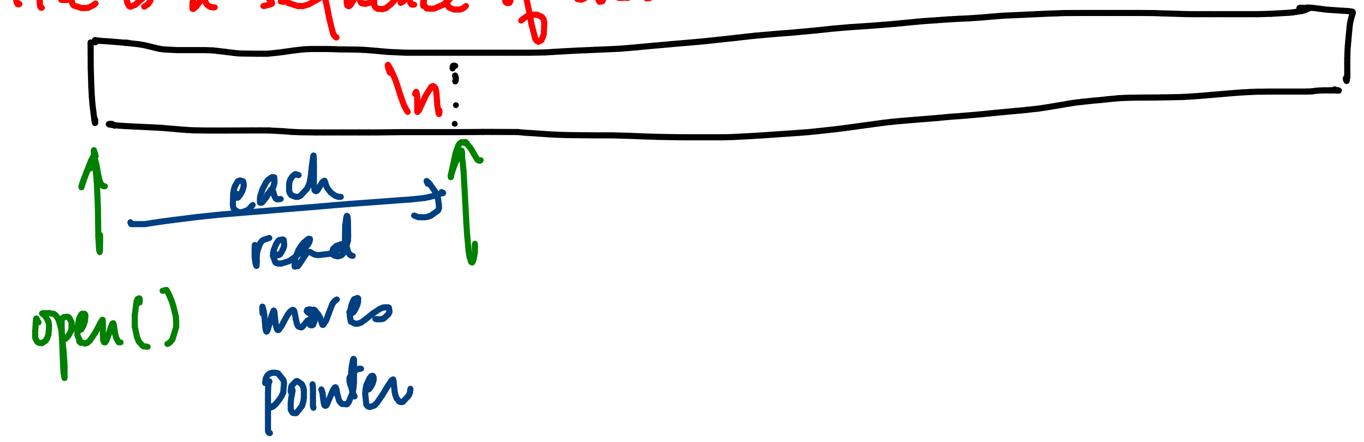
Reading contents

contents = fh.read() swallow entire file as a string

nextline = fh.readline() upto and including next \n

alllines = fh.readlines() list of strings - one per line, with \n

File is a sequence of characters



fh.seek(n) reposition the pointer to position n

fh.read(12) read a fixed no. of characters

Detecting end of file?

Write file

fh.write(s) string, must add \n manually

fh.writelines(l) l - list of strings, add \n yourself

Copy a.txt to b.txt

```
inh = open("a.txt", "r")
outh = open("b.txt", "w")
contents = inh.readlines()
outh.writelines(contents)
```

Close file

```
fh.close()     # "flushes" buffers
```

Manually force pending writes

```
fh.flush()
```

Useful operations on strings

Stripping whitespace

`s.rstrip()` - returns copy of s with trailing whitespace removed

`s.lstrip()` - leading whitespace

`s.strip()` - lstrip + rstrip

Splitting a string

`parts = s.split(':')` ← explicit character to split on

`words = s.split()` - all whitespace

User defined datatypes

Dictionary d = {'a':1, 'b':2}

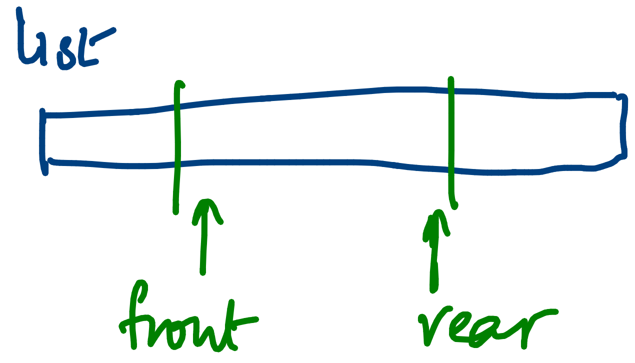
Not supposed to expect values in d to be in a particular order

"give me the second value in d"

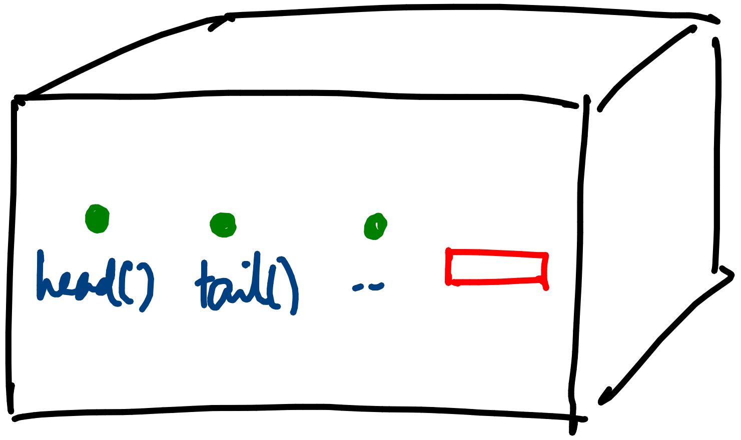
Queue

head()

append()



Limit the use of such datatypes to specified "public" interface



"Abstract Datatypes"

Define ADT without reference to implementation

e.g. stack

empty() — returns empty stack

pop(push(empty(), a)) = a

⋮

Changing implementation should not affect interface

How do we define our own ADTs ?