1 Opening a File

To open a file, we need a string that holds its ***filename***. Some ways to get a filename:

- Say it literally, if your program is always going to use the same name.
- Read it from the user using `raw_input`.
- Get it from the user using `media.choose_file`.

We ask Python to ***open*** the file with builtin function **`open`**:

```
>>> # open the file 'story.txt' for reading
>>> f = open('story.txt', 'r')
>>> print f
<open file 'story.txt', mode 'r' at 0x7e1860>
>>> # r: read from the file
>>> f   # f is type file
<open file 'story.txt', mode 'r' at 0x7e1860>
>>> dir(file)
['__class__', '__delattr__', '__doc__', '__enter__', '__exit__', '__format__', '__getattribute__', '__hash__', '__init__', '__iter__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'close', 'closed', 'encoding', 'errors', 'fileno', 'flush', 'isatty', 'mode', 'name', 'newlines', 'next', 'read', 'readinto', 'readline', 'readlines', 'seek', 'softspace', 'tell', 'truncate', 'write', 'writelines', 'xreadlines']
```

The **`open`** function returns an object of type **`file`**:

The second argument to **`open`** is says how the file is to be used. We'll use these modes:

- 'r': will only read from it
- 'w': will only write to it (and if it already exists, the old contents are erased)
- 'a': will only write to it (and if it already exists, new content will be appended to the end)

If you leave out the mode, it defaults to 'r'.

2 Reading From A File

We call an open file for reading a ***reader***. There are four ways to read from a reader:

1. ***Read a single line***

```
>>> line = f.readline()
>>> print line
Once upon a time there was a curious little girl
>>> line = f.readline()
>>> print line
named Goldilocks.  Goldilocks lived near the woods
>>> line
'named Goldilocks.  Goldilocks lived near the woods\n'
>>> line = f.readline()
>>> print line
and loved to go exploring.
```

```
>>> line = f.readline()
>>> line
>>> '\n'
>>> line = f.readline()
>>> line = f.readline()
>>> line
'The door was open, and she decided to take a peak \n'
>>> line = f.readline()
line
>>> 'inside.'
>>> line = f.readline()  # when end of file is reached, readline returns '' (empty string)
>>> line
''
>>> line = f.readline()
>>> line
''
>>> f.close()
```

2. ***Read a certain number of characters***

```
>>> myfile = open('story.txt')
>>> s = myfile.read(10)   # read 10 characters
>>> s
'Once upon '
>>> s = myfile.read(10)   # read 10 characters
>>> s
'a time the'
>>> s = myfile.read(20)
>>> s
're was a curious lit'
>>> s = myfile.read(5)
>>> s
'tle g'
```

```
>>> s = myfile.read(4)
>>> s     # \n is a single character
'irl\n'
>>> myfile.close()
>>> myfile = open('story.txt')
>>> for line in myfile:
        print line
```

```
>>> myfile.close()
>>> myfile = open('story.txt')
>>> for line in myfile:
 print line,  # suppress the newline that print usually adds
```

```
>>> myfile.close()
>>> myfile = open('story.txt')
>>> for line in myfile:
 print line.rstrip()   # strip the whitespace from the right-hand side of the line
```

3. ***Read a line at a time from beginning to end***

Q. Why is the output from the for loop double-spaced?

A.

Q. How can you single space the output?

A. Either one of these will do it:

(a)

```
>>> 'testing 123
\t\n'.rstrip()
>>> 'testing 123'
>>> len('\n')
1
>>> myfile.close()
>>> myfile = open
('story.txt')
>>> s = myfile.read()
>>> s
```

(b)

It's common to need to strip white space from either end of a line, or both.

4. ***Read everything in the file into one string***

3 Dealing With The End Of File

With the for loop approach, the loop automatically stops when the end of the file is encountered. Or never even iterates once if the file is empty!

Q. What happens if you are at the end of the file when you call **`read()`** or **`readline()`**?

A.

```
>>> myfile.readline()
''
>>> myfile.close()
>>> myfile = open('story.txt')
>>> next_line = myfile.readline()
>>> while next_line != '':
 next_line = myfile.readline()
```

Example.

```
# Detecting the end of the file while reading line by line.
 >>> next_line
 "                                                    >>> while next_line != '':
 >>> myfile.close()                                    print next_line
 >>> myfile = open('story.txt')                        next_line = myfile.readline()
 >>> next_line = myfile.readline()
 >>> print next_line
 Once upon a time there was a curious little girl

Once upon a time there was a curious little girl

named Goldilocks.  Goldilocks lived near the woods

and loved to go exploring.
```

Q. Why doesn't this loop stop when you get to any empty line?

A.

4 Closing A File

```
   One day Goldilocks came upon a comfy little cottage.

The door was open, and she decided to take a peak

inside.
>>> print next_line

>>> next_line
"
```

4.1 Writing To And Then Reading From The Same File

You can both read to and write from the same file (just not at the same time).

```
>>>  # mode 'w'
>>> output = open('letter.txt', 'w')
>>> output.write('Dear Uncle Marty,\n')
>>> output.close()
>>> myfile = open('letter.txt')
>>> s = myfile.readline()
>>> s
'Dear Uncle Marty,\n'
>>> myfile.close()

>>> # mode 'a'
>>> myfile = open('letter.txt', 'a')
>>> myfile.write('I hope you are well.')
>>> myfile.close()
[evaluate comments.py]
[evaluate comments.py]
#I hope you are well.

# another comment
[evaluate comments.py]
#I hope you are well.
# another comment
[evaluate comments.py]
 #I hope you are well.
   #comment
# another comment
```

Q. What happened to the file contents?

A.

4.2 Let's Practice!

[**comments.py**]

```
import media

def prompt_and_open():
    '''() -> file
    Prompt the user to enter a file name, open the file
    for reading, and return the open file object.'''

    filename = media.choose_file()
    input_file = open(filename)
    return input_file



def print_starts_with(our_file, char):
    '''(file, str) -> NoneType
    Print the lines of our_file that start with the character char.'''

    for line in our_file:
        if line.lstrip().startswith(char):
            print line,


if __name__ == '__main__':
```

5 Reading Files From The Web!

The module **urllib** lets you read files from the web. You give a url, and it opens the webpage, giving you an object (a complex one!) that you can iterate through, line by line.
[**slowdown.py**]