

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

**1** A Virtual University

description of an application  
the four pillars in the application

**2** Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

**3** Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

MCS 275 Lecture 36  
Programming Tools and File Management  
Jan Vershelde, 12 April 2010

# an application

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

### 1 A Virtual University

description of an application

the four pillars in the application

### 2 Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

### 3 Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# A Virtual University

description of an application

The running example in *Making Use of Python* is the administration of Techsity University.

The application rests on four pillars:

- 1 web forms with CGI scripts,
- 2 information management with databases,
- 3 multiple servers to handle the load,
- 4 multithreaded servers handle many clients.

→ chapters 10, 11, 12, and 13 in the book.

# A Virtual University

description of an application

The running example in *Making Use of Python* is the administration of Techsity University.

The application rests on four pillars:

- 1 web forms with CGI scripts,
- 2 information management with databases,
- 3 multiple servers to handle the load,
- 4 multithreaded servers handle many clients.

→ chapters 10, 11, 12, and 13 in the book.

12 Apr 2010

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to upload  
processing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offerings  
the `HTMLParser`  
module  
overriding methods  
in `HTMLParser`  
filtering attributes of  
tags

## A Virtual University

description of an application

The running example in *Making Use of Python* is the administration of Techsity University.

The application rests on four pillars:

- 1 web forms with CGI scripts,
- 2 information management with databases,
- 3 multiple servers to handle the load,
- 4 multithreaded servers handle many clients.

→ chapters 10, 11, 12, and 13 in the book.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to upload  
processing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offerings  
the `HTMLParser`  
module  
overriding methods  
in `HTMLParser`  
filtering attributes of  
tags

## A Virtual University

description of an application

The running example in *Making Use of Python* is the administration of Techsity University.

The application rests on four pillars:

- 1 web forms with CGI scripts,
- 2 information management with databases,
- 3 multiple servers to handle the load,
- 4 multithreaded servers handle many clients.

→ chapters 10, 11, 12, and 13 in the book.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to upload  
processing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offerings  
the `HTMLParser`  
module  
overriding methods  
in `HTMLParser`  
filtering attributes of  
tags

## A Virtual University

description of an application

The running example in *Making Use of Python* is the administration of Techsity University.

The application rests on four pillars:

- 1 web forms with CGI scripts,
- 2 information management with databases,
- 3 multiple servers to handle the load,
- 4 multithreaded servers handle many clients.

→ chapters 10, 11, 12, and 13 in the book.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to upload  
processing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offerings  
the `HTMLParser`  
module  
overriding methods  
in `HTMLParser`  
filtering attributes of  
tags

## A Virtual University

description of an application

The running example in *Making Use of Python* is the administration of Techsity University.

The application rests on four pillars:

- 1 web forms with CGI scripts,
- 2 information management with databases,
- 3 multiple servers to handle the load,
- 4 multithreaded servers handle many clients.

→ chapters 10, 11, 12, and 13 in the book.

# an application

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

### 1 A Virtual University

description of an application

the four pillars in the application

### 2 Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

### 3 Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# CGI Programming

web interfaces

## Course Administration:

- browsing of the course catalog,
- answering queries about courses,
- course registration.

## Online Courses:

- browsing through the course materials,
- downloading notes and slides,
- online class and lab interactions,
- uploading answers to assignments.

# CGI Programming

web interfaces

## Course Administration:

- browsing of the course catalog,
- answering queries about courses,
- course registration.

## Online Courses:

- browsing through the course materials,
- downloading notes and slides,
- online class and lab interactions,
- uploading answers to assignments.

# CGI Programming

web interfaces

## Course Administration:

- browsing of the course catalog,
- answering queries about courses,
- course registration.

## Online Courses:

- browsing through the course materials,
- downloading notes and slides,
- online class and lab interactions,
- uploading answers to assignments.

# CGI Programming

## web interfaces

### Course Administration:

- browsing of the course catalog,
- answering queries about courses,
- course registration.

### Online Courses:

- browsing through the course materials,
- downloading notes and slides,
- online class and lab interactions,
- uploading answers to assignments.

# CGI Programming

## web interfaces

### Course Administration:

- browsing of the course catalog,
- answering queries about courses,
- course registration.

### Online Courses:

- browsing through the course materials,
- downloading notes and slides,
- online class and lab interactions,
- uploading answers to assignments.

# CGI Programming

web interfaces

## Course Administration:

- browsing of the course catalog,
- answering queries about courses,
- course registration.

## Online Courses:

- browsing through the course materials,
- downloading notes and slides,
- online class and lab interactions,
- uploading answers to assignments.

# CGI Programming

## web interfaces

### Course Administration:

- browsing of the course catalog,
- answering queries about courses,
- course registration.

### Online Courses:

- browsing through the course materials,
- downloading notes and slides,
- online class and lab interactions,
- uploading answers to assignments.

# Information Management

with databases

Database to administer courses has three tables:

- 1 students: information about students,
- 2 courses: prerequisites, description, ...,
- 3 enrollment: links students with courses.

Every course has its own database, for

- detailed syllabus,
- assignments, notes and slides,
- administration of grades.

# Information Management

with databases

Database to administer courses has three tables:

- 1 students: information about students,
- 2 courses: prerequisites, description, ...,
- 3 enrollment: links students with courses.

Every course has its own database, for

- detailed syllabus,
- assignments, notes and slides,
- administration of grades.

# Information Management

with databases

Database to administer courses has three tables:

- 1 students: information about students,
- 2 courses: prerequisites, description, ...,
- 3 enrollment: links students with courses.

Every course has its own database, for

- detailed syllabus,
- assignments, notes and slides,
- administration of grades.

# Information Management

with databases

Database to administer courses has three tables:

- 1 students: information about students,
- 2 courses: prerequisites, description, ...,
- 3 enrollment: links students with courses.

Every course has its own database, for

- detailed syllabus,
- assignments, notes and slides,
- administration of grades.

# Information Management

with databases

Database to administer courses has three tables:

- 1 students: information about students,
- 2 courses: prerequisites, description, ...,
- 3 enrollment: links students with courses.

Every course has its own database, for

- detailed syllabus,
- assignments, notes and slides,
- administration of grades.

# Information Management

with databases

Database to administer courses has three tables:

- 1 students: information about students,
- 2 courses: prerequisites, description, ...,
- 3 enrollment: links students with courses.

Every course has its own database, for

- detailed syllabus,
- assignments, notes and slides,
- administration of grades.

## A Virtual University

description of an application

**the four pillars in the application**

## Uploading Files

form of the HTML code to upload processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# Multiple Servers

to handle the load

## Multiple computers to

- handle online registration,
- manage running of courses,
- backup essential data.

We expect our servers to be multifunctional:

- peak periods for registration,
- prime time for online courses.

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# Multiple Servers

to handle the load

## Multiple computers to

- handle online registration,
- manage running of courses,
- backup essential data.

We expect our servers to be multifunctional:

- peak periods for registration,
- prime time for online courses.

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# Multiple Servers

to handle the load

## Multiple computers to

- handle online registration,
- manage running of courses,
- backup essential data.

We expect our servers to be multifunctional:

- peak periods for registration,
- prime time for online courses.

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# Multiple Servers

to handle the load

## Multiple computers to

- handle online registration,
- manage running of courses,
- backup essential data.

## We expect our servers to be multifunctional:

- peak periods for registration,
- prime time for online courses.

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# Multiple Servers

to handle the load

## Multiple computers to

- handle online registration,
- manage running of courses,
- backup essential data.

## We expect our servers to be multifunctional:

- peak periods for registration,
- prime time for online courses.

# Multithreaded Servers

to handle the load

All servers are multithreaded:

- 1 handle indefinite number of requests,
- 2 for an indefinite time.

Distributed computing over multiple computers:  
→ load balancing and rescheduling of requests.

# Multithreaded Servers

to handle the load

All servers are multithreaded:

- 1 handle indefinite number of requests,
- 2 for an indefinite time.

Distributed computing over multiple computers:  
→ load balancing and rescheduling of requests.

# Multithreaded Servers

to handle the load

All servers are multithreaded:

- 1 handle indefinite number of requests,
- 2 for an indefinite time.

Distributed computing over multiple computers:  
→ load balancing and rescheduling of requests.

# an application

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

**form of the HTML code to upload**  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

### 1 A Virtual University

description of an application  
the four pillars in the application

### 2 Uploading Files

**form of the HTML code to upload**  
processing uploaded file with CGI script

### 3 Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

# Uploading Files

via an HTML form

Answers to assignments will be uploaded.

We must specify the encoding of the form adding

```
enctype = "multipart/form-data"
```

as one of the tags.

We then use an input element of type `file`.

For example:

```
<input type="file" name="upfile" size="50">
```

# Uploading Files

via an HTML form

Answers to assignments will be uploaded.

We must specify the encoding of the form adding

```
enctype = "multipart/form-data"
```

as one of the tags.

We then use an input element of type `file`.

For example:

```
<input type="file" name="upfile" size="50">
```

# Uploading Files

via an HTML form

Answers to assignments will be uploaded.

We must specify the encoding of the form adding

```
enctype = "multipart/form-data"
```

as one of the tags.

We then use an input element of type `file`.

For example:

```
<input type="file" name="upfile" size="50">
```

# HTML code

in the file `uploadfile.html`

```

<html>
<head>
<title> MCS 275 Lec 36: uploading a file </title>
</head>
<body>

<h1> form to upload a file </h1>

<form method="post"
        action="http://localhost/cgi-bin/uploadfile.p
        enctype="multipart/form-data">

<input type="file" name="upfile" size = "50">

<p> <input type="submit" value="submit your file">
    <input type="reset" value="cancel selection"> <

</form>
</body>
</html>

```

# HTML code

in the file `uploadfile.html`

```

<html>
<head>
<title> MCS 275 Lec 36: uploading a file </title>
</head>
<body>

<h1> form to upload a file </h1>

<form method="post"
      action="http://localhost/cgi-bin/uploadfile.p
      enctype="multipart/form-data">

<input type="file" name="upfile" size = "50">

<p> <input type="submit" value="submit your file">
    <input type="reset" value="cancel selection"> <

</form>
</body>
</html>

```

# HTML code

in the file `uploadfile.html`

```
<html>
<head>
<title> MCS 275 Lec 36: uploading a file </title>
</head>
<body>

<h1> form to upload a file </h1>

<form method="post"
        action="http://localhost/cgi-bin/uploadfile.p
        enctype="multipart/form-data">

<input type="file" name="upfile" size = "50">

<p> <input type="submit" value="submit your file">
    <input type="reset" value="cancel selection"> <
</form>
</body>
</html>
```

# HTML code

in the file `uploadfile.html`

```
<html>
<head>
<title> MCS 275 Lec 36: uploading a file </title>
</head>
<body>

<h1> form to upload a file </h1>

<form method="post"
        action="http://localhost/cgi-bin/uploadfile.p
        enctype="multipart/form-data">

<input type="file" name="upfile" size = "50">

<p> <input type="submit" value="submit your file">
    <input type="reset" value="cancel selection"> <
</form>
</body>
</html>
```

# an application

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

### 1 A Virtual University

description of an application  
the four pillars in the application

### 2 Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

### 3 Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

# Processing the Uploaded File

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
**processing uploaded file with CGI script**

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

name field of the input element has value 'upfile'.

In CGI script:

```
form = cgi.FieldStorage
```

Use the key name through its value as defined in the form to access the file:

```
uploaded = form['upfile']
```

Use the file attribute of `uploaded` for reading:

```
line = uploaded.file.readline()
```

# Processing the Uploaded File

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the HTMLParser module  
overriding methods in HTMLParser  
filtering attributes of tags

name field of the input element has value 'upfile'.

In CGI script:

```
form = cgi.FieldStorage
```

Use the key name through its value as defined in the form to access the file:

```
uploaded = form['upfile']
```

Use the file attribute of uploaded for reading:

```
line = uploaded.file.readline()
```

# Processing the Uploaded File

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the HTMLParser module  
overriding methods in HTMLParser  
filtering attributes of tags

name field of the input element has value 'upfile'.

In CGI script:

```
form = cgi.FieldStorage
```

Use the key name through its value as defined in the form to access the file:

```
uploaded = form['upfile']
```

Use the file attribute of `uploaded` for reading:

```
line = uploaded.file.readline()
```

# Processing the Uploaded File

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the HTMLParser module  
overriding methods in HTMLParser  
filtering attributes of tags

name field of the input element has value 'upfile'.

In CGI script:

```
form = cgi.FieldStorage
```

Use the key name through its value as defined in the form to access the file:

```
uploaded = form['upfile']
```

Use the file attribute of uploaded for reading:

```
line = uploaded.file.readline()
```

# Script to see Uploaded File

in the file `uploadfile.py`

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

```
#!/Library/Frameworks/./bin/python
# L-36 MCS 275 Mon 12 Apr 2010 : uploadfile.py

# This CGI script takes the input of the form
# uploadfile.html and writes the first line of
# the file in plaintext on the web page.

import cgi
form = cgi.FieldStorage()

print "Content-Type: text/plain\n"

uploaded = form['upfile']

line = uploaded.file.readline()
print line
```

# Script to see Uploaded File

in the file `uploadfile.py`

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

```
#!/Library/Frameworks/./bin/python
# L-36 MCS 275 Mon 12 Apr 2010 : uploadfile.py

# This CGI script takes the input of the form
# uploadfile.html and writes the first line of
# the file in plaintext on the web page.

import cgi
form = cgi.FieldStorage()

print "Content-Type: text/plain\n"

uploaded = form['upfile']

line = uploaded.file.readline()
print line
```

# Script to see Uploaded File

in the file `uploadfile.py`

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

```
#!/Library/Frameworks/./bin/python
# L-36 MCS 275 Mon 12 Apr 2010 : uploadfile.py

# This CGI script takes the input of the form
# uploadfile.html and writes the first line of
# the file in plaintext on the web page.

import cgi
form = cgi.FieldStorage()

print "Content-Type: text/plain\n"

uploaded = form['upfile']

line = uploaded.file.readline()
print line
```

12 Apr 2010

# Script to see Uploaded File

in the file `uploadfile.py`

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` moduleoverriding methods in `HTMLParser`

filtering attributes of tags

```
#!/Library/Frameworks/./bin/python
# L-36 MCS 275 Mon 12 Apr 2010 : uploadfile.py

# This CGI script takes the input of the form
# uploadfile.html and writes the first line of
# the file in plaintext on the web page.

import cgi
form = cgi.FieldStorage()

print "Content-Type: text/plain\n"

uploaded = form['upfile']

line = uploaded.file.readline()
print line
```

# an application

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

### grabbing course offerings

the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

## 1 A Virtual University

description of an application  
the four pillars in the application

## 2 Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## 3 Course Listings at UIC

**grabbing course offerings**  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UIC**grabbing course  
offerings**the `HTMLParser`  
moduleoverriding methods  
in `HTMLParser`filtering attributes of  
tags

# Storing Web Data

One quick way to create a database of considerable size is to load it with data available on web pages.

An application: a database of courses at UIC

- data is already structured
- readily available at UIC's web pages

Stages in this project:

- 1 grab the data from the web into a file
- 2 format data on file into data tuples
- 3 insert data tuples into a database.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offeringsthe `HTMLParser`  
moduleoverriding methods  
in `HTMLParser`filtering attributes of  
tags

# Storing Web Data

One quick way to create a database of considerable size is to load it with data available on web pages.

An application: a database of courses at UIC

- data is already structured
- readily available at UIC's web pages

Stages in this project:

- 1 grab the data from the web into a file
- 2 format data on file into data tuples
- 3 insert data tuples into a database.

# Storing Web Data

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

One quick way to create a database of considerable size is to load it with data available on web pages.

An application: a database of courses at UIC

- data is already structured
- readily available at UIC's web pages

Stages in this project:

- 1 grab the data from the web into a file
- 2 format data on file into data tuples
- 3 insert data tuples into a database.

# Storing Web Data

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

One quick way to create a database of considerable size is to load it with data available on web pages.

An application: a database of courses at UIC

- data is already structured
- readily available at UIC's web pages

Stages in this project:

- 1 grab the data from the web into a file
- 2 format data on file into data tuples
- 3 insert data tuples into a database.

# Storing Web Data

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

One quick way to create a database of considerable size is to load it with data available on web pages.

An application: a database of courses at UIC

- data is already structured
- readily available at UIC's web pages

Stages in this project:

- 1 grab the data from the web into a file
- 2 format data on file into data tuples
- 3 insert data tuples into a database.

# Running courselistings.py

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML

code to upload

processing uploaded

file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```
$ python courselistings.py
give subject (upper case) : LAT
Spring, Fall, or Summer (0,1,2) : 0
which year (4 digits) : 2010
opening http://www.uic.edu/depts/ims/classschedule/
S2010/LAT.htm ...
courses on http://www.uic.edu/depts/ims/classsched
ule/S2010/LAT.htm :
LAT102 Elementary Latin II
LAT103 Intermediate Latin I
LAT104 Intermediate Latin II
LAT299 Independent Reading
LAT499 Independent Reading
```

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

**grabbing course offerings**

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

# Online Data

The base of the URL is

```
http://www.uic.edu/depts/ims/classschedule/
```

followed by `S`, `SUM`, or `F`

for spring, summer, or fall semester,

followed by year as 4-digit number, e.g. 2010,

followed by the subject, e.g.: `LAT`.

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

The base of the URL is

`http://www.uic.edu/depts/ims/classschedule/`

followed by `S`, `SUM`, or `F`  
for spring, summer, or fall semester,

followed by year as 4-digit number, e.g. 2010,

followed by the subject, e.g.: `LAT`.

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

The base of the URL is

`http://www.uic.edu/depts/ims/classschedule/`

followed by `S`, `SUM`, or `F`  
for spring, summer, or fall semester,

followed by year as 4-digit number, e.g. 2010,

followed by the subject, e.g.: `LAT`.

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

The base of the URL is

```
http://www.uic.edu/depts/ims/classschedule/
```

followed by S, SUM, or F

for spring, summer, or fall semester,

followed by year as 4-digit number, e.g. 2010,

followed by the subject, e.g.: LAT.

# an application

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
**the HTMLParser module**  
overriding methods in HTMLParser  
filtering attributes of tags

### 1 A Virtual University

description of an application  
the four pillars in the application

### 2 Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

### 3 Course Listings at UIC

grabbing course offerings  
**the HTMLParser module**  
overriding methods in HTMLParser  
filtering attributes of tags

# the `HTMLParser` module

to parse html code

In the standard Python distribution:

```
>>> import HTMLParser
>>> help(HTMLParser)
```

the class `HTMLParser`

- allows to override handlers of tags
- provides a `feed` method

→ the `feed` method handles buffering

# Using HTMLParser

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

### the HTMLParser module

overriding methods in HTMLParser  
filtering attributes of tags

```
from HTMLParser import HTMLParser
from urllib import urlopen
```

```
class OurHTMLParser(HTMLParser):
    def __init__(self):
    def handle_starttag(self, tag, attrs):
    def handle_endtag(self, tag):

def main():
    f = urlopen(page)
    p = OurHTMLParser()
    while True:
        data = f.read(80)
        if data == '': break
        p.feed(data)
    p.close()
```

# Using HTMLParser

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

### the HTMLParser module

overriding methods in HTMLParser  
filtering attributes of tags

```
from HTMLParser import HTMLParser
from urllib import urlopen

class OurHTMLParser(HTMLParser):
    def __init__(self):
    def handle_starttag(self, tag, attrs):
    def handle_endtag(self, tag):

def main():
    f = urlopen(page)
    p = OurHTMLParser()
    while True:
        data = f.read(80)
        if data == '': break
        p.feed(data)
    p.close()
```

# Using HTMLParser

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

### the HTMLParser module

overriding methods in HTMLParser  
filtering attributes of tags

```
from HTMLParser import HTMLParser
from urllib import urlopen

class OurHTMLParser(HTMLParser):
    def __init__(self):
    def handle_starttag(self, tag, attrs):
    def handle_endtag(self, tag):

def main():
    f = urlopen(page)
    p = OurHTMLParser()
    while True:
        data = f.read(80)
        if data == '': break
        p.feed(data)
    p.close()
```

# Using HTMLParser

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

### the HTMLParser module

overriding methods in HTMLParser  
filtering attributes of tags

```
from HTMLParser import HTMLParser
from urllib import urlopen

class OurHTMLParser(HTMLParser):
    def __init__(self):
        def handle_starttag(self, tag, attrs):
        def handle_endtag(self, tag):

def main():
    f = urlopen(page)
    p = OurHTMLParser()
    while True:
        data = f.read(80)
        if data == '': break
        p.feed(data)
    p.close()
```

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offerings**the HTMLParser  
module**overriding methods  
in HTMLParserfiltering attributes of  
tags

# Tallying the Tags

using the class HTMLParser

Gather basic statistics about a page:

- 1 what types of tags are used,
- 2 count number of occurrences for each tag.

At end of each tag the tally is updated.

Data structure for the tally: dictionary.

- keys: string with type of tag
- values: natural number counts #occurrences

The tally is an object data attribute.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offerings**the HTMLParser  
module**overriding methods  
in HTMLParserfiltering attributes of  
tags

# Tallying the Tags

using the class HTMLParser

Gather basic statistics about a page:

- 1 what types of tags are used,
- 2 count number of occurrences for each tag.

At end of each tag the tally is updated.

Data structure for the tally: dictionary.

- keys: string with type of tag
- values: natural number counts #occurrences

The tally is an object data attribute.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offerings**the HTMLParser  
module**overriding methods  
in HTMLParserfiltering attributes of  
tags

# Tallying the Tags

using the class HTMLParser

Gather basic statistics about a page:

- 1 what types of tags are used,
- 2 count number of occurrences for each tag.

At end of each tag the tally is updated.

Data structure for the tally: dictionary.

- keys: string with type of tag
- values: natural number counts #occurrences

The tally is an object data attribute.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offeringsthe `HTMLParser`  
moduleoverriding methods  
in `HTMLParser`filtering attributes of  
tags

# Tallying the Tags

using the class `HTMLParser`

Gather basic statistics about a page:

- 1 what types of tags are used,
- 2 count number of occurrences for each tag.

At end of each tag the tally is updated.

Data structure for the tally: dictionary.

- keys: string with type of tag
- values: natural number counts `#occurrences`

The tally is an object data attribute.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offeringsthe `HTMLParser`  
moduleoverriding methods  
in `HTMLParser`filtering attributes of  
tags

# Tallying the Tags

using the class `HTMLParser`

Gather basic statistics about a page:

- 1 what types of tags are used,
- 2 count number of occurrences for each tag.

At end of each tag the tally is updated.

Data structure for the tally: dictionary.

- keys: string with type of tag
- values: natural number counts `#occurrences`

The tally is an object data attribute.

A Virtual  
University

description of an  
application  
the four pillars in the  
application

Uploading  
Files

form of the HTML  
code to upload  
processing uploaded  
file with CGI script

Course  
Listings at UIC

grabbing course  
offerings  
**the HTMLParser  
module**  
overriding methods  
in HTMLParser  
filtering attributes of  
tags

# Tallying the Tags

using the class HTMLParser

Gather basic statistics about a page:

- 1 what types of tags are used,
- 2 count number of occurrences for each tag.

At end of each tag the tally is updated.

Data structure for the tally: dictionary.

- keys: string with type of tag
- values: natural number counts #occurrences

The tally is an object data attribute.

A Virtual  
Universitydescription of an  
applicationthe four pillars in the  
applicationUploading  
Filesform of the HTML  
code to uploadprocessing uploaded  
file with CGI scriptCourse  
Listings at UICgrabbing course  
offeringsthe `HTMLParser`  
moduleoverriding methods  
in `HTMLParser`filtering attributes of  
tags

# Tallying the Tags

using the class `HTMLParser`

Gather basic statistics about a page:

- 1 what types of tags are used,
- 2 count number of occurrences for each tag.

At end of each tag the tally is updated.

Data structure for the tally: dictionary.

- keys: string with type of tag
- values: natural number counts `#occurrences`

The tally is an object data attribute.

12 Apr 2010

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

# an application

- 1 A Virtual University  
description of an application  
the four pillars in the application
- 2 Uploading Files  
form of the HTML code to upload  
processing uploaded file with CGI script
- 3 **Course Listings at UIC**  
grabbing course offerings  
the `HTMLParser` module  
**overriding methods in `HTMLParser`**  
filtering attributes of tags

# the Class TagTally

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class TagTally(HTMLParser):
    """
    Makes a tally of ending tags.
    """
    def __init__(self):
        """
        Initializes the dictionary of tags.
        """
    def handle_endtag(self, tag):
        """
        Maintains a tally of the tags.
        """
    def ShowTally(self):
        """
        Prints the tally to screen.
        """

```

# the Class TagTally

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class TagTally(HTMLParser):
    """
    Makes a tally of ending tags.
    """
    def __init__(self):
        """
        Initializes the dictionary of tags.
        """
    def handle_endtag(self, tag):
        """
        Maintains a tally of the tags.
        """
    def ShowTally(self):
        """
        Prints the tally to screen.
        """

```

# the Class TagTally

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class TagTally(HTMLParser):
    """
    Makes a tally of ending tags.
    """
    def __init__(self):
        """
        Initializes the dictionary of tags.
        """
    def handle_endtag(self, tag):
        """
        Maintains a tally of the tags.
        """
    def ShowTally(self):
        """
        Prints the tally to screen.
        """

```

# the Class TagTally

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class TagTally(HTMLParser):
    """
    Makes a tally of ending tags.
    """
    def __init__(self):
        """
        Initializes the dictionary of tags.
        """
    def handle_endtag(self, tag):
        """
        Maintains a tally of the tags.
        """
    def ShowTally(self):
        """
        Prints the tally to screen.
        """

```

# the Class TagTally

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class TagTally(HTMLParser):
    """
    Makes a tally of ending tags.
    """
    def __init__(self):
        """
        Initializes the dictionary of tags.
        """
    def handle_endtag(self, tag):
        """
        Maintains a tally of the tags.
        """
    def ShowTally(self):
        """
        Prints the tally to screen.
        """

```

# the Function `main()`

of script `tallytags.py`

```
def main():  
    """  
    Opens a web page and parses it.  
    """  
    page = 'http://www.uic.edu/'  
    print 'opening %s ...' % page  
    f = urlopen(page)  
    p = TagTally()  
    while True:  
        data = f.read(80)  
        if data == '': break  
        p.feed(data)  
    p.close()  
    print 'the tally of tags :'  
    p.ShowTally()
```

# the Function `main()`

of script `tallytags.py`

```
def main():  
    """  
    Opens a web page and parses it.  
    """  
    page = 'http://www.uic.edu/'  
    print 'opening %s ...' % page  
    f = urlopen(page)  
    p = TagTally()  
    while True:  
        data = f.read(80)  
        if data == '': break  
        p.feed(data)  
    p.close()  
    print 'the tally of tags :'  
    p.ShowTally()
```

the Function `main()`of script `tallytags.py`

```
def main():  
    """  
    Opens a web page and parses it.  
    """  
    page = 'http://www.uic.edu/'  
    print 'opening %s ...' % page  
    f = urlopen(page)  
    p = TagTally()  
    while True:  
        data = f.read(80)  
        if data == '': break  
        p.feed(data)  
    p.close()  
    print 'the tally of tags :'  
    p.ShowTally()
```

12 Apr 2010

A Virtual  
University

description of an  
application  
the four pillars in the  
application

Uploading  
Files

form of the HTML  
code to upload  
processing uploaded  
file with CGI script

Course  
Listings at UIC

grabbing course  
offerings  
the `HTMLParser`  
module  
overriding methods  
in `HTMLParser`  
filtering attributes of  
tags

# Constructor and ShowTally

of the class `TagTally`

If overriding `__init__`, we must initialize parent class.

```
def __init__(self):
    """
    Initializes the dictionary of tags.
    """
    HTMLParser.__init__(self)
    self.TagTally = {}

def ShowTally(self):
    """
    Prints the tally to screen.
    """
    for each in self.TagTally:
        print each, ':', self.TagTally[each]
```

12 Apr 2010

A Virtual  
University

description of an  
application  
the four pillars in the  
application

Uploading  
Files

form of the HTML  
code to upload  
processing uploaded  
file with CGI script

Course  
Listings at UIC

grabbing course  
offerings  
the `HTMLParser`  
module  
overriding methods  
in `HTMLParser`  
filtering attributes of  
tags

# Constructor and ShowTally

of the class `TagTally`

If overriding `__init__`, we must initialize parent class.

```
def __init__(self):
    """
    Initializes the dictionary of tags.
    """
    HTMLParser.__init__(self)
    self.TagTally = {}

def ShowTally(self):
    """
    Prints the tally to screen.
    """
    for each in self.TagTally:
        print each, ':', self.TagTally[each]
```

# Constructor and ShowTally

of the class TagTally

If overriding `__init__`, we must initialize parent class.

```
def __init__(self):
    """
    Initializes the dictionary of tags.
    """
    HTMLParser.__init__(self)
    self.TagTally = {}

def ShowTally(self):
    """
    Prints the tally to screen.
    """
    for each in self.TagTally:
        print each, ':', self.TagTally[each]
```

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the HTMLParser module  
overriding methods in HTMLParser  
filtering attributes of tags

# Update of the Tally

updating a dictionary

First check if there is already a tag...

```
def handle_endtag(self, tag):  
    """  
    Maintains a tally of the tags.  
    """  
    if self.TagTally.has_key(tag):  
        self.TagTally[tag] \  
            = self.TagTally[tag] + 1  
    else:  
        self.TagTally.update({tag:1})
```

If no tag present, update the dictionary.

# Update of the Tally

updating a dictionary

First check if there is already a tag...

```
def handle_endtag(self, tag):  
    """  
    Maintains a tally of the tags.  
    """  
    if self.TagTally.has_key(tag):  
        self.TagTally[tag] \  
            = self.TagTally[tag] + 1  
    else:  
        self.TagTally.update({tag:1})
```

If no tag present, update the dictionary.

# Update of the Tally

updating a dictionary

First check if there is already a tag...

```
def handle_endtag(self, tag):  
    """  
    Maintains a tally of the tags.  
    """  
    if self.TagTally.has_key(tag):  
        self.TagTally[tag] \  
            = self.TagTally[tag] + 1  
    else:  
        self.TagTally.update({tag:1})
```

If no tag present, update the dictionary.

# Update of the Tally

updating a dictionary

First check if there is already a tag...

```
def handle_endtag(self, tag):  
    """  
    Maintains a tally of the tags.  
    """  
    if self.TagTally.has_key(tag):  
        self.TagTally[tag] \  
            = self.TagTally[tag] + 1  
    else:  
        self.TagTally.update({tag:1})
```

If no tag present, update the dictionary.

# an application

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

### 1 A Virtual University

description of an application  
the four pillars in the application

### 2 Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

### 3 Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
**filtering attributes of tags**

# Get Links a Page refers to

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

Recall our code for a web crawler:

- double quoted strings starting with `http` could be misleading at times, cumbersome code.

A more proper way to get the hyperlinks:

- 1 look for tags of type `'a'`
- 2 name of attribute is `href`
- 3 get hyperlink corresponding to `href`

# Get Links a Page refers to

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

Recall our code for a web crawler:

- double quoted strings starting with `http` could be misleading at times, cumbersome code.

A more proper way to get the hyperlinks:

- 1 look for tags of type `'a'`
- 2 name of attribute is `href`
- 3 get hyperlink corresponding to `href`

# Get Links a Page refers to

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

Recall our code for a web crawler:

- double quoted strings starting with `http` could be misleading at times, cumbersome code.

A more proper way to get the hyperlinks:

- 1 look for tags of type `'a'`
- 2 name of attribute is `href`
- 3 get hyperlink corresponding to `href`

# Get Links a Page refers to

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

Recall our code for a web crawler:

- double quoted strings starting with `http` could be misleading at times, cumbersome code.

A more proper way to get the hyperlinks:

- 1 look for tags of type `'a'`
- 2 name of attribute is `href`
- 3 get hyperlink corresponding to `href`

# the Class HTMLRefs

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class HTMLRefs(HTMLParser):
    """
    Makes a list of all html links.
    """
    def __init__(self):
        """
        Initializes the list of links.
        """
    def handle_starttag(self, tag, attrs):
        """
        Looks for tags equal to 'a' and
        stores links for href attributes.
        """
    def ShowRefs(self):
        """
        Prints the HTML refs to screen.
        """

```

# the Class HTMLRefs

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class HTMLRefs(HTMLParser):
    """
    Makes a list of all html links.
    """
    def __init__(self):
        """
        Initializes the list of links.
        """
    def handle_starttag(self, tag, attrs):
        """
        Looks for tags equal to 'a' and
        stores links for href attributes.
        """
    def ShowRefs(self):
        """
        Prints the HTML refs to screen.
        """

```

# the Class HTMLRefs

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class HTMLRefs(HTMLParser):
    """
    Makes a list of all html links.
    """
    def __init__(self):
        """
        Initializes the list of links.
        """
    def handle_starttag(self, tag, attrs):
        """
        Looks for tags equal to 'a' and
        stores links for href attributes.
        """
    def ShowRefs(self):
        """
        Prints the HTML refs to screen.
        """

```

# the Class HTMLRefs

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

```

from HTMLParser import HTMLParser
from urllib import urlopen

class HTMLRefs(HTMLParser):
    """
    Makes a list of all html links.
    """
    def __init__(self):
        """
        Initializes the list of links.
        """
    def handle_starttag(self, tag, attrs):
        """
        Looks for tags equal to 'a' and
        stores links for href attributes.
        """
    def ShowRefs(self):
        """
        Prints the HTML refs to screen.
        """

```

# the Function `main()`

of the script `htmlrefs.py`

```
def main():  
    """  
    Opens a web page and parses it.  
    """  
    page = 'http://www.uic.edu/'  
    print 'opening %s ...' % page  
    f = urlopen(page)  
    p = HTMLRefs()  
    while True:  
        data = f.read(80)  
        if data == '': break  
        p.feed(data)  
    p.close()  
    print 'all html links :'  
    p.ShowRefs()
```

# the Function `main()`

of the script `htmlrefs.py`

```
def main():  
    """  
    Opens a web page and parses it.  
    """  
    page = 'http://www.uic.edu/'  
    print 'opening %s ...' % page  
    f = urlopen(page)  
    p = HTMLRefs()  
    while True:  
        data = f.read(80)  
        if data == '': break  
        p.feed(data)  
    p.close()  
    print 'all html links :'  
    p.ShowRefs()
```

# the Function `main()`

of the script `htmlrefs.py`

```
def main():  
    """  
    Opens a web page and parses it.  
    """  
    page = 'http://www.uic.edu/'  
    print 'opening %s ...' % page  
    f = urlopen(page)  
    p = HTMLRefs()  
    while True:  
        data = f.read(80)  
        if data == '': break  
        p.feed(data)  
    p.close()  
    print 'all html links :'  
    p.ShowRefs()
```

# Constructor and ShowRefs

of the class `HTMLRefs`

We now use a list as object data attribute.

```

def __init__(self):
    """
    Initializes the list of links.
    """
    HTMLParser.__init__(self)
    self.refs = []

def ShowRefs(self):
    """
    Prints the HTML refs to screen.
    """
    for each in self.refs:
        print each

```

# Constructor and ShowRefs

of the class `HTMLRefs`

We now use a list as object data attribute.

```
def __init__(self):
    """
    Initializes the list of links.
    """
    HTMLParser.__init__(self)
    self.refs = []

def ShowRefs(self):
    """
    Prints the HTML refs to screen.
    """
    for each in self.refs:
        print each
```

# Constructor and ShowRefs

of the class `HTMLRefs`

We now use a list as object data attribute.

```
def __init__(self):
    """
    Initializes the list of links.
    """
    HTMLParser.__init__(self)
    self.refs = []

def ShowRefs(self):
    """
    Prints the HTML refs to screen.
    """
    for each in self.refs:
        print each
```

A Virtual  
University

description of an  
application  
the four pillars in the  
application

Uploading  
Files

form of the HTML  
code to upload  
processing uploaded  
file with CGI script

Course  
Listings at UIC

grabbing course  
offerings  
the `HTMLParser`  
module  
overriding methods  
in `HTMLParser`  
filtering attributes of  
tags

12 Apr 2010

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings

the HTMLParser module

overriding methods in HTMLParser

filtering attributes of tags

## Filtering Attributes

lambda expression, filter, and list comprehension

Attributes are lists of tuples: [ ( . , . ) , ( . , . ) , . . . ]

e.g.: [ ( 'href' , 'learning.shtml' ) , . . . ]

→ link is the *y* in a (*x*,*y*) tuple

def handle\_starttag(self, tag, attrs):

"""

Looks for tags equal to 'a' and stores links for href attributes.

"""

if tag == 'a':

f = lambda (x,y): x == 'href'

F = filter(f,attrs)

L = [ y for (x,y) in F ]

self.refs = self.refs + L

12 Apr 2010

# Filtering Attributes

lambda expression, filter, and list comprehension

Attributes are lists of tuples: `[ ( . , . ) , ( . , . ) , ... ]`

e.g.: `[ ( 'href' , 'learning.shtml' ) , ... ]`

→ link is the *y* in a  $(x,y)$  tuple

```
def handle_starttag(self, tag, attrs):
```

```
    """
```

```
    Looks for tags equal to 'a' and
    stores links for href attributes.
```

```
    """
```

```
    if tag == 'a':
```

```
        f = lambda (x,y): x == 'href'
```

```
        F = filter(f,attrs)
```

```
        L = [ y for (x,y) in F ]
```

```
        self.refs = self.refs + L
```

## A Virtual University

description of an application

the four pillars in the application

## Uploading Files

form of the HTML code to upload

processing uploaded file with CGI script

## Course

### Listings at UIC

grabbing course offerings

the `HTMLParser` module

overriding methods in `HTMLParser`

filtering attributes of tags

12 Apr 2010

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

## Filtering Attributes

lambda expression, filter, and list comprehension

Attributes are lists of tuples: `[ ( . , . ) , ( . , . ) , ... ]`e.g.: `[ ( 'href' , 'learning.shtml' ) , ... ]`→ link is the `y` in a `(x,y)` tuple

```
def handle_starttag(self, tag, attrs):
```

```
    """
```

```
    Looks for tags equal to 'a' and
    stores links for href attributes.
```

```
    """
```

```
    if tag == 'a':
```

```
        f = lambda (x,y): x == 'href'
```

```
        F = filter(f,attrs)
```

```
        L = [ y for (x,y) in F ]
```

```
        self.refs = self.refs + L
```

12 Apr 2010

## A Virtual University

description of an application  
the four pillars in the application

## Uploading Files

form of the HTML code to upload  
processing uploaded file with CGI script

## Course Listings at UIC

grabbing course offerings  
the `HTMLParser` module  
overriding methods in `HTMLParser`  
filtering attributes of tags

## Filtering Attributes

lambda expression, filter, and list comprehension

Attributes are lists of tuples: `[ ( . , . ) , ( . , . ) , ... ]`e.g.: `[ ( 'href' , 'learning.shtml' ) , ... ]`→ link is the `y` in a `(x,y)` tuple

```
def handle_starttag(self, tag, attrs):
```

```
    """
```

```
    Looks for tags equal to 'a' and
    stores links for href attributes.
```

```
    """
```

```
    if tag == 'a':
        f = lambda (x,y): x == 'href'
        F = filter(f,attrs)
        L = [ y for (x,y) in F ]
        self.refs = self.refs + L
```

## Summary + Assignments

`HTMLParser` is convenient to parse HTML

→ makes it easier to retrieve data from web

Assignments:

- 1 Use one script to upload files instead of using separate HTML code. Combine the HTML code and the code to process the uploaded file into functions `PrintForm()` and `ProcessFile()`.
- 2 Write a script that prompts the user for an URL and that finds the number of forms on the web page. The script should not crash when the page fails to open, but it should then display an error message.
- 3 Write a script to look for files with the extension `.py`.
- 4 Consider `webcrawler.py` of lecture 34. Use `HTMLParser` to write a shorter version.
- 5 Change class `BoldText` so all text formatted in bold is stored in a list in an object data attribute.