# CS1116/CS5018

# Web Development 2

Dr Derek Bridge
School of Computer Science & Information Technology
University College Cork

## Revision: A question about HTML

- Suppose you want to put 'funny' characters into your Web page. How do you do it?

  © á Ω

- These aren't 'funny' characters. But they often need special treatment too. Why?

  < &

## Two server-side programs that expect user input

- `challenge.html` and `response.py`
- `bmi.html` and `bmi.py`

## Questions about `response.py`

- How can you supply data to this program without using the form?
- What happens if you submit no data?

## A better version of `response.py`

```
#!/usr/local/bin/python3

from cgitb import enable
enable()

from cgi import FieldStorage

print('Content-Type: text/html')
print()

form_data = FieldStorage()
fname = form_data.getfirst('firstname', '').strip()
sname = form_data.getfirst('surname', '').strip()

outcome = ''
if fname and sname:
    outcome = 'Hello, %s %s. You may go on your way.' % (fname, sname)
else:
    outcome = 'You did not enter a first name and a surname. You are under arrest.'

print("""
    <!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="utf-8" />
            <title>Challenge</title>
        </head>
        <body>
            <p>
                %s
            </p>
        </body>
    </html>""" % (outcome))
```

## User data that contains HTML tags

- In `response.py`, user data is shown in an HTML document
- Suppose the user data contains HTML tags
- E.g. the user enters: `<h1>Hugh</h1>`
- Q: What will happen?

## A script injection attack

- Hence, suppose the user enters: `<script>window.alert("Watch out!")</script>`
- Q: What will happen?
- To thwart these attacks, we must **escape** characters in the user's data that have a special meaning in HTML

# An even better version of `response.py`

```
#!/usr/local/bin/python3

from cgitb import enable
enable()

from cgi import FieldStorage
from html import escape

print('Content-Type: text/html')
print()

form_data = FieldStorage()
fname = escape(form_data.getfirst('firstname', '').strip())
sname = escape(form_data.getfirst('surname', '').strip())

outcome = ''
if fname and sname:
    outcome = 'Hello, %s %s. You may go on your way.' % (fname, sname)
else:
    outcome = 'You did not enter a first name and a surname. You are under arrest.'

print("""
    <!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="utf-8" />
            <title>Challenge</title>
        </head>
        <body>
            <p>
                %s
            </p>
        </body>
    </html>""" % (outcome))
```

# Questions about `bmi.py`

- What problems can the user cause us?

# Ways to test for a decimal number

- **Regular expressions** give us a form of 'pattern matching'
- Testing whether s contains a decimal number:

```
import re

if re.match('\d+(\.\d+)?$', s) is None:
    # s does not contain a decimal number
else:
    # s does contain a decimal number
```

- But this regular expression is too strict
  Q: Give an example
- A better way is to use `try` and `except`

# A better version of `bmi.py`

```
#!/usr/local/bin/python3

from cgitb import enable
enable()

from cgi import FieldStorage

print('Content-Type: text/html')
print()

form_data = FieldStorage()
mass_kg = form_data.getfirst('mass_kg', '').strip()
height_m = form_data.getfirst('height_m', '').strip()

outcome = ''
try:
    mass_kg = float(mass_kg)
    height_m = float(height_m)
    bmi = mass_kg / (height_m * height_m)
    category = ''
    if bmi < 18.5:
        category = 'underweight'
    elif bmi > 25:
        category ='overweight'
    else:
        category = 'normal'
    outcome = """Your mass in kg is %.1f. Your height in m is %.1f.
                 Your BMI is %.2f. You are %s.""" % (mass_kg, height_m, bmi, category)
except ValueError:
    outcome = 'You should enter numbers for your weight and height.'

print("""
    <!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="utf-8" />
            <title>BMI</title>
        </head>
        <body>
            <p>
                %s
            </p>
        </body>
    </html>""" % (outcome))
```

## Notes

- `try/except` (simplified):
  - If it successfully coerces the string into a float,
    - it executes the rest of the `try` block
    - it skips the `except` block
  - If the coercion fails,
    - it skips the rest of the `try` block
    - it executes the `except` block