Week 09 Laboratory Sample Solutions

Objectives

- Developing Python & Shell skills
- Exploring simple approaches to scraping data from the web

Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples.

Getting Started

Set up for the lab by creating a new directory called `lab09` and changing to this directory.

```
$ mkdir lab09
$ cd lab09
```

There are no provided files for this lab.

EXERCISE:

What Courses Does UNSW Have this Year - Shell

Write a POSIX-compatible shell script `courses.sh` which given a course prefix, e.g. COMP, prints the course codes and names of all UNSW courses with that prefix offered this year on the Kensington Campus.

courses should be sorted by course number (lowest to highest).

duplicate course codes should be removed, keeping the course whose name is alphabetically first.

```
$ ./courses.sh VISN
VISN1101 Seeing the World: Perspectives from Vision Science
VISN1111 Geometrical and Physical Optics
VISN1221 Visual Optics
VISN2111 Ocular Anatomy and Physiology
VISN2211 Organisation and Function of the Visual System
VISN3111 Development and Aging of the Visual System
VISN4016 Vision Science Honours
VISN5511 The Visual System, Impairments and Implications
VISN5512 Sensory Processes and Movement
VISN5513 Orientation and Mobility Foundations: Disability, Diversity and Inclusion
VISN5521 Orientation and Mobility Techniques
VISN5522 Vision Rehabilitation
VISN5523 Orientation and Mobility in Practice
VISN5531 Development and Ageing: Implications for Orientation and Mobility
$ ./courses.sh COMP | tail
COMP9491 Applied Artificial Intelligence
COMP9511 Human Computer Interaction
COMP9517 Computer Vision
COMP9727 Recommender Systems
COMP9801 Extended Design and Analysis of Algorithms
```

HINT:

Make the first line of your shell-script `#!/bin/dash`

The information you need for the course code prefix `COMP` can be found in this web page: http://www.timetable.unsw.edu.au/2022/COMPKENS.html . You can assume this is the case for all valid prefixes.

The command `curl` will download a URL and print it to standard output.e.g.:

```
$ curl --location --silent http://www.timetable.unsw.edu.au/2022/COMPKENS.html|head
<title>Class Search by Teaching Period</title>
<link rel="stylesheet" type="text/css" href="../layout/2020/myunsw.css">
<head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

   <table width="100%" cellspacing="0" cellpadding="0">
      <form name="googleForm" method="GET"
action="http://www.google.com/u/theuniversityofnewsouthwales" target="_blank">
         <tr>
            <td width="30%" style="height:120px; border-bottom:10px solid #FFCC00; background-
color:#fff;"><a href="http://www.unsw.edu.au" target="_blank"><img border="0" src="/images-
timetable/banner2020.jpg" alt="The University of New South Wales" width="836" height="179"
style="float:left; margin-left:20px; margin-top:25px;"></a></td>
            <td width="70%" style="height:120px; border-bottom:10px solid #FFCC00; background-
color:#fff; vertical-align:bottom; " align="right">
```

In a script it is best run as `curl --silent` so it doesn't print extra information on standard error.

The `--location` is required so `curl` will follow a HTTP redirect from the URL.

You may find uniq's **-w** option useful when removing duplicate courses.

NOTE:

You may not use non-POSIX-compatible shell features such as bash extensions.
Your script must work when run by `/bin/dash` on a CSE system.
You are not permitted to rely on the extra features provided by `/bin/bash` or `/bin/sh` .
You can assume anything that works with the version of `/bin/dash` on CSE systems is POSIX compatible.

You may not use Perl, C, Python, or any other language.

You must use the URL provided above to download the course list.

No error checking is necessary.

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest shell_courses
```

When you are finished working on this exercise, you must submit your work by running `give` :

```
$ give cs2041 lab09_shell_courses courses.sh
```

before **Monday 01 August 12:00** to obtain the marks for this lab exercise.

SOLUTION:

Sample solution for `courses.sh`

```
#!/bin/dash

case "$#" in
    1) course_prefix="$1" ;;
    *)
        echo "Usage: $0 <course prefix>"
        exit 1
        ;;
esac

case "${course_prefix}" in
    [A-Z][A-Z][A-Z][A-Z]) ;;
    *)
        echo "Invalid course prefix: ${course_prefix}"
        exit 1
        ;;
esac

url="http://www.timetable.unsw.edu.au/2022/${course_prefix}KENS.html"

curl --location --silent "${url}" |
grep -E "${course_prefix}[0-9]{4}\.html" |
sed '
    s/.*href="//
    s/.html">/ /
    s/<.*//
' |
grep -Ev "${course_prefix}[0-9]{4} ${course_prefix}[0-9]{4}$" |
sort |
uniq -w8
```

EXERCISE:

What Courses Does UNSW Have this Year - Python/subprocess

Write a Python script `courses_subprocess.py` which given a course prefix, e.g. COMP, prints the course codes and names of all UNSW courses with that prefix offered this year on the Kensington Campus.

```
$ ./courses_subprocess.py VISN
VISN1101 Seeing the World: Perspectives from Vision Science
VISN1111 Geometrical and Physical Optics
VISN1221 Visual Optics
VISN2111 Ocular Anatomy and Physiology
VISN2211 Organisation and Function of the Visual System
VISN3111 Development and Aging of the Visual System
VISN4016 Vision Science Honours
VISN5511 The Visual System, Impairments and Implications
VISN5512 Sensory Processes and Movement
VISN5513 Orientation and Mobility Foundations: Disability, Diversity and Inclusion
VISN5521 Orientation and Mobility Techniques
VISN5522 Vision Rehabilitation
VISN5523 Orientation and Mobility in Practice
VISN5531 Development and Ageing: Implications for Orientation and Mobility
$ ./courses_subprocess.py COMP | tail
COMP9491 Applied Artificial Intelligence
COMP9511 Human Computer Interaction
COMP9517 Computer Vision
COMP9727 Recommender Systems
COMP9801 Extended Design and Analysis of Algorithms
```

You should use the `subprocess` module to download the web page.
Using the same `curl` command as the last activity.

HINT:

The information you need for the course code prefix `COMP` can be found in this web page:
http://www.timetable.unsw.edu.au/2022/COMPKENS.html . You can assume this is the case for all prefixes.

NOTE:

Your answer must be Python only. You can not use other languages such as Shell, Perl or C.

You **may** run external programs with the `subprocess` module.

You must use the URL provided above to download the course list.

No error checking is necessary.

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest python_courses_subprocess
```

When you are finished working on this exercise, you must submit your work by running `give` :

```
$ give cs2041 lab09_python_courses_subprocess courses_subprocess.py
```

before **Monday 01 August 12:00** to obtain the marks for this lab exercise.

SOLUTION:

Sample solution for `courses_subprocess.py`

```
#! /usr/bin/env python3

import subprocess
import sys
import re

YEAR = "2022"

assert len(sys.argv) == 2, f"Usage: {sys.argv[0]} <course prefix>"
course_prefix = sys.argv[1]
assert re.fullmatch(r"[A-Z]{4}", course_prefix), f"Invalid course prefix: {course_prefix}"

url = f"http://www.timetable.unsw.edu.au/{YEAR}/{course_prefix}KENS.html"

proc = subprocess.run(['curl', '--location', '--silent', url], capture_output=True, text=True)

courses = []

for m in re.findall(rf"^.*{course_prefix}[0-9]{{4}}\.html.*$", proc.stdout, flags=re.MULTILINE):
    m = re.search(r"""<a href="(?P<code>[A-Z]{4}[0-9]{4})\.html">(?P<name>.*?)</a>""", m)
    code = m.group('code')
    name = m.group('name')
    if code != name:
        courses.append((code, name))

for code, name in sorted(set(courses)):
    print(f"{code} {name}")
```

EXERCISE:

What Courses Does UNSW Have this Year - Python/requests

Write a Python script `courses_requests.py` which given a course prefix, e.g. COMP, prints the course codes and names of all UNSW courses with that prefix offered this year on the Kensington Campus.

```
$ ./courses_requests.py VISN
VISN1101 Seeing the World: Perspectives from Vision Science
VISN1111 Geometrical and Physical Optics
VISN1221 Visual Optics
VISN2111 Ocular Anatomy and Physiology
VISN2211 Organisation and Function of the Visual System
VISN3111 Development and Aging of the Visual System
VISN4016 Vision Science Honours
VISN5511 The Visual System, Impairments and Implications
VISN5512 Sensory Processes and Movement
VISN5513 Orientation and Mobility Foundations: Disability, Diversity and Inclusion
VISN5521 Orientation and Mobility Techniques
VISN5522 Vision Rehabilitation
VISN5523 Orientation and Mobility in Practice
VISN5531 Development and Ageing: Implications for Orientation and Mobility
$ ./courses_requests.py COMP | tail
COMP9491 Applied Artificial Intelligence
COMP9511 Human Computer Interaction
COMP9517 Computer Vision
COMP9727 Recommender Systems
COMP9801 Extended Design and Analysis of Algorithms
```

You should use the `requests` module to download the web page.

You should use the `BeautifulSoup` and `html5lib` modules to parse the HTML.

HINT:

The `fetch_website_text.py` lecture example uses `BeautifulSoup` (but not `requests`).

Use `BeautifulSoup` to find all links (`a` tags) in a page then look at the link's `href` and `text` fields.

The information you need for the course code prefix `COMP` can be found in this web page:
`http://www.timetable.unsw.edu.au/2022/COMPKENS.html`. You can assume this is the case for all prefixes.

NOTE:

Your answer must be Python only. You can not use other languages such as Shell, Perl or C.

You may not run external programs.

You must use the URL provided above to download the course list.

No error checking is necessary.

You can find documentation for the `requests` module here: https://pypi.org/project/requests/
You can find documentation for the `BeautifulSoup` module here: https://pypi.org/project/beautifulsoup4/

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest python_courses_requests
```

When you are finished working on this exercise, you must submit your work by running `give`:

```
$ give cs2041 lab09_python_courses_requests courses_requests.py
```

before **Monday 01 August 12:00** to obtain the marks for this lab exercise.

SOLUTION:

Sample solution for `courses_requests.py`

```python
#! /usr/bin/env python3

import requests
import bs4
import sys
import re

YEAR = "2022"

assert len(sys.argv) == 2, f"Usage: {sys.argv[0]} <course prefix>"
course_prefix = sys.argv[1]
assert re.fullmatch(r"[A-Z]{4}", course_prefix), f"Invalid course prefix: {course_prefix}"

url = f"http://www.timetable.unsw.edu.au/{YEAR}/{course_prefix}KENS.html"

soup = bs4.BeautifulSoup(requests.get(url).text, 'html5lib')

courses = []

for tag in soup.find_all('a'):
    code = tag.get('href', '')
    name = tag.text
    if re.fullmatch(r'[A-Z]{4}[0-9]{4}\.html', code) and code[:-5] != name:
        courses.append((code[:-5], name))


for code, name in sorted(set(courses)):
    print(f"{code} {name}")
```

CHALLENGE EXERCISE:

What Can't Regexes Do?

Write a regular expression which matches a unary number iff it is composite (not prime).

In other words, write a regex that matches a string of *n* ones iff *n* is composite.

Here is a test program assist you in doing this:

```python
#! /usr/bin/env python3

from sys import argv
from re import search
from math import log, floor

assert len(argv) == 4, f"Usage: {argv[0]} <min> <max> <regex>"

min, max, regex = argv[1], argv[2], argv[3]

assert len(regex) <= 80, "regex too large";

padding = floor(log(int(max) + 1, 10)) + 1

for i in range(int(min), int(max) + 1):
    unary = '1' * i
    print(f"{i:{padding}} = {unary} unary -", "composite" if search(regex, unary) else "prime")
```

Download test_regex_prime.py, or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/22T2/activities/regex_prime/test_regex_prime.py test_regex_prime.py
```

For example to test the regex **`^1{7,10}$`** against the integers 2 to 12, you can run

```
$ chmod 755 test_regex_prime.py
$ ./test_regex_prime.py 2 12 '^1{7,10}$'
 2 = 11 unary - prime
 3 = 111 unary - prime
 4 = 1111 unary - prime
 5 = 11111 unary - prime
 6 = 111111 unary - prime
 7 = 1111111 unary - composite
 8 = 11111111 unary - composite
 9 = 111111111 unary - composite
10 = 1111111111 unary - composite
11 = 11111111111 unary - prime
12 = 111111111111 unary - prime
```

Put your solution in `regex_prime.txt` , for example:

```
$ ./test_regex_prime.py 40 50 "$(cat regex_prime.txt)"
40 = 1111111111111111111111111111111111111111 unary - composite
41 = 11111111111111111111111111111111111111111 unary - prime
42 = 111111111111111111111111111111111111111111 unary - composite
43 = 1111111111111111111111111111111111111111111 unary - prime
44 = 11111111111111111111111111111111111111111111 unary - composite
45 = 111111111111111111111111111111111111111111111 unary - composite
46 = 1111111111111111111111111111111111111111111111 unary - composite
47 = 11111111111111111111111111111111111111111111111 unary - prime
48 = 111111111111111111111111111111111111111111111111 unary - composite
49 = 1111111111111111111111111111111111111111111111111 unary - composite
50 = 11111111111111111111111111111111111111111111111111 unary - composite
```

> NOTE:
>
> This exercise is not possible with true regular expression, i.e using `|*()` alone,
> You will need to use additional regular expression syntax to achieve the same result.
> Python regular expression syntax is described in the In the RE module documentation.
>
> Your regex must be less than 80 characters.
>
> This is a *(in)?famous* problem to try and solve with regex.
> There will be answers easily found online.
> Don't google for other people solutions - see if you can come up with your own.

When you think your program is working, you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest regex_prime
```

When you are finished working on this exercise, you must submit your work by running `give` :

```
$ give cs2041 lab09_regex_prime regex_prime.txt
```

before **Monday 01 August 12:00** to obtain the marks for this lab exercise.

> SOLUTION:
>
> Sample solution for `regex_prime.txt`
>
> ```
> ^1?$|^(11+?)\1+$
> ```

Submission

When you are finished each exercises make sure you submit your work by running `give` .

You can run `give` multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Week 10 Monday 12:00:00** to submit your work.

You cannot obtain marks by e-mailing your code to tutors or lecturers.

You check the files you have submitted here.

Automarking will be run by the lecturer several days after the submission deadline, using test cases different to those `autotest` runs for you. (Hint: do your own testing as well as running `autotest` .)

After automarking is run by the lecturer you can view your results here. The resulting mark will also be available via give's web interface.

Lab Marks

When all components of a lab are automarked you should be able to view the the marks via give's web interface or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```

COMP(2041|9044) 22T2: Software Construction is brought to you by
the School of Computer Science and Engineering
at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs2041@cse.unsw.edu.au
CRICOS Provider 00098G