Advanced CGI Scripts

Cookies

personalized web browsing

using cookies: count number of visits

Password Encryption

secure hash algorithm

Login Forms

using cookies for login data

the script `cookie_login.py`

the script `cookie_userpass.py`

MCS 275 Lecture 32
Programming Tools and File Management
Jan Verschelde, 4 April 2008

Advanced CGI Scripts

Cookies

- personalized web browsing
- using cookies: count number of visits

Password Encryption

- secure hash algorithm

Login Forms

- using cookies for login data
- the script `cookie_login.py`
- the script `cookie_userpass.py`

Personalized Web Browsing

more clever cgi scripts

Storing information about a client:

1. previous selections made
   → passing data from one script to another
2. personal information
   → identification and passwords
3. information from previous visits
   → counting number of visits

Potential applications:

1. customize displayed content
2. store encrypted password for fast access
3. personalized pricing ...

Personalized Web Browsing

more clever cgi scripts

Storing information about a client:

1. previous selections made
   → passing data from one script to another
2. personal information
   → identification and passwords
3. information from previous visits
   → counting number of visits

Potential applications:

1. customize displayed content
2. store encrypted password for fast access
3. personalized pricing ...

Personalized Web Browsing

more clever cgi scripts

Storing information about a client:

1. previous selections made
   → passing data from one script to another
2. personal information
   → identification and passwords
3. information from previous visits
   → counting number of visits

Potential applications:

1. customize displayed content
2. store encrypted password for fast access
3. personalized pricing ...

Personalized Web Browsing

more clever cgi scripts

Storing information about a client:

1. previous selections made
   → passing data from one script to another
2. personal information
   → identification and passwords
3. information from previous visits
   → counting number of visits

Potential applications:

1. customize displayed content
2. store encrypted password for fast access
3. personalized pricing ...

Personalized Web Browsing

more clever cgi scripts

Storing information about a client:

1. previous selections made
   → passing data from one script to another
2. personal information
   → identification and passwords
3. information from previous visits
   → counting number of visits

Potential applications:

1. customize displayed content
2. store encrypted password for fast access
3. personalized pricing ...

Personalized Web Browsing

more clever cgi scripts

Storing information about a client:

1. previous selections made
   → passing data from one script to another
2. personal information
   → identification and passwords
3. information from previous visits
   → counting number of visits

Potential applications:

1. customize displayed content
2. store encrypted password for fast access
3. personalized pricing ...

Cookies

storing information about client

Cookies are data

- stored by web server on client computer,
- managed by the browser.

Using the Cookie module:

```
>>> import Cookie
>>> c = Cookie.Cookie()
>>> c[’L’] = 32
>>> c[’date’] = ’Fri 4 Apr 2008’
>>> print c
Set-Cookie: L=32
Set-Cookie: date="Fri 4 Apr 2008"
```

Cookies are objects like dictionaries.
Reserved keys: `expires` and `path`.

Cookies

storing information about client

Cookies are data

- stored by web server on client computer,
- managed by the browser.

Using the Cookie module:

```
>>> import Cookie
>>> c = Cookie.Cookie()
>>> c['L'] = 32
>>> c['date'] = 'Fri 4 Apr 2008'
>>> print c
Set-Cookie: L=32
Set-Cookie: date="Fri 4 Apr 2008"
```

Cookies are objects like dictionaries.
Reserved keys: `expires` and `path`.

Cookies

storing information about client

Cookies are data

- stored by web server on client computer,
- managed by the browser.

Using the Cookie module:

```
>>> import Cookie
>>> c = Cookie.Cookie()
>>> c['L'] = 32
>>> c['date'] = 'Fri 4 Apr 2008'
>>> print c
Set-Cookie: L=32
Set-Cookie: date="Fri 4 Apr 2008"
```

Cookies are objects like dictionaries.
Reserved keys: `expires` and `path`.

Cookies

storing information about client

Cookies are data

- stored by web server on client computer,
- managed by the browser.

Using the Cookie module:

```
>>> import Cookie
>>> c = Cookie.Cookie()
>>> c['L'] = 32
>>> c['date'] = 'Fri 4 Apr 2008'
>>> print c
Set-Cookie: L=32
Set-Cookie: date="Fri 4 Apr 2008"
```

Cookies are objects like dictionaries.
Reserved keys: `expires` and `path`.

Cookies

storing information about client

Cookies are data

- stored by web server on client computer,
- managed by the browser.

Using the Cookie module:

```
>>> import Cookie
>>> c = Cookie.Cookie()
>>> c[’L’] = 32
>>> c[’date’] = ’Fri 4 Apr 2008’
>>> print c
Set-Cookie: L=32
Set-Cookie: date="Fri 4 Apr 2008"
```

Cookies are objects like dictionaries.
Reserved keys: `expires` and `path`.

Advanced CGI Scripts

Cookies

- personalized web browsing
- using cookies: count number of visits

Password Encryption

- secure hash algorithm

Login Forms

- using cookies for login data
- the script `cookie_login.py`
- the script `cookie_userpass.py`

Using Cookies

Count number of times browser visited.

Write a script to

1. retrieve cookie, initialize counter to zero,
2. or increment the value of counter by one,
3. and display counter value on the page.

Script is `cookie_counter.py`.

1. in `/Library/WebServer/CGI-Executables`,
2. browser settings must accept cookies.

Note: each browser has its own cookies.

Using Cookies

Count number of times browser visited.

Write a script to

1. retrieve cookie, initialize counter to zero,
2. or increment the value of counter by one,
3. and display counter value on the page.

Script is `cookie_counter.py`.

1. in `/Library/WebServer/CGI-Executables`,
2. browser settings must accept cookies.

Note: each browser has its own cookies.

Using Cookies

Count number of times browser visited.

Write a script to

1. retrieve cookie, initialize counter to zero,
2. or increment the value of counter by one,
3. and display counter value on the page.

Script is `cookie_counter.py`.

1. in `/Library/WebServer/CGI-Executables`,
2. browser settings must accept cookies.

Note: each browser has its own cookies.

Using Cookies

Count number of times browser visited.

Write a script to

1. retrieve cookie, initialize counter to zero,
2. or increment the value of counter by one,
3. and display counter value on the page.

Script is `cookie_counter.py`.

1. in `/Library/WebServer/CGI-Executables`,
2. browser settings must accept cookies.

Note: each browser has its own cookies.

Using Cookies

Count number of times browser visited.

Write a script to

1. retrieve cookie, initialize counter to zero,
2. or increment the value of counter by one,
3. and display counter value on the page.

Script is `cookie_counter.py`.

1. in `/Library/WebServer/CGI-Executables`,
2. browser settings must accept cookies.

Note: each browser has its own cookies.

Using Cookies

Count number of times browser visited.

Write a script to

1. retrieve cookie, initialize counter to zero,
2. or increment the value of counter by one,
3. and display counter value on the page.

Script is `cookie_counter.py`.

1. in `/Library/WebServer/CGI-Executables`,
2. browser settings must accept cookies.

Note: each browser has its own cookies.

Using Cookies

Count number of times browser visited.

Write a script to

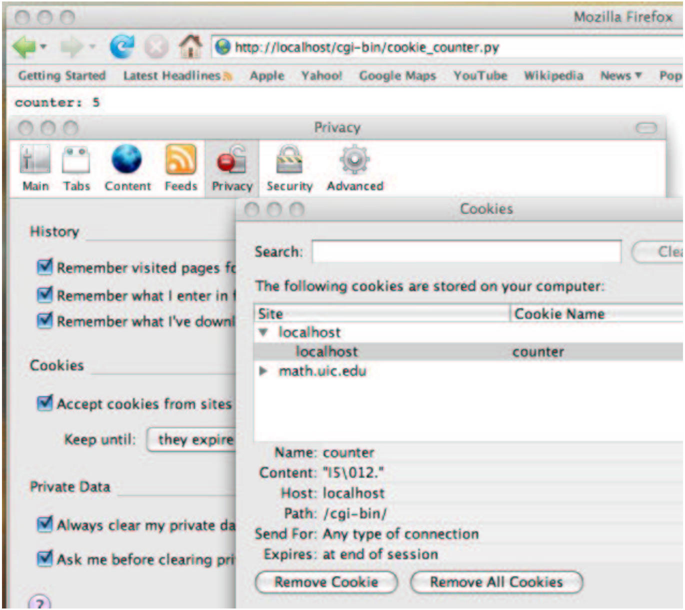
1. retrieve cookie, initialize counter to zero,
2. or increment the value of counter by one,
3. and display counter value on the page.

Script is `cookie_counter.py`.

1. in `/Library/WebServer/CGI-Executables`,
2. browser settings must accept cookies.

Note: each browser has its own cookies.

Running `cookie_counter.py`



```
counter: 5
```

The following cookies are stored on your computer:

Site	Cookie Name
▼ localhost	
localhost	counter
▶ math.uic.edu	

Name: counter
Content: "15\012."
Host: localhost
Path: /cgi-bin/
Send For: Any type of connection
Expires: at end of session

Code for main()

```
#!/Library/Frameworks/.../bin/python

import Cookie, os

def GetCookie():
   """
   Retrieves cookie, either initializes counter,
   or increments the counter by one.
   """

def main():
   """
   Retrieves a cookie and writes
   the value of counter to the page.
   """
   c = GetCookie()
   print c
   print "Content-Type: text/plain\n"
   print "counter: %d" % c['counter'].value
```

Code for main()

```
#!/Library/Frameworks/.../bin/python
import Cookie, os
def GetCookie():
   """
   Retrieves cookie, either initializes counter,
   or increments the counter by one.
   """

def main():
   """
   Retrieves a cookie and writes
   the value of counter to the page.
   """
   c = GetCookie()
   print c
   print "Content-Type: text/plain\n"
   print "counter: %d" % c['counter'].value
```

Code for `main()`

```
#!/Library/Frameworks/.../bin/python
import Cookie, os
def GetCookie():
   """
   Retrieves cookie, either initializes counter,
   or increments the counter by one.
   """

def main():
   """
   Retrieves a cookie and writes
   the value of counter to the page.
   """
   c = GetCookie()
   print c
   print "Content-Type: text/plain\n"
   print "counter: %d" % c['counter'].value
```

Code for main()

```
#!/Library/Frameworks/.../bin/python

import Cookie, os

def GetCookie():
   """
   Retrieves cookie, either initializes counter,
   or increments the counter by one.
   """

def main():
   """
   Retrieves a cookie and writes
   the value of counter to the page.
   """
   c = GetCookie()
   print c
   print "Content-Type: text/plain\n"
   print "counter: %d" % c[’counter’].value
```

Code for GetCookie()

```
def GetCookie():
   """
   Retrieves cookie, either initializes counter,
   or increments the counter by one.
   """
   if os.environ.has_key(’HTTP_COOKIE’):
      c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
   else:
      c = Cookie.Cookie()
   if not c.has_key(’counter’):
      c[’counter’] = 0
   else:
      c[’counter’] = c[’counter’].value + 1
   return c
```

Code for GetCookie()

```
def GetCookie():
   """
   Retrieves cookie, either initializes counter,
   or increments the counter by one.
   """
   if os.environ.has_key(’HTTP_COOKIE’):
      c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
   else:
      c = Cookie.Cookie()
   if not c.has_key(’counter’):
      c[’counter’] = 0
   else:
      c[’counter’] = c[’counter’].value + 1
   return c
```

Code for GetCookie()

```
def GetCookie():
    """
    Retrieves cookie, either initializes counter,
    or increments the counter by one.
    """
    if os.environ.has_key(’HTTP_COOKIE’):
        c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
    else:
        c = Cookie.Cookie()
    if not c.has_key(’counter’):
        c[’counter’] = 0
    else:
        c[’counter’] = c[’counter’].value + 1
    return c
```

Code for GetCookie()

```
def GetCookie():
   """
   Retrieves cookie, either initializes counter,
   or increments the counter by one.
   """
   if os.environ.has_key(’HTTP_COOKIE’):
      c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
   else:
      c = Cookie.Cookie()
   if not c.has_key(’counter’):
      c[’counter’] = 0
   else:
      c[’counter’] = c[’counter’].value + 1
   return c
```

Advanced CGI Scripts

Cookies

- personalized web browsing
- using cookies: count number of visits

Password Encryption

- secure hash algorithm

Login Forms

- using cookies for login data
- the script `cookie_login.py`
- the script `cookie_userpass.py`

Secure Hash Algorithm

for password encryption

We can use cookies for login names and passwords.

If passwords are unencrypted, then insecure.

Secure hash algorithm:

1. computationally infeasible to compute inverse
   → is *trapdoor* function
2. very low collision rate
   → very low chance that two different messages will generate the same key

Server encrypts password before sending to client.

Authentication by comparing encrypted passwords.

Secure Hash Algorithm

for password encryption

We can use cookies for login names and passwords.

If passwords are unencrypted, then insecure.

Secure hash algorithm:

1. computationally infeasible to compute inverse
   → is *trapdoor* function
2. very low collision rate
   → very low chance that two different messages will generate the same key

Server encrypts password before sending to client.

Authentication by comparing encrypted passwords.

Secure Hash Algorithm

for password encryption

We can use cookies for login names and passwords.

If passwords are unencrypted, then insecure.

Secure hash algorithm:

1. computationally infeasible to compute inverse
   → is *trapdoor* function
2. very low collision rate
   → very low chance that two different messages will generate the same key

Server encrypts password before sending to client.

Authentication by comparing encrypted passwords.

Secure Hash Algorithm

for password encryption

We can use cookies for login names and passwords.

If passwords are unencrypted, then insecure.

Secure hash algorithm:

1. computationally infeasible to compute inverse
   → is *trapdoor* function
2. very low collision rate
   → very low chance that two different messages will generate the same key

Server encrypts password before sending to client.

Authentication by comparing encrypted passwords.

Secure Hash Algorithm

for password encryption

We can use cookies for login names and passwords.

If passwords are unencrypted, then insecure.

Secure hash algorithm:

1. computationally infeasible to compute inverse
   → is *trapdoor* function
2. very low collision rate
   → very low chance that two different messages will generate the same key

Server encrypts password before sending to client.

Authentication by comparing encrypted passwords.

Using the sha Module

```
>>> import sha
>>> m = 'this is me'
>>> h = sha.new()
>>> h
<sha1 HASH object @ 0x99ff80>
>>> h.hexdigest()
'da39a3ee5e6b4b0d3255bfef95601890afd80709'
```

Using the sha Module

```
>>> import sha
>>> m = ’this is me’

>>> h = sha.new()
>>> h
<sha1 HASH object @ 0x99ff80>

>>> h.hexdigest()
’da39a3ee5e6b4b0d3255bfef95601890afd80709’
```

Using the sha Module

```
>>> import sha
>>> m = ’this is me’

>>> h = sha.new()
>>> h
<sha1 HASH object @ 0x99ff80>

>>> h.hexdigest()
’da39a3ee5e6b4b0d3255bfef95601890afd80709’
```

Advanced CGI Scripts

Cookies
- personalized web browsing
- using cookies: count number of visits

Password Encryption
- secure hash algorithm

Login Forms
- using cookies for login data
- the script `cookie_login.py`
- the script `cookie_userpass.py`

Login Forms

with cookies to remember data

Accessing `cookie_login.py` for the first time:

1. user submits login name and password,
2. submitted data processed by `cookie_userpass.py`,
3. cookie stores login name and encrypted password.

Connecting to `cookie_login.py` a second time:

1. cookie is retrieved,
2. login name is displayed if not empty,
3. user must type no password if in cookie.

Login Forms

with cookies to remember data

Accessing cookie_login.py for the first time:

1. user submits login name and password,
2. submitted data processed by cookie_userpass.py,
3. cookie stores login name and encrypted password.

Connecting to cookie_login.py a second time:

1. cookie is retrieved,
2. login name is displayed if not empty,
3. user must type no password if in cookie.

Login Forms

with cookies to remember data

Accessing `cookie_login.py` for the first time:

1. user submits login name and password,
2. submitted data processed by `cookie_userpass.py`,
3. cookie stores login name and encrypted password.

Connecting to `cookie_login.py` a second time:

1. cookie is retrieved,
2. login name is displayed if not empty,
3. user must type no password if in cookie.

Login Forms

with cookies to remember data

Accessing `cookie_login.py` for the first time:

1. user submits login name and password,
2. submitted data processed by `cookie_userpass.py`,
3. cookie stores login name and encrypted password.

Connecting to `cookie_login.py` a second time:

1. cookie is retrieved,
2. login name is displayed if not empty,
3. user must type no password if in cookie.

Login Forms

with cookies to remember data

Accessing `cookie_login.py` for the first time:

1. user submits login name and password,
2. submitted data processed by `cookie_userpass.py`,
3. cookie stores login name and encrypted password.

Connecting to `cookie_login.py` a second time:

1. cookie is retrieved,
2. login name is displayed if not empty,
3. user must type no password if in cookie.

Login Forms

with cookies to remember data

Accessing `cookie_login.py` for the first time:

1. user submits login name and password,
2. submitted data processed by `cookie_userpass.py`,
3. cookie stores login name and encrypted password.

Connecting to `cookie_login.py` a second time:

1. cookie is retrieved,
2. login name is displayed if not empty,
3. user must type no password if in cookie.

Advanced CGI Scripts

Cookies
- personalized web browsing
- using cookies: count number of visits

Password Encryption
- secure hash algorithm

Login Forms
- using cookies for login data
- the script `cookie_login.py`
- the script `cookie_userpass.py`

Script cookie_login.py

```
#!/Library/Frameworks/../bin/python

import cgi, Cookie, os

def main():
   """
   Form to process login.
   """
   c = GetCookie()
   print c
   print "Content-type: text/html\n"
   print "<html><body>\n"
   AskName(c)
   print "</body></html>\n"
```

Script cookie_login.py

```
#!/Library/Frameworks/../bin/python

import cgi, Cookie, os

def main():
   """
   Form to process login.
   """
   c = GetCookie()
   print c
   print "Content-type: text/html\n"
   print "<html><body>\n"
   AskName(c)
   print "</body></html>\n"
```

The Function GetCookie()

```
def GetCookie():
   """
   Retrieves cookie, and initializes it.
   """
   if os.environ.has_key(’HTTP_COOKIE’):
      c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
   else:
      c = Cookie.Cookie()
   if not c.has_key(’login’):
      c[’login’] = ’’
      c[’passw’] = ’’
   return c
```

The Function GetCookie()

```
def GetCookie():
   """
   Retrieves cookie, and initializes it.
   """
   if os.environ.has_key(’HTTP_COOKIE’):
      c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
   else:
      c = Cookie.Cookie()
   if not c.has_key(’login’):
      c[’login’] = ’’
      c[’passw’] = ’’
   return c
```

The Function AskName( )

first part

```
def AskName(c):
   """
   Form to enter user name, using cookie c
   to show user name.
   """
   print """<form method = "post"
               action = "cookie_userpass.py">
   """
   v = c[’login’].value
   w = c[’passw’].value
```

→ values of v and w
determine what will be asked from the user.

The Function AskName()

first part

```
def AskName(c):
    """
    Form to enter user name, using cookie c
    to show user name.
    """
    print """<form method = "post"
                   action = "cookie_userpass.py">
    """
    v = c[’login’].value
    w = c[’passw’].value
```

→ values of v and w
determine what will be asked from the user.

AskName() continued

```
if v == ’’:
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20>"""
else:
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20 value = %s>""" % v
if w == ’’:
   print """<p> Password:
   <input type = "password" name = "passw"
          size = 20>"""
print """
   <input type = "submit" value = "submit">
   </form>"""
```

AskName() continued

```
if v == ’’:
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20>"""
else:
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20 value = %s>""" % v
if w == ’’:
   print """<p> Password:
   <input type = "password" name = "passw"
          size = 20>"""
print """
   <input type = "submit" value = "submit">
   </form>"""
```

AskName() continued

```
if v == '':
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20>"""
else:
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20 value = %s>""" % v
if w == '':
   print """<p> Password:
   <input type = "password" name = "passw"
          size = 20>"""
print """
   <input type = "submit" value = "submit">
   </form>"""
```

AskName() continued

```
if v == ’’:
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20>"""
else:
   print """<p>
   Login Name:
   <input type = "text" name = "login"
          size = 20 value = %s>""" % v
if w == ’’:
   print """<p> Password:
   <input type = "password" name = "passw"
          size = 20>"""
print """
   <input type = "submit" value = "submit">
   </form>"""
```

Advanced CGI Scripts

Cookies

- personalized web browsing
- using cookies: count number of visits

Password Encryption

- secure hash algorithm

Login Forms

- using cookies for login data
- the script `cookie_login.py`
- the script `cookie_userpass.py`

Script `cookie_userpass.py`

```
#!/Library/Frameworks/../bin/python

import cgi, Cookie, os, sha

def main():
   """
   Form to process login.
   """
   form = cgi.FieldStorage()
   c = GetCookie(form)
   print c
   print "Content-type: text/html\n"
   print "<html><body>\n"
   error = ProcessName(form)
   if not error:
      ProcessPass(c)
   print "</body></html>\n"
```

Script cookie_userpass.py

```
#!/Library/Frameworks/../bin/python

import cgi, Cookie, os, sha

def main():
   """
   Form to process login.
   """
   form = cgi.FieldStorage()
   c = GetCookie(form)
   print c
   print "Content-type: text/html\n"
   print "<html><body>\n"
   error = ProcessName(form)
   if not error:
      ProcessPass(c)
   print "</body></html>\n"
```

Script cookie_userpass.py

```
#!/Library/Frameworks/../bin/python

import cgi, Cookie, os, sha

def main():
   """
   Form to process login.
   """
   form = cgi.FieldStorage()
   c = GetCookie(form)
   print c
   print "Content-type: text/html\n"
   print "<html><body>\n"
   error = ProcessName(form)
   if not error:
      ProcessPass(c)
   print "</body></html>\n"
```

The Function GetCookie()

```
def GetCookie(form):
   """
   Retrieves cookie and uses form to update.
   """
   if os.environ.has_key(’HTTP_COOKIE’):
      c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
   else:
      c = Cookie.Cookie()
   if c.has_key(’login’):
      if form.has_key(’login’):
         c[’login’] = form[’login’].value
   if c.has_key(’passw’):
      if form.has_key(’passw’):
         p = form[’passw’].value
         d = sha.new(p).hexdigest()
         c[’passw’] = d
   return c
```

The Function GetCookie()

```
def GetCookie(form):
   """
   Retrieves cookie and uses form to update.
   """
   if os.environ.has_key('HTTP_COOKIE'):
      c = Cookie.Cookie(os.environ['HTTP_COOKIE'])
   else:
      c = Cookie.Cookie()
   if c.has_key('login'):
      if form.has_key('login'):
         c['login'] = form['login'].value
   if c.has_key('passw'):
      if form.has_key('passw'):
         p = form['passw'].value
         d = sha.new(p).hexdigest()
         c['passw'] = d
   return c
```

The Function GetCookie()

```
def GetCookie(form):
   """
   Retrieves cookie and uses form to update.
   """
   if os.environ.has_key(’HTTP_COOKIE’):
      c = Cookie.Cookie(os.environ[’HTTP_COOKIE’])
   else:
      c = Cookie.Cookie()
   if c.has_key(’login’):
      if form.has_key(’login’):
         c[’login’] = form[’login’].value
   if c.has_key(’passw’):
      if form.has_key(’passw’):
         p = form[’passw’].value
         d = sha.new(p).hexdigest()
         c[’passw’] = d
   return c
```

Processing the Name

```
def ProcessName(form):
   """
   Processes name of login form.
   Returns True if error, else False.
   """
   error = False
   try:
      n = form[’login’].value
   except KeyError:
      print "please enter your name"
      error = True
   if not error:
      print ’welcome ’ + n + ’\n’
   return error
```

Processing Password

```
def ProcessPass(c):
   """
   Processes password of login form.
   """
   print "<p>your password is "
   print c['passw'].value
```

→ instead of printing the password,
compare against password on file.

Summary + Assignments

We covered more of chapter 14 in *Making Use of Python*.

Assignments:

1. Extend `cookie_counter.py` so that a different HTML page is displayed based on whether the counter is zero or not.
2. Extend `cookie_userpass.py` for it to do proper password authentication: compare the given password with the one that was earlier entered and stored on file.