

CSE 503

Introduction to Computer Science for Non-Majors

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 17
Read from Files

Announcements

- Lab #2 due Monday @ 11:59PM

Reading Files

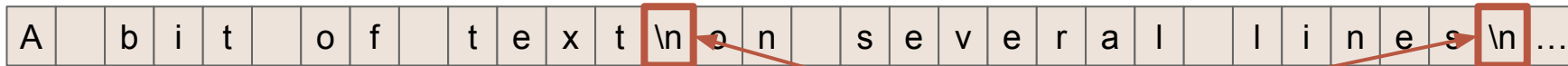
A text file is a sequence of characters:

A		b	i	t		o	f		t	e	x	t	\n	o	n		s	e	v	e	r	a	l		l	i	n	e	s	\n	...
---	--	---	---	---	--	---	---	--	---	---	---	---	----	---	---	--	---	---	---	---	---	---	---	--	---	---	---	---	---	----	-----

Reading Files

A text file is a sequence of characters:

A b i t o f t e x t \n o n s e v e r a l l i n e s \n ...



These are "newline" characters

Reading Files

A text file is a sequence of characters:

A		b	i	t		o	f		t	e	x	t	\n	o	n		s	e	v	e	r	a	l		l	i	n	e	s	\n	...
---	--	---	---	---	--	---	---	--	---	---	---	---	----	---	---	--	---	---	---	---	---	---	---	--	---	---	---	---	---	----	-----

We can read a file line by line:

A		b	i	t		o	f		t	e	x	t	\n
---	--	---	---	---	--	---	---	--	---	---	---	---	----

o	n		s	e	v	e	r	a	l		l	i	n	e	s	\n
---	---	--	---	---	---	---	---	---	---	--	---	---	---	---	---	----

...

Reading Files

A text file is a sequence of characters:

A b i t o f t e x t \n o n s e v e r a l l i n e s \n ...

We can read a file line by line:

A b i t o f t e x t \n

o n s e v e r a l l i n e s \n

...

Each line ends with a newline character

Opening Files in Python

The `open()` function opens a file:

```
open("test1.txt", "r")
```

It takes two arguments:

1. the name of the file (a string)
2. how you would like to open it (another string: "r", "w", "rw", "a", etc...)

Opening Files in Python

The `open(...)` function is usually used with a [with...as](#) statement:

```
with open("test1.txt", "r") as f:  
    # do something with the file...
```

`f` is a variable. It refers to a [file object](#).

The `with...as` statement ensures that the file is automatically closed at the end of the suite of statements, no matter what happens.

File Objects and Iteration

File objects support iteration so we can use a for loop to iterate over each line in a file:

```
with open("test1.txt", "r") as f:  
    for line in f:  
        # do something with each line...
```

File Objects and Iteration

File objects support iteration so we can use a for loop to iterate over each line in a file:

```
with open("test1.txt", "r") as f:  
    for line in f:  
        # do something with each line...  
        print(line)
```

Lines end with a newline

A text file is a sequence of characters:

A b i t o f t e x t \n o n s e v e r a l l i n e s \n ...

We can read a file line by line:

A b i t o f t e x t \n
o n s e v e r a l l i n e s \n
...

Each line ends with a newline character

Option #1

Our first option is to specify a different end string for the `print` function:

```
with open("test1.txt", "r") as f:  
    for line in f:  
        # do something with each line...  
        print(line, end="")
```

Option #1

Our first option is to specify a different end string for the `print` function:

```
with open("test1.txt", "r") as f:  
    for line in f:  
        # do something with each line...  
        print(line, end="")
```

What happens with different end strings?

Option #2

Our second option is to remove the new lines from the lines in the file:

```
with open("test1.txt", "r") as f:
    for line in f:
        # do something with each line...
        line = line.rstrip("\n")
        print(line)
```

Formatted Output

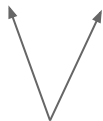
We can use string formatting to make our output look a little nicer:

```
with open("test1.txt", "r") as f:
    count = 0
    for line in f:
        # do something with each line...
        line = line.rstrip("\n")
        count = count + 1
        print("Line #{0}: {1}".format(count, line))
```

Formatted Output

We can use string formatting to make our output look a little nicer:

```
with open("test1.txt", "r") as f:
    count = 0
    for line in f:
        # do something with each line...
        line = line.rstrip("\n")
        count = count + 1
        print("Line #{0}: {1}".format(count, line))
```



{0} and {1} are placeholders

Formatted Output

We can use string formatting to make our output look a little nicer:

```
with open("test1.txt", "r") as f:
    count = 0
    for line in f:
        # do something with each line...
        line = line.rstrip("\n")
        count = count + 1
        print("Line #{0:03d}: {1}".format(count, line))
```

We have more control over how numbers are printed. More on string formatting [here](#).

Exercise

Define a function that takes a filename as an argument, and returns a dictionary of character counts for the file.

For example, if the file contains the character "a" 12 times, and "e" 17 times, the returned dictionary would have `"a":12` and `"e":17`, in addition to the counts of the other characters in the file.