# Information Infrastructure II

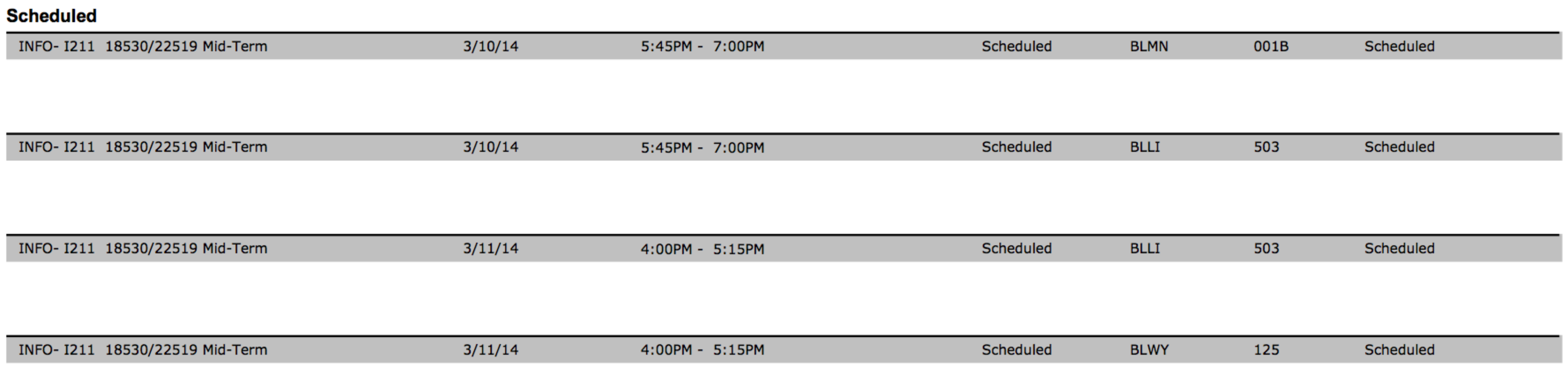
**INFO I211 – Spring 2014 – Sections 18530 & 22719**

***Lecture 14 – 2014.03.03 & 2014.03.04***

**Instructor:**
**Mitja Hmeljak,**
**http://mypage.iu.edu/~mitja**
**mitja@indiana.edu**

# I211 – Midterm exam

*when?* Monday March 10 and Tuesday March 11, at I211 lecture time



**Scheduled**

| | | | | | | |
|---|---|---|---|---|---|---|
| INFO- I211 18530/22519 Mid-Term | 3/10/14 | 5:45PM - 7:00PM | Scheduled | BLMN | 001B | Scheduled |
| INFO- I211 18530/22519 Mid-Term | 3/10/14 | 5:45PM - 7:00PM | Scheduled | BLLI | 503 | Scheduled |
| INFO- I211 18530/22519 Mid-Term | 3/11/14 | 4:00PM - 5:15PM | Scheduled | BLLI | 503 | Scheduled |
| INFO- I211 18530/22519 Mid-Term | 3/11/14 | 4:00PM - 5:15PM | Scheduled | BLWY | 125 | Scheduled |

*where:* Monday March 10 : MN001B and LI503

Tuesday March 11: LI503 and WY125

## I211 – Midterm exam

I211 Midterm – two tasks:

- Task 1 – Questions (short & concise)
- Task 2 – Programming exercise

during the exam you may consult:

- anything on the Oncourse section for this class (either printed or online)
- your I211 textbook
- Python 2.6 on silo.soic.indiana.edu
- your preferred text editor

*no other printouts, no internet*

*except to access Oncourse, and silo.soic.indiana.edu*

## Connecting to CGI

For URL addresses, HTTP supports A-Z, a-z, 0-9, /, : and .

It also supports many special characters, like ~ and space

- However, these may need to be converted into special character encodings, for all http sites, browsers, etc. to understand them
- Just like \n and \t are understood to be special characters for printing
- For URLs, this is called **quoting**.

## Connecting to CGI

Special characters are converted into hexadecimal codes (base 16 numbers)

A single character is represented by a single code, even if that code uses multiple numbers or letters. Ex: C8 is hex for 200 base 10.

To get the integer code for a character:

```
ord("A")   -> 65          #ord stands for 'ordinal'
ord("~")    ->  126       #these numbers are base 10
```

## Connecting to CGI

So, to quote a URL, we:

1. Identify a special character **char = “~”**
2. Get its integer code **c = ord(char)**
3. Convert the code to hex
   1. **h = hex(c)[2:] -> “7e” #we remove the “0x”**
   2. **h = "%X" % c -> “7E”#second method**
4. Need to add a “%” in front of hex codes

Why quoting?

To form a standard URL that will always work

which also answers the question '*What are those "%20" codes in* URLs?'

[http://www.blooberry.com/indexdot/html/topics/urlencoding.htm](http://www.blooberry.com/indexdot/html/topics/urlencoding.htm)

## URL Quote (Group Work)

Write a function called **quoteURL** that takes one parameter, **url**, and returns that url with all special characters quoted. (Don't use the built-in function) A special character is anything other than **A-Z**, **a-z**, **0-9**, **/**, **:** and **.**

Once **quoteURL** is working, paste your **getContent** function into the file, and make sure this works:

```
getContent(quoteURL("http://www.cs.indiana.edu/~mitja/tmp/I211/I211test.html"))
```

## URL Quote (Solution)

```
import quoteURL
import urllib, os


def getContent(url):
    web_page = urllib.urlopen(url)
    lines = web_page.readlines()

    filename = os.path.basename(url)

    f = open(filename, "w")

    for line in lines:
        line = line.decode("utf-8")
        f.write(line)

    f.close()

    web_page.close()

q = quoteURL.quoteURL("http://www.cs.indiana.edu/~mitja/tmp/I211/I211test.html")
print q
getContent(q)
print "done!"
```

## URL Quote (Complete Solution)

```
import os, urllib

def quoteURL(url):
    newURL = ""

    for letter in url:
        if (letter >= "A" and letter <= "Z") \
           or (letter >= "a" and letter <= "z") \
           or (letter >= "0" and letter <= "9") \
           or (letter == ":") or (letter == "/") or (letter == ".") :
            newURL += letter
        else:
            decimalCode = ord(letter)
            hexadecimalCode = hex(decimalCode)[2:]
            newURL += "%" + hexadecimalCode

    return newURL

def getContent(url):
    fname = os.path.basename(url)
    if (fname == "") or (fname == "/") or (fname == "\\"):
        fname = "index.html"

    myConnection = urllib.urlopen(url)
    content = myConnection.readlines()
    f = open(fname, "w")
    for i in range(len(content)):
        data = content[i].decode("utf-8")
        f.write(data)

    f.close()
    myConnection.close()

# main program:
if __name__ == "__main__":
    theQuotedUrl = \
        quoteURL("http://www.cs.indiana.edu/~mitja/tmp/I211/I211test.html")
    print theQuotedUrl
    getContent(theQuotedUrl)
    print "done!"
```

## Connecting to CGI

Python actually has a built-in method for this!

```
import urllib
urllib.quote("http://www.cs.indiana.edu/~mitja/tmp/I211/I211test.html")
```

->

```
'http%3A//www.cs.indiana.edu/%7Emitja/tmp/I211/I211test.html'
```

This also quotes **:**, which can give mixed results. You may prefer using your version!

## Connecting to CGI

If you want your original url back (to display it, for example), use the unquote method:

```
import urllib
urllib.unquote('http%3A//www.cs.indiana.edu/%7Emitja/tmp/I211/I211test.html')
```

→

```
'http://www.cs.indiana.edu/~mitja/tmp/I211/I211test.html'
```

## CGI scripts, reviewed

**CGI**, the **C**ommon **G**ateway **I**nterface

**CGI** is an interface to pass requests from a web server to an executable program, and return the program's results to a web browser.

The Yahoo finance page and the Boston Market locator we saw last time were both **CGI scripts**.

CGI scripts can be written in many different scripting languages.

## CGI scripts, reviewed

https://help.soic.indiana.edu/homes/

Creating a CGI/PHP homepage enables you to upload CGI scripts to your burrows account (in a folder called **cgi-pub**) and run them from a URL.

Your scripts arereachable thus:

http://cgi.soic.indiana.edu/~username/scriptname.cgi

## CGI scripts, reviewed

On Unix systems, a user's *home directory* is also referred to as:

**~username/**

or simply:

**~/**

for the current user.

This is also a good moment to review commonly used Unix Commands, etc.

## CGI scripts, reviewed

How do we turn a ***.py*** source code file into an *executable* (i.e. a program that can be run directly) on UNIX systems?

This way:

the first line of a Python CGI script needs to be:

```
#!/usr/bin/env python
```

This line tells the UNIX system that this is a Python 3 CGI script, as opposed to one in some other language (it also asks the "env" command to find python for us).

We also need to run **`chmod ugo+rx`** on the ***.py*** file.

## CGI scripts, reviewed: CGI Hello World - *hello.cgi*

What does this program do?:

```
#!/usr/bin/env python
print "Content-type: text/html\n\n"
# the above is necessary for CGI to work
print "<html><head><title>First CGI</title></head>"
print "<body>Hello World!<br></body>"
print "</html>"
```

To run this program as CGI, you need to place this file into your ~/cgi-pub/ directory and then change the rights on it *to be executable* by running this command:

**chmod ugo+rx hello.cgi**

You'll then be able to see the results of this file at this URL:

http://cgi.soic.indiana.edu/~yourusername/hello.cgi

## CGI scripts, reviewed

**Content-type: text/html**

this first line tells the browser to expect a text-based HTML file content, as opposed to some other form of data.

You can see a list of content types here: http://en.wikipedia.org/wiki/Internet_media_type#List_of_common_media_types

These are called **MIME types**.

The **\n** at the end of the MIME type line is needed to tell the browser that the actual content begins there!