Using Objects; Working with Text Files

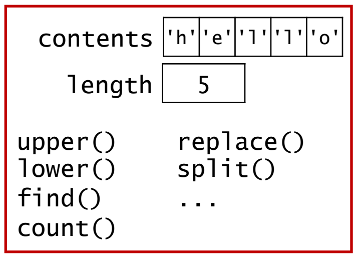
Computer Science 111
Boston University

Vahid Azadeh-Ranjbar, Ph.D.

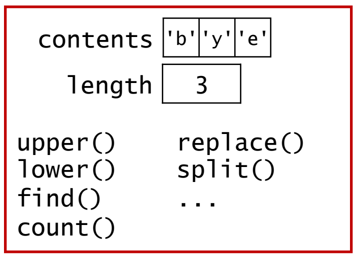
Recall: Strings Are Objects

- In Python, a string is an object.
  - ***attributes:***
    - the characters in the string
    - the length of the string
  - ***methods:*** functions inside the string that we can use to operate on the string

string object for `'hello'`



string object for `'bye'`



Recall: String Methods (partial list)

- `s.lower()`: return a copy of `s` with all lowercase characters
- `s.upper()`: return a copy of `s` with all uppercase characters
- `s.find(sub)`: return the index of the first occurrence of the substring `sub` in the string `s` (-1 if not found)
- `s.count(sub)`: return the number of occurrences of the substring `sub` in the string `s` (0 if not found)
- `s.replace(target, repl)`: return a new string in which all occurrences of `target` in `s` are replaced with `repl`

Examples of Using String Methods

```
>>> chant = 'We are the Terriers!'
>>> chant.upper()

>>> chant.lower()

>>> chant.replace('e', 'o')
```

Examples of Using String Methods

```
>>> chant = 'We are the Terriers!'

>>> chant.upper()
'WE ARE THE TERRIERS!'

>>> chant.lower()


>>> chant.replace('e', 'o')
```

Examples of Using String Methods

```
>>> chant = 'We are the Terriers!'

>>> chant.upper()
'WE ARE THE TERRIERS!'

>>> chant.lower()
'we are the terriers!'

>>> chant.replace('e', 'o')
```

Examples of Using String Methods

```
>>> chant = 'We are the Terriers!'

>>> chant.upper()
'WE ARE THE TERRIERS!'

>>> chant.lower()
'we are the terriers!'

>>> chant.replace('e', 'o')
'Wo aro tho Torriors!'
```

Splitting a String

- The `split()` method breaks a string into a list of substrings.

```
>>> name = 'Martin Luther King'
>>> name.split()
['Martin', 'Luther', 'King']
```

- By default, it uses *whitespace characters* (spaces, tabs, and newlines) to determine where the splits should occur.

Splitting a String

- The `split()` method breaks a string into a list of substrings.

```
>>> name = 'Martin Luther King'
>>> name.split()
['Martin', 'Luther', 'King']
>>> components = name.split()
```

- By default, it uses *whitespace characters* (spaces, tabs, and newlines) to determine where the splits should occur.

Splitting a String

- The `split()` method breaks a string into a list of substrings.

```
>>> name = 'Martin Luther King'
>>> name.split()
['Martin', 'Luther', 'King']
>>> components = name.split()
>>> components[0]
'Martin'
```

- By default, it uses *whitespace characters* (spaces, tabs, and newlines) to determine where the splits should occur.

Splitting a String

- The `split()` method breaks a string into a list of substrings.

```
>>> name = 'Martin Luther King'
>>> name.split()
['Martin', 'Luther', 'King']
>>> components = name.split()
>>> components[0]
'Martin'
```

- By default, it uses *whitespace characters* (spaces, tabs, and newlines) to determine where the splits should occur.
- You can specify a different separator:

```
>>> date = '11/10/2014'
>>>
```

Splitting a String

- The `split()` method breaks a string into a list of substrings.

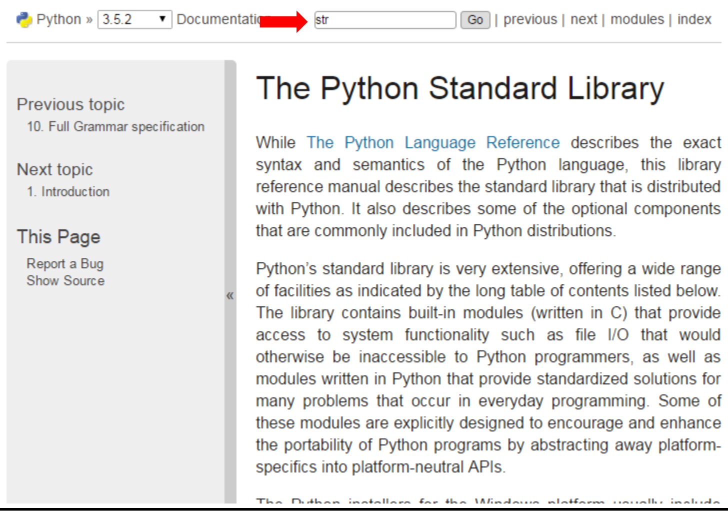
```
>>> name = 'Martin Luther King'
>>> name.split()
['Martin', 'Luther', 'King']
>>> components = name.split()
>>> components[0]
'Martin'
```

- By default, it uses *whitespace characters* (spaces, tabs, and newlines) to determine where the splits should occur.
- You can specify a different separator:

```
>>> date = '11/10/2014'
>>> date.split('/')
['11', '10', '2014']
```

Discovering What An Object Can Do

- Use the documentation for the **Python Standard Library**:
  docs.python.org/3/library



Python » 3.5.2 Documentation str Go | previous | next | modules | index

Previous topic
10. Full Grammar specification

Next topic
1. Introduction

This Page
Report a Bug
Show Source

The Python Standard Library

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

What is the output of this program?

```
s = '    programming    '
s = s.strip()
s.upper()
s = s.split('r')
print(s)
```

A. `['    p', 'og', 'amming    ']`
B. `['p', 'og', 'amming']`
C. `['    P', 'OG', 'AMMING    ']`
D. `['P', 'OG', 'AMMING']`
E. none of the above

What is the output of this program?

```
s = '    programming   '
s = s.strip()
s.upper()
s = s.split('r')
print(s)
```

A. `['    p', 'og', 'amming   ']`

B. `['p', 'og', 'amming']`

C. `['    P', 'OG', 'AMMING   ']`

D. `['P', 'OG', 'AMMING']`

E. none of the above

What is the output of this program?

```
s = '    programming   '
s = s.strip()
s.upper()
s = s.split('r')
print(s)
```

A. `['    p', 'og', 'amming   ']`

B. `['p', 'og', 'amming']`

C. `['    P', 'OG', 'AMMING   ']`

D. `['P', 'OG', 'AMMING']`

E. none of the above

What is the output of this program?

```
s = '    programming   '
s = s.strip()      # s = 'programming'
s.upper()
s = s.split('r')
print(s)
```

A. `['    p', 'og', 'amming   ']`
B. `['p', 'og', 'amming']`
C. `['    P', 'OG', 'AMMING   ']`
D. `['P', 'OG', 'AMMING']`
E. `none of the above`

What is the output of this program?

```
s = '    programming   '
s = s.strip()      # s = 'programming'
s.upper()
s = s.split('r')
print(s)
```

A. `['    p', 'og', 'amming   ']`
B. `['p', 'og', 'amming']`
C. `['    P', 'OG', 'AMMING   ']`
D. `['P', 'OG', 'AMMING']`
E. `none of the above`

What is the output of this program?

```
s = '    programming   '
s = s.strip()      # s = 'programming'
s.upper()          # 'PROGRAMMING' (no change to s!)
s = s.split('r')
print(s)
```

A. `['    p', 'og', 'amming   ']`

B. `['p', 'og', 'amming']`

C. `['    P', 'OG', 'AMMING   ']`

D. `['P', 'OG', 'AMMING']`

E. none of the above

What is the output of this program?

```
s = '    programming   '
s = s.strip()      # s = 'programming'
s.upper()          # 'PROGRAMMING' (no change to s!)
s = s.split('r')
print(s)
```

A. `['    p', 'og', 'amming   ']`

B. `['p', 'og', 'amming']`

C. `['    P', 'OG', 'AMMING   ']`

D. `['P', 'OG', 'AMMING']`

E. none of the above

What is the output of this program?

```
s = '    programming   '
s = s.strip()       # s = 'programming'
s.upper()           # 'PROGRAMMING' (no change to s!)
s = s.split('r')    # s = ['p', 'og', 'amming']
print(s)
```

A. `['    p', 'og', 'amming   ']`

B. `['p', 'og', 'amming']`

C. `['    P', 'OG', 'AMMING   ']`

D. `['P', 'OG', 'AMMING']`

E. none of the above

What is the output of this program?

```
s = '    programming   '
s = s.strip()       # s = 'programming'
s.upper()           # 'PROGRAMMING' (no change to s!)
s = s.split('r')    # s = ['p', 'og', 'amming']
print(s)
```

A. `['    p', 'og', 'amming   ']`

B. **`['p', 'og', 'amming']`**

C. `['    P', 'OG', 'AMMING   ']`

D. `['P', 'OG', 'AMMING']`

E. none of the above

Recall: Text Files

- A text file can be thought of as one long string.
- The end of each line is stored as a newline character (`'\n'`).
- Example: the following three-line text file

```
Don't forget!

Test your code fully!
```

is equivalent to the following string:

```
'Don't forget!\n\nTest your code fully!\n'
```

Recall: Opening a Text File

- Before we can read from a text file, we need to *open* a connection to the file.
- Example:

```
f = open('reminder.txt', 'r')
```

where:

  - `'reminder.txt'` is the name of the file we want to read
  - `'r'` indicates that we want to read from the file (if we leave this out, Python will assume it)

- Doing so creates an object known as a *file handle*.
  - we use the file handle to perform operations on the file

Recall: Processing a File Using Methods

- A file handle is an object.
- We can use its methods to process a file.

reminder.txt
```
Don't forget!

Test your code fully!
```

```
>>> f = open('reminder.txt', 'r')
>>> f.readline()
"Don't forget!\n"
>>> f.readline()
'\n'
>>> f.readline()
'Test your code fully!\n'
>>> f.readline()
''
>>> f = open('reminder.txt', 'r')   # start over at top
>>> f.read()
"Don't forget!\n\nTest your code fully!\n"
```

Processing a File Using a `for` Loop

- We often want to read and process a file one line at a time.
- We could use readline() inside a loop, but...
  - what's the problem we would face?

Processing a File Using a `for` Loop

- We often want to read and process a file one line at a time.
- We could use readline() inside a loop, but...
  - what's the problem we would face?
    we don't know how many lines there are

Processing a File Using a `for` Loop

- We often want to read and process a file one line at a time.
- We could use readline() inside a loop, but...
  - what's the problem we would face?
    we don't know how many lines there are
- Python makes it easy!

```
for line in file-handle:
    # code to process line goes here
```

  - reads one line at a time and assigns it to `line`
  - continues looping until there are no lines left

Processing a CSV File

- CSV = comma-separated values

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

Processing a CSV File

- CSV = comma-separated values
  - each line is one *record*

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

Processing a CSV File

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

- CSV = comma-separated values
  - each line is one *record*
  - the *fields* in a given record are separated by commas

How Should We Fill in the Blank?

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = ___________________
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

A. `file.split()`

B. `line.split()`

C. `file.split(',')`

D. `line.split(',')`

E. none of the above

How Should We Fill in the Blank?

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = ___________________
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

A. `file.split()`

B. `line.split()`

C. `file.split(',')`

D. `line.split(',')`

E. none of the above

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

```
line                     fields                        output     count
                                                                  0
'CS,111,MWF 10-11\n'
```

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

```
line                     fields                        output     count
                                                                  0
'CS,111,MWF 10-11\n'
'CS,111,MWF 10-11'
```

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

<u>line</u>	<u>fields</u>	<u>output</u>	<u>count</u>
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'			

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

<u>line</u>	<u>fields</u>	<u>output</u>	<u>count</u>
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	**['CS','111','MWF 10-11']**		

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

```
line                    fields                        output   count
                                                               0
'CS,111,MWF 10-11\n'
'CS,111,MWF 10-11'      ['CS','111','MWF 10-11']
```

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

```
line                    fields                        output   count
                                                               0
'CS,111,MWF 10-11\n'
'CS,111,MWF 10-11'      ['CS','111','MWF 10-11']    CS 111
```

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

```
line                 fields                     output    count
                                                          0
'CS,111,MWF 10-11\n'
'CS,111,MWF 10-11'   ['CS','111','MWF 10-11']   CS 111    1
'MA,123,TR 3-5\n'
```

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

```
line                 fields                     output    count
                                                          0
'CS,111,MWF 10-11\n'
'CS,111,MWF 10-11'   ['CS','111','MWF 10-11']   CS 111    1
'MA,123,TR 3-5\n'
'MA,123,TR 3-5'
```

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	**['MA','123','TR 3-5']**		

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	**['MA','123','TR 3-5']**		

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			
'CS,105,MWF 1-2'			

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			
'CS,105,MWF 1-2'	**['CS','105','MWF 1-2']**		

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			
'CS,105,MWF 1-2'	**['CS','105','MWF 1-2']**		

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			
'CS,105,MWF 1-2'	['CS','105','MWF 1-2']	**CS 105**	

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			
'CS,105,MWF 1-2'	['CS','105','MWF 1-2']	CS 105	2

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			
'CS,105,MWF 1-2'	['CS','105','MWF 1-2']	CS 105	2
...			

Closing a File

- When you're done with a file, close your connection to it:

```
    file.close()      # file is the file handle
```

  - another example of a method inside an object!

Closing a File

- When you're done with a file, close your connection to it:

```
    file.close()      # file is the file handle
```

  - another example of a method inside an object!
- This isn't crucial when reading from a file.
- It *is* crucial when writing to a file, which we'll do later.
  - text that you write to file may not make it
    to disk until you close the file handle!

Extracting Relevant Data from a File

- Assume that the results of a track meet are summarized in a comma-delimited text file (a *CSV file*) that looks like this:

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

- We'd like to have a function that reads in such a results file and extracts just the results for a particular school.
  - example:

```
>>> extract_results('track_results.txt', 'BU')
Mike Mercury mile 4:50:00
Len Lightning half-mile 2:15:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        # fill in the rest of the loop body...
        # when you find a match for target_school,
        # print the athlete, event, and time.




    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
fields[0]  fields[1]  fields[2]  fields[3]
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
fields[0]  fields[1]  fields[2]  fields[3]
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

fields[0] fields[1] fields[2] fields[3]

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
    # return!
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Handling Schools with No Records

- We'd like to print a message when the target school does not appear in the file.
- Would this work?

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')

        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])
        else:
            print(target_school, 'not found')

    file.close()
```

Handling Schools with No Records

- We'd like to print a message when the target school does not appear in the file.
- Would this work? ***no!***

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')

        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])
        else:
            print(target_school, 'not found')

    file.close()
```

Handling Schools with No Records (cont.)

- Solution: use a variable to count how many matches we find.
- Would this work?

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    count = 0
    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])
            count += 1
        if count == 0:
            print(target_school, 'not found')

    file.close()
```

Handling Schools with No Records (cont.)

- Solution: use a variable to count how many matches we find.
- Would this work? ***no!***

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    count = 0
    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])
            count += 1
        if count == 0:
            print(target_school, 'not found')

    file.close()
```

Handling Schools with No Records (cont.)

- Solution: use a variable to count how many matches we find.
- This *does* work:

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    count = 0
    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])
            count += 1
    if count == 0:
        print(target_school, 'not found')

    file.close()
```

Reminders

- Midterm 2: **next Wednesday night, 6:30-7:30 pm**
  - see the info. sheet on course website
  - includes practice problems