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 bit of text
non several lines
...

A text file is a sequence of characters:

A bit of text

These are "newline" characters
A text file is a sequence of characters:

A bit of text

on several lines

We can read a file line by line:

A bit of text

on several lines

...
A text file is a sequence of characters:

```
A bit of text
```
on several lines

...
Opening Files in Python

The `open()` function opens a file:

```python
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:

```python
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:

```python
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:

```python
with open("test1.txt","r") as f:
    for line in f:
        # do something with each line...
        print(line)
```
A text file is a sequence of characters:

```
A bit of text
on several lines
```

We can read a file line by line:

```
A bit of text
on several lines
```

Each line ends with a newline character.
Option #1

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

```python
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:

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

```python
with open("test1.txt","r") as f:
    for line in f:
        # do something with each line...
        line = line.rstrip("\n")
        print(line)
```
We can use string formatting to make our output look a little nicer:

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

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

```python
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](#).
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.