Outline

1 Internet Basics
   evolution of the web
   IP addresses and URLs
   client/server and HTTP

2 Markup Languages
   HTML, XML, MathML
   MathML generated by Maple

3 Retrieving Data
   the weather forecast

4 CGI Programming
   showing current date and time in browser

5 Summary + Assignments

MCS 260 Lecture 33
Introduction to Computer Science
Jan Verschelde, 10 November 2008
networking and the internet markup languages

1 Internet Basics
   evolution of the web
   IP addresses and URLs
   client/server and HTTP

2 Markup Languages
   HTML, XML, MathML
   MathML generated by Maple

3 Retrieving Data
   the weather forecast

4 CGI Programming
   showing current date and time in browser

5 Summary + Assignments
The World Wide Web
WWW: historical development


• mid 1980s: connections between computer facilities. In 1989, Tim Berners-Lee of CERN developed HTML, HyperText Markup Language.

• Mosaic was the first web browser developed at NCSA, released in 1993, leading to Netscape. Search engines originated at the end of the nineties.

• Web 2.0: publishing → participation. Wiki is server software that allows users to freely create and edit Web pages using any Web browser.
The World Wide Web
WWW: historical development

  Main benefit: capability of electronic mail (email).


- Mosaic was the first web browser developed at NCSA, released in 1993, leading to Netscape.
  Search engines originated at the end of the nineties.

- Web 2.0: publishing → participation. Wiki is server software that allows users to freely create and edit Web pages using any Web browser.
The World Wide Web
WWW: historical development


- Mosaic was the first web browser developed at NCSA, released in 1993, leading to Netscape. Search engines originated at the end of the nineties.

- Web 2.0: publishing → participation. *Wiki* is server software that allows users to freely create and edit Web pages using any Web browser.
The World Wide Web

WWW: historical development


- Mosaic was the first web browser developed at NCSA, released in 1993, leading to Netscape. Search engines originated at the end of the nineties.

- Web 2.0: publishing → participation. Wiki is server software that allows users to freely create and edit Web pages using any Web browser.
networking and the internet markup languages

1. Internet Basics
   - evolution of the web
   - IP addresses and URLs
   - client/server and HTTP

2. Markup Languages
   - HTML, XML, MathML
   - MathML generated by Maple

3. Retrieving Data
   - the weather forecast

4. CGI Programming
   - showing current date and time in browser

5. Summary + Assignments
Network Addresses
and domain names

Each node on the internet has an *IP address*. An IP address consists of four bytes.

Each node has a symbolic name. For example, `icarus.cc.uic.edu`. The `edu` stands for universities. The other principal domains are `gov` (government), `mil` (military), `com` (commercial), and `org` (other organizations).

The command `nslookup icarus.cc.uic.edu` or `host icarus.cc.uic.edu` returns the numerical IP address for `icarus`: `128.248.155.80`

The Internet is a *Wide Area Network* (WAN), linking machines over a greater distance. A *Local Area Network* (LAN), links computers in one room or building.
Network Addresses
and domain names

Each node on the internet has an *IP address*. An IP address consists of four bytes.

Each node has a symbolic name. For example, *icarus.cc.uic.edu*. The *edu* stands for universities. The other principal domains are *gov* (government), *mil* (military), *com* (commercial), and *org* (other organizations).

The command *nslookup icarus.cc.uic.edu* or *host icarus.cc.uic.edu* returns the numerical IP address for *icarus*: 128.248.155.80

The Internet is a *Wide Area Network* (WAN), linking machines over a greater distance. A *Local Area Network* (LAN), links computers in one room or building.
Network Addresses
and domain names

Each node on the internet has an *IP address*. An IP address consists of four bytes.

Each node has a symbolic name. For example, *icarus.cc.uic.edu*. The *edu* stands for universities. The other principal domains are *gov* (government), *mil* (military), *com* (commercial), and *org* (other organizations).

The command `nslookup icarus.cc.uic.edu` or `host icarus.cc.uic.edu returns` the numerical IP address for *icarus*: `128.248.155.80`

The Internet is a *Wide Area Network* (WAN), linking machines over a greater distance. A *Local Area Network* (LAN), links computers in one room or building.
Network Addresses
and domain names

Each node on the internet has an *IP address*. An IP address consists of four bytes.

Each node has a symbolic name. For example, *icarus.cc.uic.edu*. The *edu* stands for universities. The other principal domains are *gov* (government), *mil* (military), *com* (commercial), and *org* (other organizations).

The command `nslookup icarus.cc.uic.edu` or `host icarus.cc.uic.edu` returns the numerical IP address for *icarus*: 128.248.155.80

The Internet is a *Wide Area Network* (WAN), linking machines over a greater distance. A *Local Area Network* (LAN), links computers in one room or building.
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:


The format of a URL is

protocol://host.domain-name/path/dataname

where

- **protocol** refers to the type of protocol to be used
- **host** refers to the server where the resource is stored
- **domain-name** contains the name and type of the domain of the server
- **path/dataname** refers to the location of the data

To preview pages offline, use the protocol file.
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:


The format of a URL is

protocol://host.domain-name/path/dataname

where

- protocol refers to the type of protocol to be used
- host refers to the server where the resource is stored
- domain-name contains the name and type of the domain of the server
- path/dataname refers to the location of the data

To preview pages offline, use the protocol file.
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:


The format of a URL is

protocol://host.domain-name/path/dataname

where

- **protocol** refers to the type of protocol to be used
- **host** refers to the server where the resource is stored
- **domain-name** contains the name and type of the domain of the server
- **path/dataname** refers to the location of the data

To preview pages offline, use the protocol file.
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:

The format of a URL is
protocol://host.domain-name/path/dataname

where

- **protocol** refers to the type of protocol to be used
- **host** refers to the server where the resource is stored
- **domain-name** contains the name and type of the domain of the server
- **path/dataname** refers to the location of the data

To preview pages offline, use the protocol file.
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:

The format of a URL is
protocol://host.domain-name/path/dataname

where

- **protocol** refers to the type of protocol to be used
- **host** refers to the server where the resource is stored
- **domain-name** contains the name and type of the domain of the server
- **path/dataname** refers to the location of the data

To preview pages offline, use the protocol `file`. 
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:


The format of a URL is

protocol://host.domain-name/path/dataname

where

- **protocol** refers to the type of protocol to be used
- **host** refers to the server where the resource is stored
- **domain-name** contains the name and type of the domain of the server
- **path/dataname** refers to the location of the data

To preview pages offline, use the protocol **file**.
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:

The format of a URL is

protocol://host.domain-name/path/dataname

where

- **protocol** refers to the type of protocol to be used
- **host** refers to the server where the resouce is stored
- **domain-name** contains the name and type of the domain of the server
- **path/dataname** refers to the location of the data

To preview pages offline, use the protocol **file**.
Uniform Resource Locator (URL)

A URL is an addressing scheme to provide a path to an internet resource.

Example:

The format of a URL is
protocol://host.domain-name/path/dataname

where

- **protocol** refers to the type of protocol to be used
- **host** refers to the server where the resource is stored
- **domain-name** contains the name and type of the domain of the server
- **path/dataname** refers to the location of the data

To preview pages offline, use the protocol file.
networking and the internet markup languages

1 Internet Basics
   evolution of the web
   IP addresses and URLs
   client/server and HTTP

2 Markup Languages
   HTML, XML, MathML
   MathML generated by Maple

3 Retrieving Data
   the weather forecast

4 CGI Programming
   showing current date and time in browser

5 Summary + Assignments
A *client* is a computer in the network that *requests* for access to data and services from another computer.

A *server* is a computer in the network *receives and processes* requests from clients. Access permissions are determined by the server.

A *client/server network* consists of several computers connected in a network, acting as clients and/or servers. Client/server computing emerged in the nineties to distribute applications (such as database administration) over a network.
Client/Server Networks

A **client** is a computer in the network that *requests* for access to data and services from another computer.

A **server** is a computer in the network *receives and processes* requests from clients. Access permissions are determined by the server.

A **client/server network** consists of several computers connected in a network, acting as clients and/or servers. Client/server computing emerged in the nineties to distribute applications (such as database administration) over a network.
Client/Server Networks

A client is a computer in the network that requests for access to data and services from another computer.

A server is a computer in the network receives and processes requests from clients. Access permissions are determined by the server.

A client/server network consists of several computers connected in a network, acting as clients and/or servers.

Client/server computing emerged in the nineties to distribute applications (such as database administration) over a network.
Client/Server Networks

A *client* is a computer in the network that *requests* for access to data and services from another computer.

A *server* is a computer in the network *receives and processes* requests from clients. Access permissions are determined by the server.

A *client/server network* consists of several computers connected in a network, acting as clients and/or servers. Client/server computing emerged in the nineties to distribute applications (such as database administration) over a network.
Hypertext Transfer Protocol (HTTP)
exchange data between the client and the server

HTTP is based on request-response between a web browser (the client) and a web server.
A typical transaction between browser and server:

1. A TCP/IP connection is established between browser and server.
2. The browser sends a request for a web page.
3. The server locates the file and responds, sending the content of the requested web page.
4. The TCP/IP connection is closed.
Hypertext Transfer Protocol (HTTP)
exchange data between the client and the server

HTTP is based on request-response between a web browser (the client) and a web server. A typical transaction between browser and server:

1. A TCP/IP connection is established between browser and server.
2. The browser sends a request for a web page.
3. The server locates the file and responds, sending the content of the requested web page.
4. The TCP/IP connection is closed.
HTTP is based on request-response between a web browser (the client) and a web server. A typical transaction between browser and server:

1. A TCP/IP connection is established between browser and server.
2. The browser sends a request for a web page.
3. The server locates the file and responds, sending the content of the requested web page.
4. The TCP/IP connection is closed.
Hypertext Transfer Protocol (HTTP)

exchange data between the client and the server

HTTP is based on request-response between a web browser (the client) and a web server. A typical transaction between browser and server:

1. A TCP/IP connection is established between browser and server.
2. The browser sends a request for a web page.
3. The server locates the file and responds, sending the content of the requested web page.
4. The TCP/IP connection is closed.
Creating your own homepage

Use your icarus account. Another good reason to learn Unix!

Personal web pages are stored in the subdirectory public_html in your homedirectory.

Make sure this file is accessible, by setting permissions chmod a+r * of all files in public_html.

URLs will start as
http://www2.uic.edu/~your_netid.
Creating your own homepage

Use your icarus account. Another good reason to learn Unix!

Personal web pages are stored in the subdirectory public_html in your homedirectory.

Make sure this file is accessible, by setting permissions

```
chmod a+r * of all files in public_html.
```

URLs will start as

```
http://www2.uic.edu/~your_netid.
```
Creating your own homepage

Use your icarus account. Another good reason to learn Unix!

Personal web pages are stored in the subdirectory public_html in your homedirectory.

Make sure this file is accessible, by setting permissions chmod a+r * of all files in public_html.

URLs will start as

http://www2.uic.edu/~your_netid.
Creating your own homepage

Use your icarus account.  

Another good reason to learn Unix!

Personal web pages are stored in the subdirectory public_html in your homedirectory.

Make sure this file is accessible, by setting permissions
chmod a+r * of all files in public_html.

URLs will start as
http://www2.uic.edu/~your_netid.
networking and the internet markup languages

1. Internet Basics
   evolution of the web
   IP addresses and URLs
   client/server and HTTP

2. Markup Languages
   HTML, XML, MathML
   MathML generated by Maple

3. Retrieving Data
   the weather forecast

4. CGI Programming
   showing current date and time in browser

5. Summary + Assignments
To write web pages, we use

**HTML** hypertext markup language written to display information, the language in which web pages are written.

**XML** extensible markup language
XML is a widely supported open technology for describing data.

**MathML** mathematical markup language
MathML can display complex mathematical expressions. As it is created with XML, MathML is a so-called XML *vocabulary*.

The world wide web consortium ([http://www.w3c.org](http://www.w3c.org)) is a source for many protocol standards for the web.
To write web pages, we use

**HTML**  hypertext markup language  
written to display information, the language in which web pages are written.

**XML**  extensible markup language  
XML is a widely supported open technology for describing data.

**MathML**  mathematical markup language  
MathML can display complex mathematical expressions. As it is created with XML, MathML is a so-called XML *vocabulary*.

The world wide web consortium ([http://www.w3c.org](http://www.w3c.org)) is a source for many protocol standards for the web.
To write web pages, we use

**HTML** hypertext markup language written to display information, the language in which web pages are written.

**XML** extensible markup language XML is a widely supported open technology for describing data.

**MathML** mathematical markup language MathML can display complex mathematical expressions. As it is created with XML, MathML is a so-called *XML vocabulary*.

The world wide web consortium (http://www.w3c.org) is a source for many protocol standards for the web.
Our First Web Page

*a sample of HTML: hello_world.html*

```html
<HTML>
  <HEAD>
    <TITLE>MCS 260 Lec 34: hello world</TITLE>
  </HEAD>

  <BODY>
    <H1>Hello World!</H1>
    This is our <EM>first</EM> web page!
    <P>
    To go to <A HREF="http://www.uic.edu">UIC’s home page</A>, click on the underlined text.
  </BODY>
</HTML>
```
XML to exchange data

a sample of XML: hello_world.xml

XML focuses on data, not its formatting

```xml
<xml>
<head>our first xml example</head>
<body>
<article>
<title>hello world</title>
<date>14 November 2007</date>
<href>http://www.uic.edu</href>
<content>This is our first XML example.</content>
</article>
</body>
</xml>
```

A browser will display the document tree.

XML + HTML = XHTML
XML to exchange data

a sample of XML: `hello_world.xml`

XML focuses on data, not its formatting

```xml
<xml>
  <head>our first xml example</head>
  <body>
    <article>
      <title>hello world</title>
      <date>14 November 2007</date>
      <href>http://www.uic.edu</href>
      <content>This is our first XML example.</content>
    </article>
  </body>
</xml>
```

A browser will display the document tree.

XML + HTML = XHTML
XML to exchange data

A sample of XML: `hello_world.xml`

XML focuses on data, not its formatting

```xml
<xml>
<head>our first xml example</head>
<body>
<article>
<title>hello world</title>
<date>14 November 2007</date>
<href>http://www.uic.edu</href>
<content>This is our first XML example.</content>
</article>
</body>
</xml>
```

A browser will display *the document tree*.

XML + HTML = XHTML
XML to exchange data

a sample of XML: hello_world.xml

XML focuses on data, not its formatting

```xml
<xml>
  <head>our first xml example</head>
  <body>
    <article>
      <title>hello world</title>
      <date>14 November 2007</date>
      <href>http://www.uic.edu</href>
      <content>This is our first XML example.</content>
    </article>
  </body>
</xml>
```

A browser will display the document tree.

XML + HTML = XHTML
Document Object Model (DOM)

The tree structure for `article.xml`:

```
article
  └── title
      ├── date
      │    └── author
      │         └── firstName
      │            └── lastName
      └── summary
          └── contents
```
The tree structure for article.xml:

```
article
  └── title
  └── date
      ├── author
      │   └── firstName
      │       ├── summary
      │       └── lastName
      └── contents
```
The tree structure for `article.xml`:

```
article
- title
- date
- author
  - firstName
  - lastName
- summary
- contents
```
Document Object Model (DOM)

The tree structure for article.xml:

```
article
  └── title
  └── date
      └── author
          └── firstName
              └── summary
              └── contents
              └── lastName
```
Document Object Model (DOM)

The tree structure for article.xml:

```
article
  └── title
  └── date
  └── author
      └── firstName
      └── lastName
           └── summary
           └── contents
```
Document Object Model (DOM)

The tree structure for article.xml:

```
article
  ├─ title
  ├─ date
  │   └─ author
  │       └─ firstName
  │           └─ lastName
  └─ summary
      └─ contents
```
Document Object Model (DOM)

The tree structure for `article.xml`:

```
article
  └── title
  └── date
      └── title
      └── author
          └── firstName
          └── summary
              └── contents
              └── lastName
```
networking and the internet markup languages

1. Internet Basics
   - evolution of the web
   - IP addresses and URLs
   - client/server and HTTP

2. Markup Languages
   - HTML, XML, MathML
   - MathML generated by Maple

3. Retrieving Data
   - the weather forecast

4. CGI Programming
   - showing current date and time in browser

5. Summary + Assignments
Typesetting Mathematical Expressions
using Maple and MathML

Mathematical expressions like
\[ \int_{-1}^{10} \frac{e^{-x^2} \sin(20x)}{\sqrt{2 + x^8}} \, dx \]
are encoded with \(\text{\LaTeX}\) as
\[
\int_{-1}^{10} \frac{e^{-x^2} \sin(20 \times)}{\sqrt{2 + x^8}} \, dx
\]

With Maple (also to produce \(\text{\LaTeX}\) code):
> f := Int(exp(-x^2)*sin(20*x)/sqrt(2+x^8),x=-1..10);

Menu: File, Export As, choose HTML with MathML to generate MathML.
Typesetting Mathematical Expressions
using Maple and MathML

Mathematical expressions like

\[
\int_{-1}^{10} \frac{e^{-x^2} \sin(20x)}{\sqrt{2 + x^8}} \, dx
\]

are encoded with \texttt{\LaTeX} as

\[
\int_{-1}^{10} \frac{e^{-x^2} \sin(20 \times x)}{\sqrt{2 + x^8}} \, dx
\]

With Maple (also to produce \texttt{\LaTeX} code):

\[
> f := \text{Int}(\exp(-x^2) \times \sin(20 \times x) / \sqrt{2 + x^8}, x=-1..10);
\]

**Menu:** File, Export As, choose HTML with MathML to generate MathML.
MathML generated by Maple

\[
\text{encodes } f := \int_{-1}^{10}. 
\]
Internet Basics
- evolution of the web
- IP addresses and URLs
- client/server and HTTP

Markup Languages
- HTML, XML, MathML
- MathML generated by Maple

Retrieving Data
- the weather forecast

CGI Programming
- showing current date and time in browser

Summary + Assignments
Internet Basics
- evolution of the web
- IP addresses and URLs
- client/server and HTTP

Markup Languages
- HTML, XML, MathML
- MathML generated by Maple

Retrieving Data
- the weather forecast

CGI Programming
- showing current date and time in browser

Summary + Assignments

```
$ python forecast.py
opening http://weather.noaa.gov/pub/data/forecasts/state/il/ilz013.txt ...

<table>
<thead>
<tr>
<th>TODAY</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOV 10</td>
<td>NOV 11</td>
<td>NOV 12</td>
<td>NOV 13</td>
<td>NOV 14</td>
<td>NOV 15</td>
<td>NOV 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHICAGO</th>
<th>DOWNTOWN</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PTCLDY</td>
<td>MOCLDY</td>
<td>CLOUDY</td>
<td>MOCLDY</td>
<td>PTCLDY</td>
<td>MOCLDY</td>
<td>PTCLDY</td>
</tr>
<tr>
<td>/40</td>
<td>33/45</td>
<td>38/47</td>
<td>42/50</td>
<td>43/49</td>
<td>38/45</td>
<td>32/44</td>
</tr>
<tr>
<td>/10</td>
<td>10/40</td>
<td>70/40</td>
<td>20/30</td>
<td>30/30</td>
<td>30/20</td>
<td>10/10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHICAGO</th>
<th>OHARE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PTCLDY</td>
<td>MOCLDY</td>
<td>CLOUDY</td>
<td>MOCLDY</td>
<td>PTCLDY</td>
<td>MOCLDY</td>
<td>PTCLDY</td>
</tr>
<tr>
<td>/40</td>
<td>33/45</td>
<td>37/46</td>
<td>41/49</td>
<td>42/48</td>
<td>37/45</td>
<td>30/43</td>
</tr>
<tr>
<td>/10</td>
<td>10/40</td>
<td>70/40</td>
<td>20/30</td>
<td>30/30</td>
<td>30/20</td>
<td>10/10</td>
</tr>
</tbody>
</table>
```
Data from the Web

The module `urllib` exports `urlopen`, which returns a file-like object.

Template for retrieving data from web pages:

```python
from urllib import urlopen
url = 'internet address'
f = urlopen(url)
s = f.readline()
```

To get the weather forecast: [http://www.weather.gov/](http://www.weather.gov/)
National Oceanic and Atmospheric Administration's National Weather Service

The web site provides data for downloading: [http://weather.noaa.gov/pub/data/]
Data from the Web

The module `urllib` exports `urlopen`, `urlopen` returns a file like object.

Template for retrieving data from web pages:

```python
from urllib import urlopen
url = '<internet address>'
f = urlopen(url)
s = f.readline()
```

To get the weather forecast: http://www.weather.gov/
National Oceanic and Atmospheric Administration’s
National Weather Service
The web site provides data for downloading:
http://weather.noaa.gov/pub/data/
Data from the Web

The module `urllib` exports `urlopen`, `urlopen` returns a file like object.

Template for retrieving data from web pages:

```python
from urllib import urlopen
url = ' < internet address > '
f = urlopen(url)
s = f.readline()
```

To get the weather forecast: [http://www.weather.gov/](http://www.weather.gov/)
National Oceanic and Atmospheric Administration’s
National Weather Service
The web site provides data for downloading:
[http://weather.noaa.gov/pub/data/](http://weather.noaa.gov/pub/data/)
The Script `forecast.py`

```python
from urllib import urlopen
host = 'http://weather.noaa.gov/
fcst = '/pub/data/forecasts/state/
url = host + fcst + '/il/ilz013.txt'
print 'opening ' + url + ' ...
'
f = urlopen(url)
while True:
    s = f.readline()
    if s == '': break
    L = s.split(' ')
    if 'FCST' in L:
        s = f.readline()
        s = s + f.readline(); print s
    if 'CHICAGO' in L:
        s = s + f.readline()
        s = s + f.readline()
        s = s + f.readline(); print s
```
The Script `forecast.py`

```python
from urllib import urlopen
host = 'http://weather.noaa.gov/
fcst = '/pub/data/forecasts/state/
url = host + fcst + '/il/ilz013.txt'
print 'opening ' + url + ' ...

f = urlopen(url)
while True:
    s = f.readline()
    if s == '': break
    L = s.split(' ')
    if 'FCST' in L:
        s = f.readline()
        s = s + f.readline(); print s
    if 'CHICAGO' in L:
        s = s + f.readline()
        s = s + f.readline()
        s = s + f.readline(); print s
```
The Script `forecast.py`

```python
from urllib import urlopen
host = 'http://weather.noaa.gov/
fcst = '/pub/data/forecasts/state/
url = host + fcst + '/il/ilz013.txt'
print 'opening ' + url + ' ...

f = urlopen(url)
while True:
    s = f.readline()
    if s == '': break
    L = s.split(' ')
    if 'FCST' in L:
        s = f.readline()
        s = s + f.readline(); print s
    if 'CHICAGO' in L:
        s = s + f.readline()
        s = s + f.readline()
        s = s + f.readline(); print s
```
from urllib import urlopen
host = 'http://weather.noaa.gov/
fcst = '/pub/data/forecasts/state/
url = host + fcst + '/il/ilz013.txt'
print 'opening ' + url + ' ...

f = urlopen(url)
while True:
    s = f.readline()
    if s == '': break
    L = s.split(' ')
    if 'FCST' in L:
        s = f.readline()
        s = s + f.readline(); print s
    if 'CHICAGO' in L:
        s = s + f.readline()
        s = s + f.readline()
        s = s + f.readline(); print s
networking and the internet markup languages

1. Internet Basics
   evolution of the web
   IP addresses and URLs
   client/server and HTTP

2. Markup Languages
   HTML, XML, MathML
   MathML generated by Maple

3. Retrieving Data
   the weather forecast

4. CGI Programming
   showing current date and time in browser

5. Summary + Assignments
A sequel to GUIs: run programs through a browser. Web interfaces are also *event driven*.

Python as scripting language for the web: transform XML into XHTML pages.

The *Common Gateway Interface* (CGI) describes a set of protocols through which CGI programs interact with web servers and browsers.

CGI is common because it is not specific to any operating system or to any programming language or to any web server software.

We will use Python to generate dynamic web pages.
Python and CGI
Common Gateway Interface

A sequel to GUIs: run programs through a browser. Web interfaces are also *event driven*.

Python as scripting language for the web: transform XML into XHTML pages.

The Common Gateway Interface (CGI) describes a set of protocols through which CGI programs interact with web servers and browsers.

CGI is common because it is not specific to any operating system or to any programming language or to any web server software.

We will use Python to generate dynamic web pages.
Python and CGI
Common Gateway Interface

A sequel to GUls: run programs through a browser. Web interfaces are also *event driven*.

Python as scripting language for the web: transform XML into XHTML pages.

The *Common Gateway Interface* (CGI) describes a set of protocols through which CGI programs interact with web servers and browsers.

CGI is common because it is not specific to any operating system or to any programming language or to any web server software.

We will use Python to generate dynamic web pages.
A sequel to GUls: run programs through a browser. Web interfaces are also *event driven*.

Python as scripting language for the web: transform XML into XHTML pages.

The *Common Gateway Interface* (CGI) describes a set of protocols through which CGI programs interact with web servers and browsers.

CGI is common because it is not specific to any operating system or to any programming language or to any web server software.

We will use Python to generate dynamic web pages.
Python and CGI
Common Gateway Interface

A sequel to GUIs: run programs through a browser. Web interfaces are also *event driven*.

Python as scripting language for the web: transform XML into XHTML pages.

The *Common Gateway Interface* (CGI) describes a set of protocols through which CGI programs interact with web servers and browsers.

CGI is common because it is not specific to any operating system or to any programming language or to any web server software.

We will use Python to generate dynamic web pages.
Showing Current Time

Three steps to run Python scripts on the web:

1. **Open browser at** [http://localhost](http://localhost)
   
   *What should be visible is the message* If you can see this, it means that the installation of the Apache web server software on this system was successful.

2. **On MacOS X:**
   
   `/Library/Webserver/CGI-Executables`
   
   contains Python scripts.

3. **Write the Python script to show current time.**
   
   *From the module* `time` *we use the functions* `time()` *and `ctime()`.*
### Showing Current Time

Three steps to run Python scripts on the web:

1. **Open browser at** `http://localhost`
   
   What should be visible is the message *If you can see this, it means that the installation of the Apache web server software on this system was successful.*

2. **On MacOS X:**
   
   `/Library/Webserver/CGI-Executables` contains Python scripts.

3. **Write the Python script to show current time.**
   
   From the module `time` we use the functions `time()` and `ctime()`.
Showing Current Time

Three steps to run Python scripts on the web:

1. **Open browser at** `http://localhost`
   What should be visible is the message *If you can see this, it means that the installation of the Apache web server software on this system was successful.*

2. **On MacOS X:**
   `/Library/Webserver/CGI-Executables` contains Python scripts.

3. **Write the Python script to show current time.**
   From the module `time` we use the functions `time()` and `ctime()`.

Three steps to run Python scripts on the web:
showtime.py

#!/Library/Frameworks/Python.framework/Versions/Current/bin/python

import time

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

print time.ctime(time.time())

Two points:

1. The first line is the location of the Python interpreter.
2. The first print indicates that plain text is written (opposed to HTML code).
Internet Basics

evolution of the web
IP addresses and URLs
client/server and HTTP

Markup Languages

HTML, XML, MathML
MathML generated by Maple

Retrieving Data

the weather forecast

CGI Programming

showing current date and time in browser

Summary + Assignments

showtime.py

#!/Library/Frameworks/Python.framework/Versions/Current/bin/python

import time
print "Content-Type: text/plain\n"
print time.ctime(time.time())

Two points:

1. The first line is the location of the Python interpreter.
2. The first print indicates that plain text is written (opposed to HTML code).
#!/Library/Frameworks/Python.framework/Versions/Current/bin/python

import time
print "Content-Type: text/plain\n"
print time.ctime(time.time())

Two points:

1. The first line is the location of the Python interpreter.
2. The first print indicates that plain text is written (opposed to HTML code).
Summary + Assignments

Background: §4.2,3 in *Computer Science, an overview.* Start of chapter 13 in *Python Power!*

Assignments:

1. Make your own web page. Consult the ACCC help pages on web publishing.
2. Use Maple to generate MathML to display monomials, e.g. $8x^3$, and general polynomials.
3. Add a legend to the `forecast.py` script, using a dictionary to spell out the abbreviations PTCLDY, MOCLDY, etc.
4. Design a GUI dedicated to browsing weather forecasts. Which widgets will you use? What is the layout?
5. Write Python code for the GUI of the previous exercise.