# CGI Scripting

## 10.1 The Python `cgi` Module

This section is a summary of the CGI material from lectures:

### CGI scripts

- Apache passes information via environment variables/**`stdin`**
- **`GET`** data is passed in environment variables and **`POST`** data is passed on standard input. The main reason is that standard output is more efficient for large quantities of data.
- It expects the returned web content on **`stdout`**
- "Hello World" as a CGI script is thus pretty simple:

```
#!/usr/bin/python

print 'Content-Type: text/plain\r\n\r'
print 'Hello World'
```

- the `#!/usr/bin/python` or `#!/usr/bin/env python` invokes Python
- The output must always start with the **`Content-Type`** line followed by a blank line
- This tells the server and browser what type of file to expect
- Note the **`\r\n\r\n`** – that's because technically these fields should use Windows CRLF pairs

### Python CGI

- The Python **`cgi`** module handles both `GET` and `POST` requests
- The programmer doesn't need to know if it is `GET` or `POST`
- It gives simple dictionary-like access to the fields:
- The key difference to using a regular dictionary is that you need to add **`.value`** to get the value attribute out of the object returned by the dictionary lookup. The object is called a MiniFieldStorage, and it has attributes to help if the field is a file or some other type of form field.

```
import cgi

fields = cgi.FieldStorage()

first = fields['first'].value
last = fields['last'].value

print """<html>
<body>
  <h1>Hello %s %s</h1>
</body>
</html>""" % (first, last)
```

- Since **`POST`** requests use standard input, it means that you can only call **`FieldStorage`** once, otherwise all of the **`POST`** data will have already been read.

### Checking input

- The user can enter arbitrary URLs into the browser
- Typically, if the fields are not specified you want to send back an error page which asks them to fill out the necessary missing fields.
- So we must check that every field exists that we expect:

```
import cgi

FIRST = 'first'

fields = cgi.FieldStorage()

first = 'NONE'
if FIRST in fields:
  first = fields[FIRST].value
```

### Debugging CGI scripts

- Debugging CGI scripts can be a bit painful
- Error messages (on standard error) are recorded in the web server log not sent to the browser
- We need to redirect standard error so we can see them:

```
import sys

sys.stderr = sys.stdout
```

- This makes anything written to **`stderr`** go to **`stdout`**
- This does not help if there are syntax errors
- The **`sys.stderr`** trick only redirects the errors
- They can still be hard to read intermingled with HTML
- The **`cgitb`** module by Ka-Ping Yee *pretty prints* error messages

```
import sys
sys.stderr = sys.stdout

import cgitb
cgitb.enable()
```

- You should use this in any CGI script you are developing
- This and the previous technique should only be done for development systems – showing the error message to the user is a bit of a security risk since it gives hackers more information about the program internals. It is much better to have the errors go to a log file in a real system.

## 10.2 Walkthroughs

**Note:** Walkthrough 1 is critical for Stage 3 of the major project as it explains how to set up a CGI script on our system. Please ask us if you have any problems getting it to work.

### 10.2.1 Walkthrough 1: Hello Web

In this Walkthrough we create a simple CGI script and get it running on the ICRAR **`gpu02.icrar.org`** web server. You can copy the files from **`~james`** on **`gpu02`**.

Remember that the basic idea of CGI is that rather than retrieving a static (unchanging) HTML file, the web server can run a program which creates the web page on the fly. This allows for customization and dynamic updating of the content of the page.

1. Firstly write the following program into a file **`hello.cgi`** in your **`public_html/cgi-bin`** directory:

```
1  #!/usr/bin/python
2  print 'Content-Type: text/html\r\n\r'
3
4  print """<html>
5  <head>
6    <title>Hello Web</title>
7  </head>
8  <body>
9    <h1>Hello Web!!</h1>
10 </body>
11 </html>"""
```

2. Make the file executable

```
1  % chmod a+x hello.cgi
```

3. To see what is happening, you can run this program directly from the shell prompt which produces the following HTML output

```
1  % ./hello.cgi
2  Content-Type: text/html
3
4  <html>
5  <head>
6    <title>Hello Web</title>
7  </head>
8  <body>
9    <h1>Hello Web!!</h1>
10 </body>
11 </html>
```

4. Now run your script through a web browser by going to the URL
   `http://gpu02.icrar.org/~username/cgi-bin/hello.cgi`
   you should see a web page with the text **Hello Web!!**.

5. Note that the file suffix must be **`.cgi`** otherwise the web server may not run the script (depending on the Apache configuration).

All CGI scripts **must** first print a header `Content-Type: text/html` informing the web browser what type of content is being sent. `text/html` tells the browser that the data is a plain text file containing HTML. Following the `Content-Type` header, there must be a blank line. In the example above, this is handled by an empty print statement on line 3. **Leaving this out is a common mistake**.

### 10.2.2 Walkthrough 2: Hello Web, who are you?

In this walkthrough we build on the previous example to allow the user to enter information and have the CGI script process it and return a result. This is the basis from which more complex CGI scripts can be created.

1. Firstly we need to create a HTML page **`hello2.html`** including a form which will ask for the user's name. This needs to go into the **`public_html`** directory (not the **`cgi-bin`** directory). The HTML should look something like this:

```
<html>
<head>
  <title>Hello, who are you?</title>
</head>
<body>
  <h1>Hello, who are you?</h1>
  <p>
    <form action="cgi-bin/hello2.cgi" method="GET">
      Please enter your name:
      <input type="text" name="name" width="20"><br/>
      <input type="submit">
    </form>
  </p>
</body>
</html>
```

2. Put this page in your **`public_html`** directory and make it world readable **`chmod a+r hello.html`**.

3. Now we need to write the CGI script that is declared as the `action` property of the `form` element, in this case **`hello2.cgi`**. This script retrieves the name from the CGI request and pastes it into the HTML output.

4. The code should look something like this, which you can download from the course website

```
#!/usr/bin/python
import cgi
NAMEFIELD = 'name'

fields = cgi.FieldStorage()

name = 'Anonymous'
if NAMEFIELD in fields:
  name = fields[NAMEFIELD].value

print 'Content-Type: text/html'
print """
<html>
<head>
  <title>Hello %s</title>
</head>
<body>
  <h1>Hello %s!!</h1>
</body>
</html>""" % (name, name)
```

5. Make sure this file is executable **`chmod a+rx hello2.cgi`** and copy it into your **`public_html/cgi-bin`** directory.

6. Now point your browser at the HTML page and test the form
   http://gpu02.icrar.org/~username/hello2.html

### 10.2.3 Walkthrough 3: HTML Forms

Make sure you are familiar with creating forms in HTML. This was covered in the basic HTML tutorial you did in the last lab but if you didn't get to that bit, you should read through the forms section at
http://www.w3schools.com/html/html_forms.asp
By the time you finish this section you should be able to create your own basic form and know how to point it to a CGI script (you don't have to write the CGI script yet).

### 10.2.4 Walkthrough 4: The access and error logs

If your system administrator has made the files accessible, it is often helpful to inspect the access and error log files. These typically live in the **/var/log/apache2** directory and are called **access.log** and **error.log**. The error log is especially useful for debugging CGI script errors.

```
% tail -f /var/log/apache2/error.log
...
[Wed Feb 16 20:44:20 2011] [error] (13)Permission denied: exec of '/home/james/public_html/cgi-bin/test.cgi' failed
[Wed Feb 16 20:44:20 2011] [error] [client 202.8.37.138] Premature end of script headers: test.cgi
```

The **tail** command with the **-f** is very useful because it sits there checking to see if anything has been added to the end of the **error.log** file, and if so it prints it to the screen.

## 10.3 Exercises

These exercises combine what we have done previously in Python with the CGI module.

### 10.3.1 Exercise 1: Words to digits

Write a CGI script and webpage that takes a number as a series of words (e.g. one two three five seven) and converts it a numerical expression (e.g. 12357) which is displayed in a webpage.

### 10.3.2 Exercise 2: Crossword solver

Write a CGI script and webpage which returns a list of words matching a given pattern for crossword solvers, where **?** is used to indicate unknown letters. In other words, the user should be able to enter `h??se` and get back a list of words matching this pattern.

## 10.4 Capability checklist

When you've finished this lab, check that you know how to...

1. Write a website that includes a form for user input
2. Write a basic CGI script in Python
3. Get a website and CGI script working on the IT machines
4. Debug CGI scripts
5. Check the web server error log to find problems

If you don't know how to do any of these things once you have completed the lab, please come and ask us.