2 - Reading RESTful APIs

1. Learning Outcomes

After completing the exercises in this lab you should be able to the curl utility and the Python language to read, parse and process responses from RESTful API endpoints

2. Organisation

Please attempt this lab individually as you will need the skills acquired in this lab in order to complete later labs in this module.

3. Grading

This worksheet is worth up to 10% (DT211C/3) or 5% (DT249/4) of your overall module grade. You must attend and sign in at the labs in order to obtain the credit for the associated worksheets. You may work on this worksheet during lab 3 and lab 4 with instructor assistance. You must also demonstrate your submission in order to receive credit - see below.

4. Submission

The deadline for submission is Wednesday Oct 12, 2016 @23:59 through Webcourses.

5. Demonstration

You will give a brief demonstration of your submission to the lab instructor in lab 5.

6. Requirements

For this lab you will need to sign up for the following services if you've not already done so:

- Access to a Linux/Mac/Windows shell environment on your laptop or in the cloud or a free account with a hosted environment such as Nitrous.IO (https://nitrous.io) or TutorialsPoint (http://tutorialspoint.com) or Cloud 9 (https://c9.io) or similar
- A local copy of the curl utility in your environment
- An installation of the Python language in your environment

7. Resources

You are free to research whatever you need to solve the problems in this lab. Some recommended resources include:

- The official *curl* tutorial here
- Online course on the *Python* here

8. Quick Primer

A RESTful API is a HTTP-accessible interface to some server-side resource, typically an XML or JSON document describing a resource such as a blog post or user.

A typical API will support **C**reate/**R**ead/**U**pdate/**D**elete operations on those resources which are often backed by a persistent database.

The HTTP protocol supports a number of operation types (sometimes called verbs). These are:

- GET - used for reading the contents of a resource or collection of resources
- POST - used for creating a new resource
- PATCH - used to partially update an existing resource
- PUT - used to fully replace an existing resource
- DELETE - used to remove a resource

The curl utility is a HTTP client which runs on the command-line. You can think it of it as a partial browser but without the ability to display formatted HTML or execute downloaded Javascript.

As a HTTP client, curl can connect to HTTP server endpoint and send one of the HTTP verbs listed above with a request payload

In this lab, you will focus on GET requests which can be specified using simple URL requests

For example, to fetch a webpage using curl you can do this:

```
curl -s -X GET https://google.com
```

To get the same page using Python you can run this code:

```
import urllib.request
page = urllib.request.urlopen('https//google.com')
print(page.read())
```

9. Problem Set

Using the test API available at the url below, you will provide solutions using curl and python to retrieve and analyse the test data in this API

http://jsonplaceholder.typicode.com/

You should solve each of these problems twice.

1. First, run through them using curl and other command-line utilities.
2. Then do them again using the Python language. In your solution, you will need to parse the JSON responses and convert them into internal Python data structures such a list or dictionary

   This particular API endpoint requires a user agent to be set when making requests in order to work around a 403 response. You can do this using something like the following Python code

```
from urllib.request import Request, urlopen
from json import loads

url = 'http://jsonplaceholder.typicode.com/users'
req = Request(url, None, {
    'User-agent' : 'Mozilla/5.0 (Windows; U; Windows NT 5.1; de;
rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5'
})

data = loads(urlopen(req).read().decode("utf-8"))
```

1	How many users are returned by the API?	
2	How many posts have been made by user id 4?	
3	List the email addresses of the the commenters to post 1 in alphabetical order	
4	Which user has the most TODOs?	
5	How many thumbnails are there in album id 76?	
6	List the thumbnail identifiers (the trailing hex numbers in the URL) from question 5	
7	List the album titles for user id 4	
8	List the 5 albums with the lowest number of photos	
9	List the geo coordinates (lat and lng) for all users	

10	Which user has the most uncompleted TODOs?	