52957 -- DATA REPRESENTATION

5 Credit Module

Lecturer: Andrew Beatty

Assignment Schedule

* **Quick Assignments (30%): submit in Moodle**
* **GitHub Activity (10%)** - Looking at activity in the labs

I have created my repository;

<https://github.com/EtainUpton/dataRepresentation---52957-DATA-REPRESENTATION>

and

[**https://github.com/EtainUpton/52957---DATA-REPRESENTATION**](https://github.com/EtainUpton/52957---DATA-REPRESENTATION)

* **Project (60%)** – Deadline;

*The official deadline for the project is Friday the 13thDecember, but I will give an automatic extension to Friday 20th December for anyone who asks for it. My absolute “drop dead” deadline is Tuesday the 7th January, but that is if you are really, really stuck!*

**Indicative schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week** | **Topic** |  | **Week** | **Topic** |
| 1 | Introduction  XML and [HTML](https://learnonline.gmit.ie/mod/url/view.php?id=52690)  The DOM tree | DONE | 7 | Consolidate |
| 2 | Navigating the DOM tree with JavaScript | DONE | 8 DONE | Serving the API  And hosting |
| 3 | HTTP: URLs CURL  Using python to consume XML from the web | DONE | 9 DONE | Linking to Database on the server-side |
| 4 | JSON and RESTful API | DONE | 10 DONE | Pulling it all together |
| 5 | Consuming the API:  AJAX and JQUERY | DONE | 11 DONE | Review and revise |
| 6 | Consuming the API with Python | DONE |  |  |

**Week 1**

**Introduction Video –** watched.

### [XML HTML and DOM](https://learnonline.gmit.ie/course/view.php?id=1318#section-2)

The purpose of this week's topic is to:

* + - Acquaint you with XML and the DOM tree, XML is very straightforward once you get the hang of it.
    - Show you how to write basic [HTML](https://learnonline.gmit.ie/mod/url/view.php?id=52690), the tags I specifically what you to get are:
      1. The page tags (<[html](https://learnonline.gmit.ie/mod/url/view.php?id=52690)> <title><body>)
      2. <div>
      3. The table tags (<table> <tr><th><td>)
      4. The form tags (<form><input><button><select><option>)

***(Notes from Moodle – Andrew Beatty)***

**Review Andrew’s PowerPoint slides (on moodle)**

**Done:**

* **XML and DOM**
* **HTML**

<https://www.w3schools.com/html/>

‘Try it yourself’

Tables

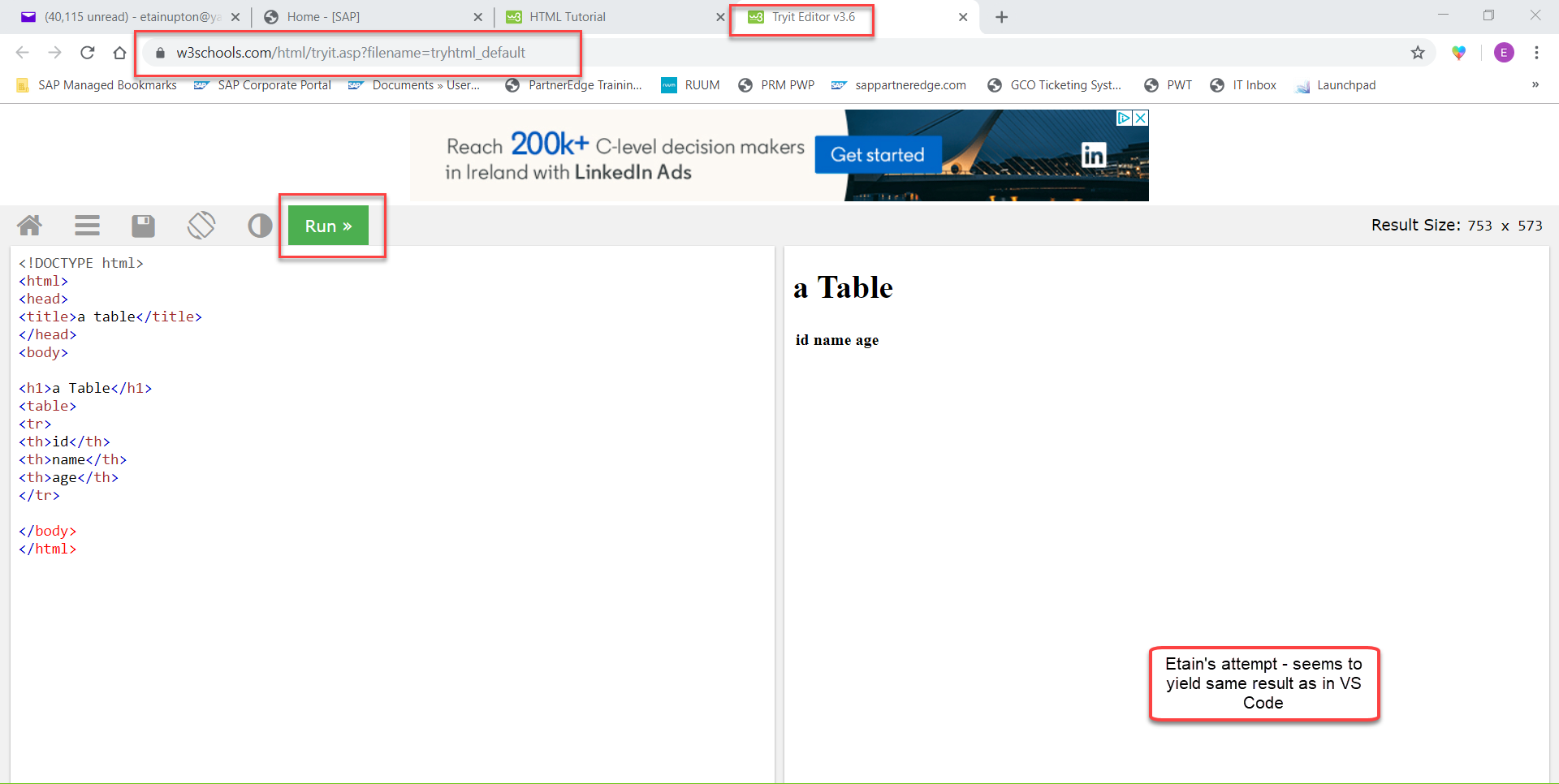
Forms

I can write my code on the ‘Try it yourself’ section (I think) or in VS Code.

Webpage in this vid.

Table @ 8mins

<!DOCTYPE html>  
<html>  
<head>  
<title>Page Title</title>  
</head>  
<body>  
  
<h1>This is a Heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>



*To write basic* [*HTML*](https://learnonline.gmit.ie/mod/url/view.php?id=52690)*, the tags I specifically what you to get are:*

* + - 1. The page tags (<[html](https://learnonline.gmit.ie/mod/url/view.php?id=52690)> <title><body>)
      2. <div>
      3. The table tags (<table> <tr><th><td>)
      4. The form tags (<form><input><button><select><option>)

### These may be handy for me;

### [References and extra reading](https://learnonline.gmit.ie/course/view.php?id=1318#section-20)

* W3Schools have some excellent tutorials on [html](https://learnonline.gmit.ie/mod/url/view.php?id=52690) and Javascript

<https://www.w3schools.com/html/>

<https://www.w3schools.com/js/default.asp>

* and on DOM

<https://www.w3schools.com/js/js_htmldom.asp>

* a blog on VS code and how to customise it

<https://dev.to/lampewebdev/my-web-development-vs-code-settings-theme-extensions-tips-and-tricks-1324>

\*\*\*

The main things I will be looking at in Github are:

1. The Labs
2. The project I will be asking you to do in about 4 weeks (the labs are working towards this)
3. Any other small projects I will be asking of you.

You do not need to save the exercises there, but you may if you would like me to look at them

**Andrew has supplied the code in GitHub repository;** [**https://github.com/andrewbeattycourseware/dataRepresentation/blob/master/code/week01/table.html**](https://github.com/andrewbeattycourseware/dataRepresentation/blob/master/code/week01/table.html)

**Week 2**

PowerPoint 2 downloaded

Boolean (True/False)

**Week 3:**

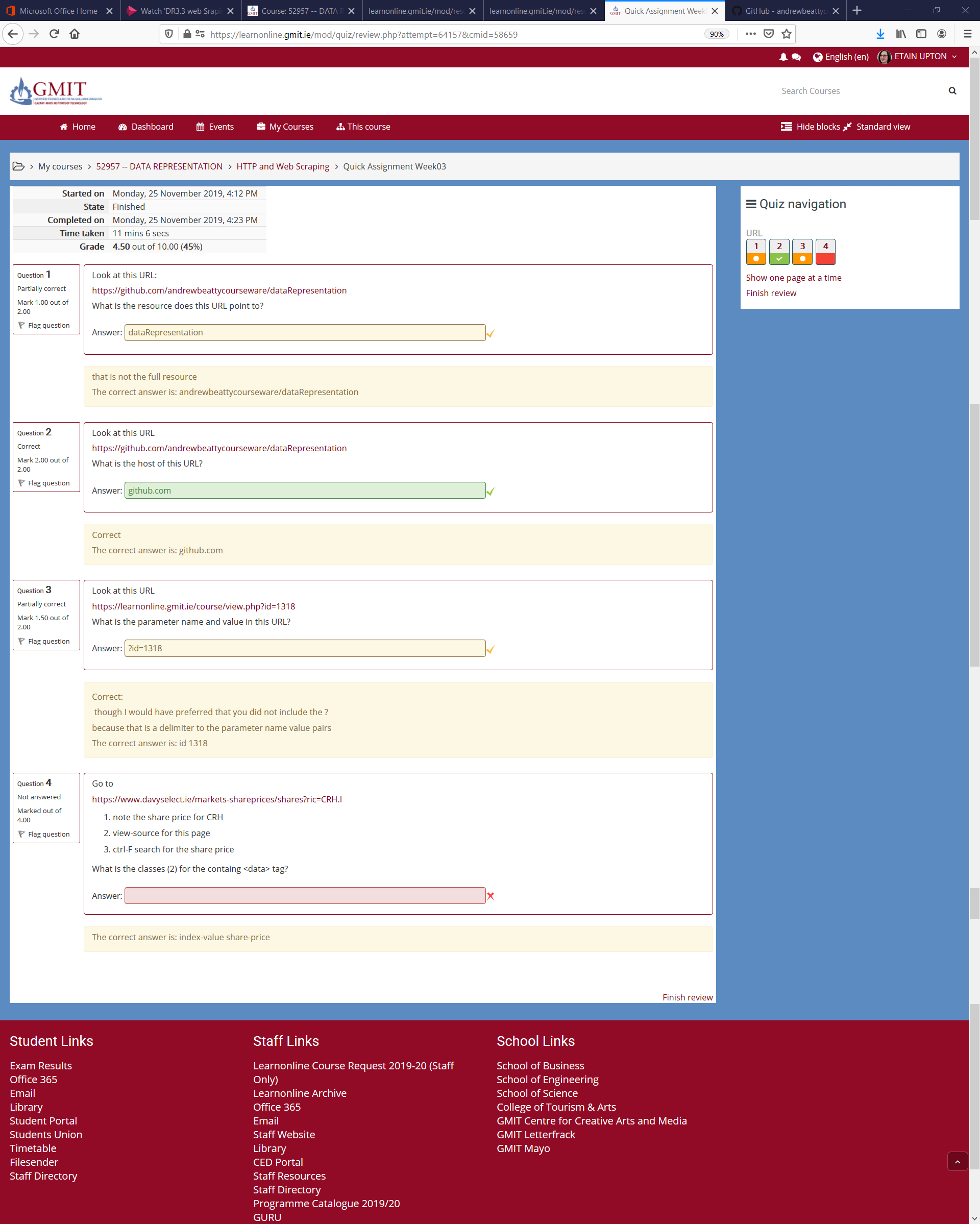
If I want to use beautifulsoup it needs to be downloaded.

Lecture 2: Web Scraping

Video: [Part 1](https://web.microsoftstream.com/video/4fcf53da-d337-4d13-9187-1206cc1bc721?list=studio)                  WATCHED              [Slides](https://learnonline.gmit.ie/pluginfile.php/119922/mod_label/intro/DR3.3%20web%20Sraping.pptx) DOWNLOADED

Video: [Part 2](https://web.microsoftstream.com/video/48cef90d-b9ef-441e-ab2a-080b7298831a?list=studio)                WATCHED              [Slides](https://learnonline.gmit.ie/pluginfile.php/119922/mod_label/intro/DR3.3%20web%20Sraping%20part%202.pptx) DOWNLOADED

**I attempted** [Quick Assignment Week03](https://learnonline.gmit.ie/mod/quiz/view.php?id=58659) **twice – I got 45% in first attempt and 100% in second attempt.**



**Week 4:**

**I skipped ahead to week 9**

**Week 9:**

**Linking to Database**

**Lectures and Labs**

1. Databases

Lecture – 17 mins - watched

Sides – downloaded

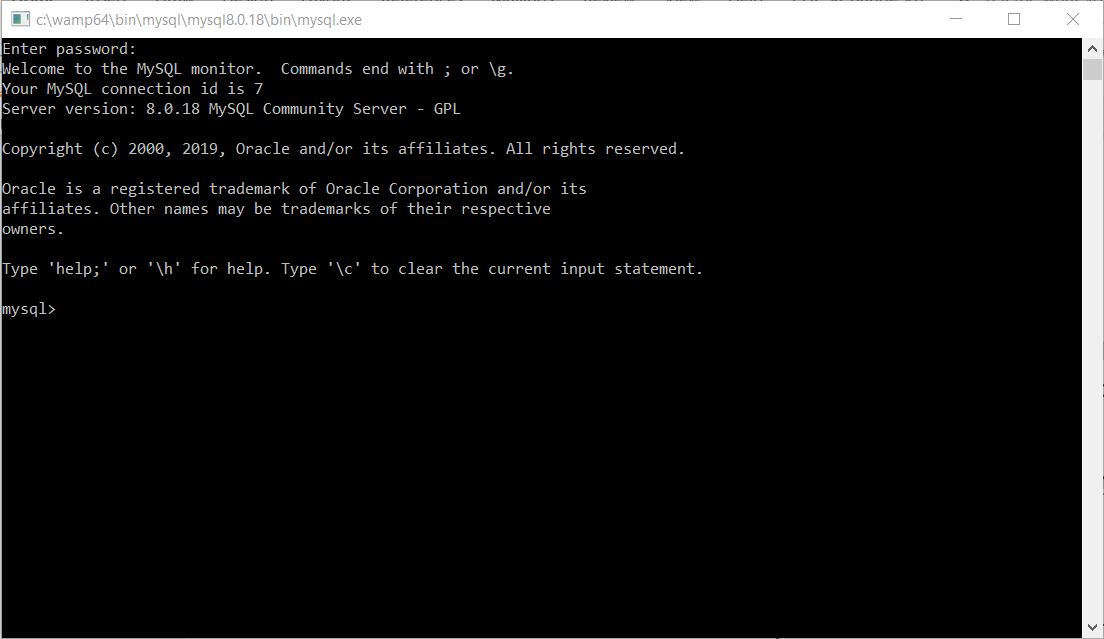
Lab - DONE

<http://www.wampserver.com/en/>

I followed the steps to install wamp. I already had Visual C++installed on my device so didn’t need to install it again. I downloaded 64 bit wamp. Once downloaded I typed ‘Wamp server’ into the search functionality on my device.

Note: To access MySQL in wamp for this module access it via Wamp My SQL Console.

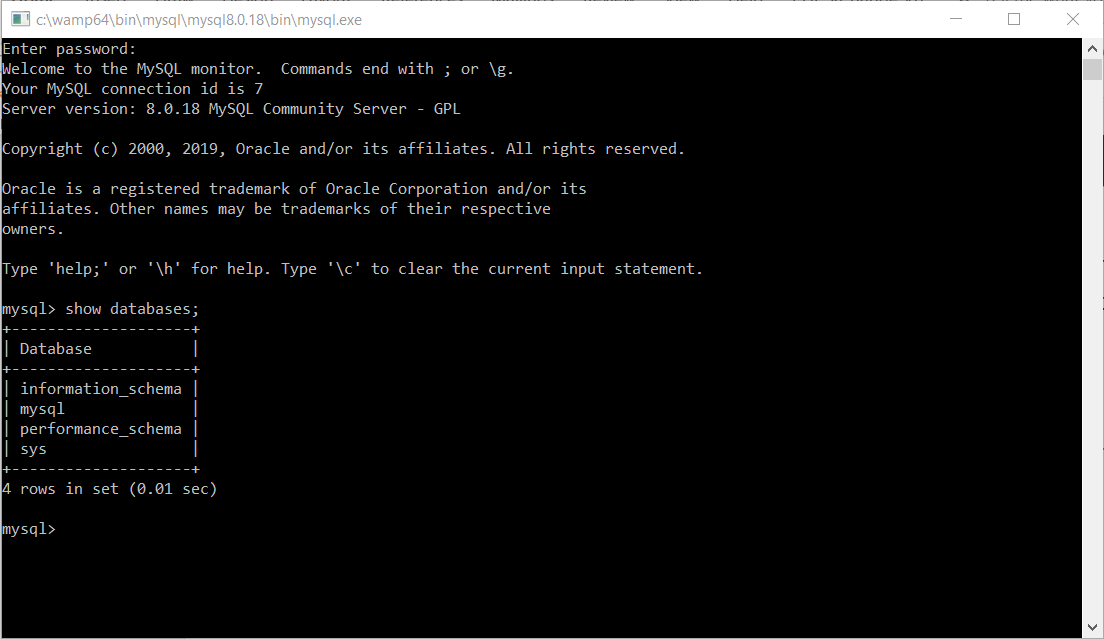
Root. Password is blank, simply click enter.



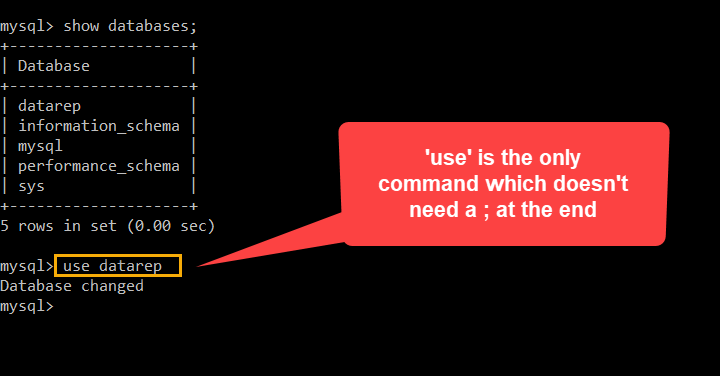
To access phpMyAdmin – username is root, password is blank. This will allow me to interact with the database. I was instructed by the download wizard to restart all services when Wamp was successfully downloaded to change the icon from red to green. I did this, now it is green; 

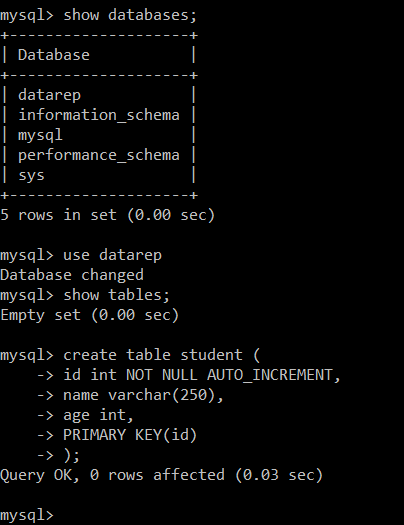
There is an SQL Tutorial on W3Schools <https://www.w3schools.com/sql/>

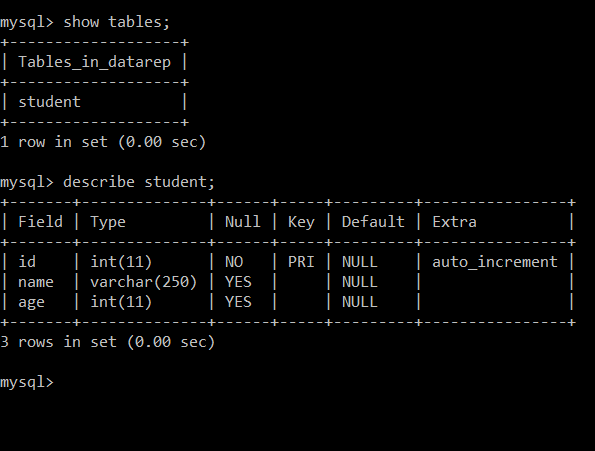
To see the databases I have – show databases;



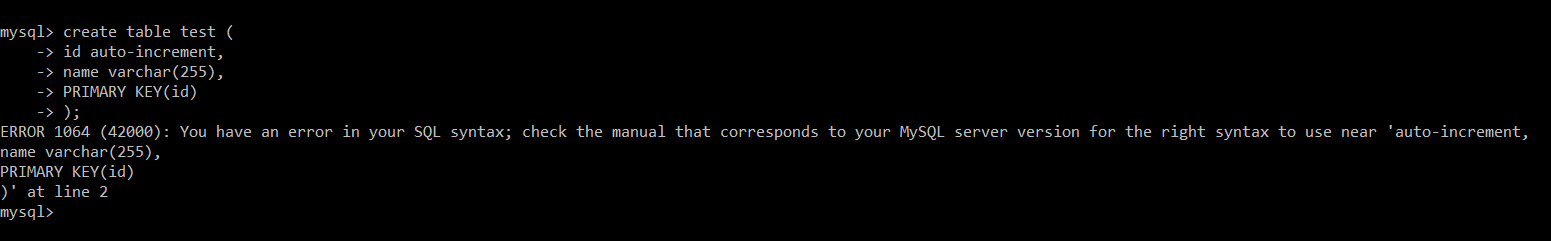


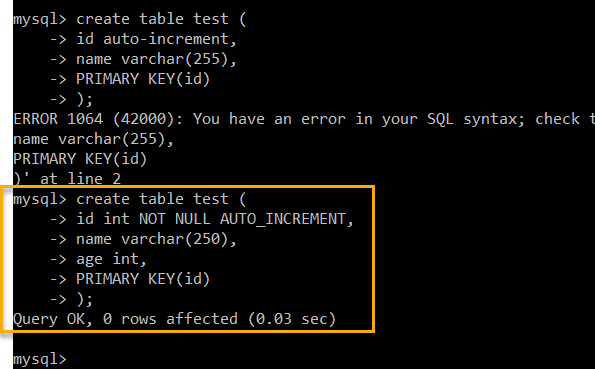




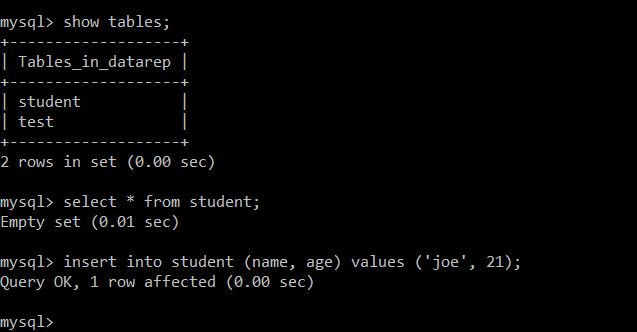


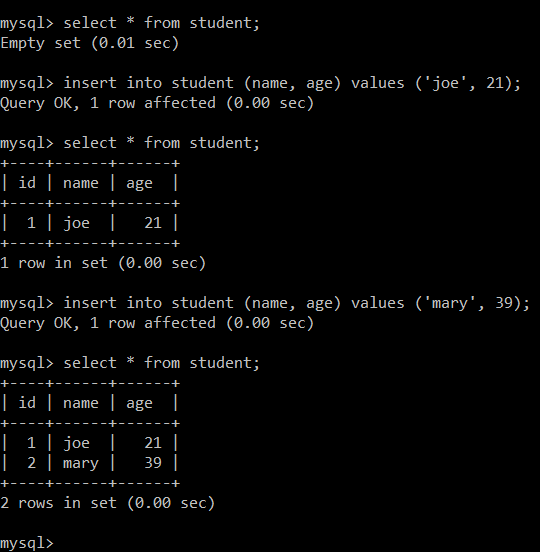
Lecturer deliberately included an error;

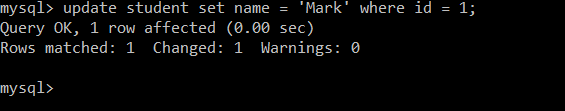


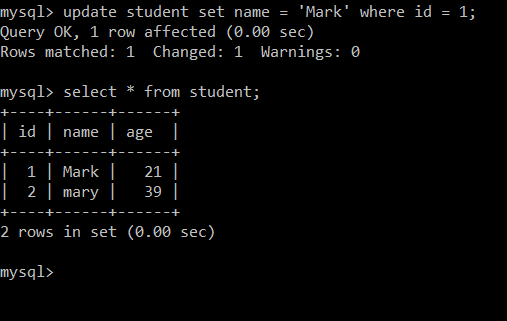


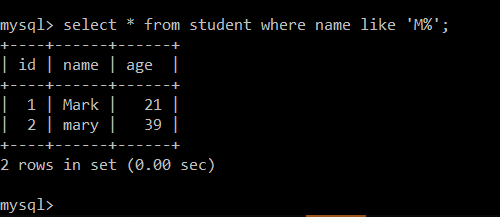
This made the table.

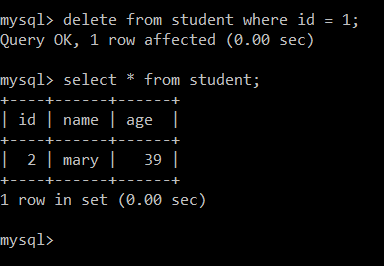




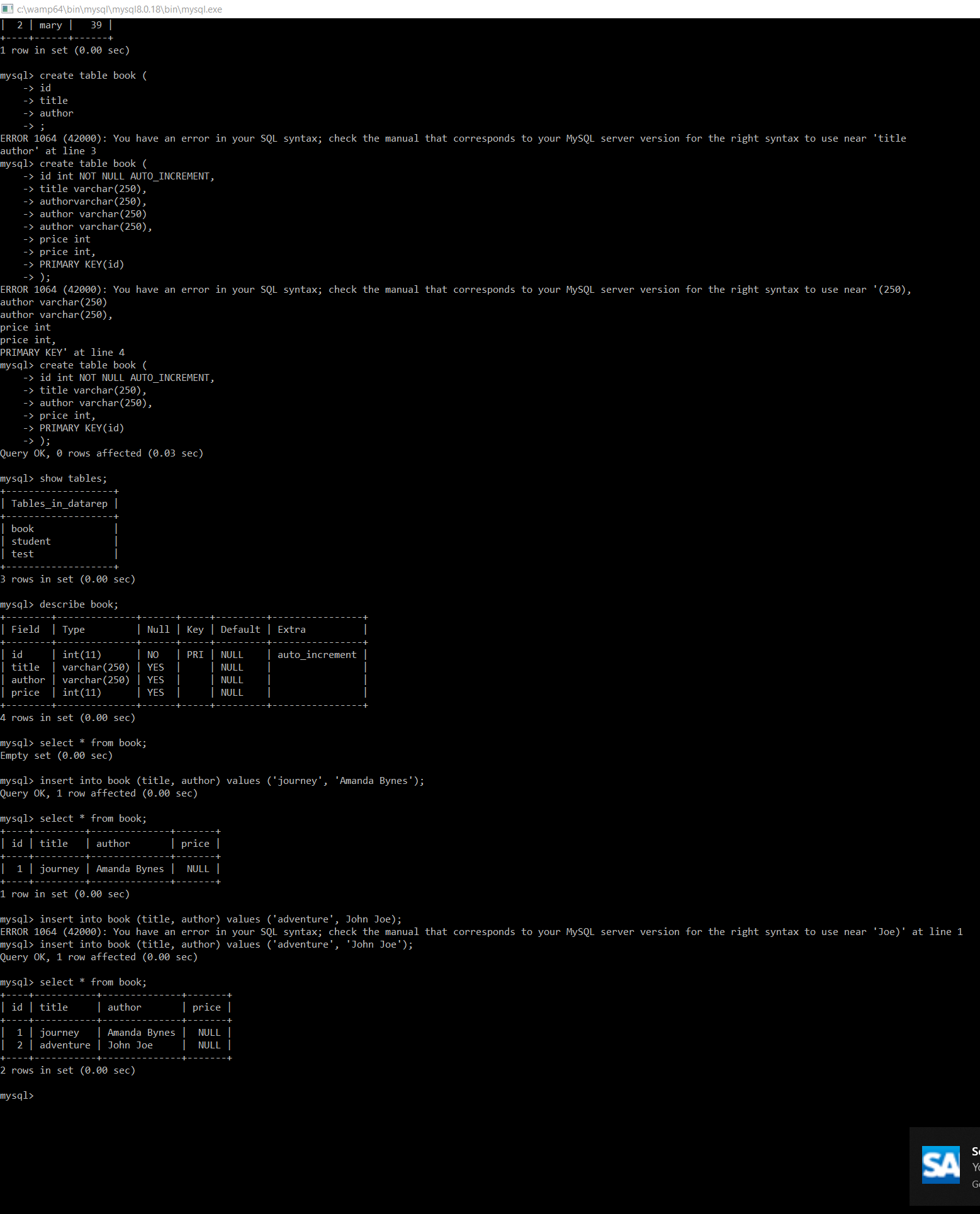








LAB: 2.



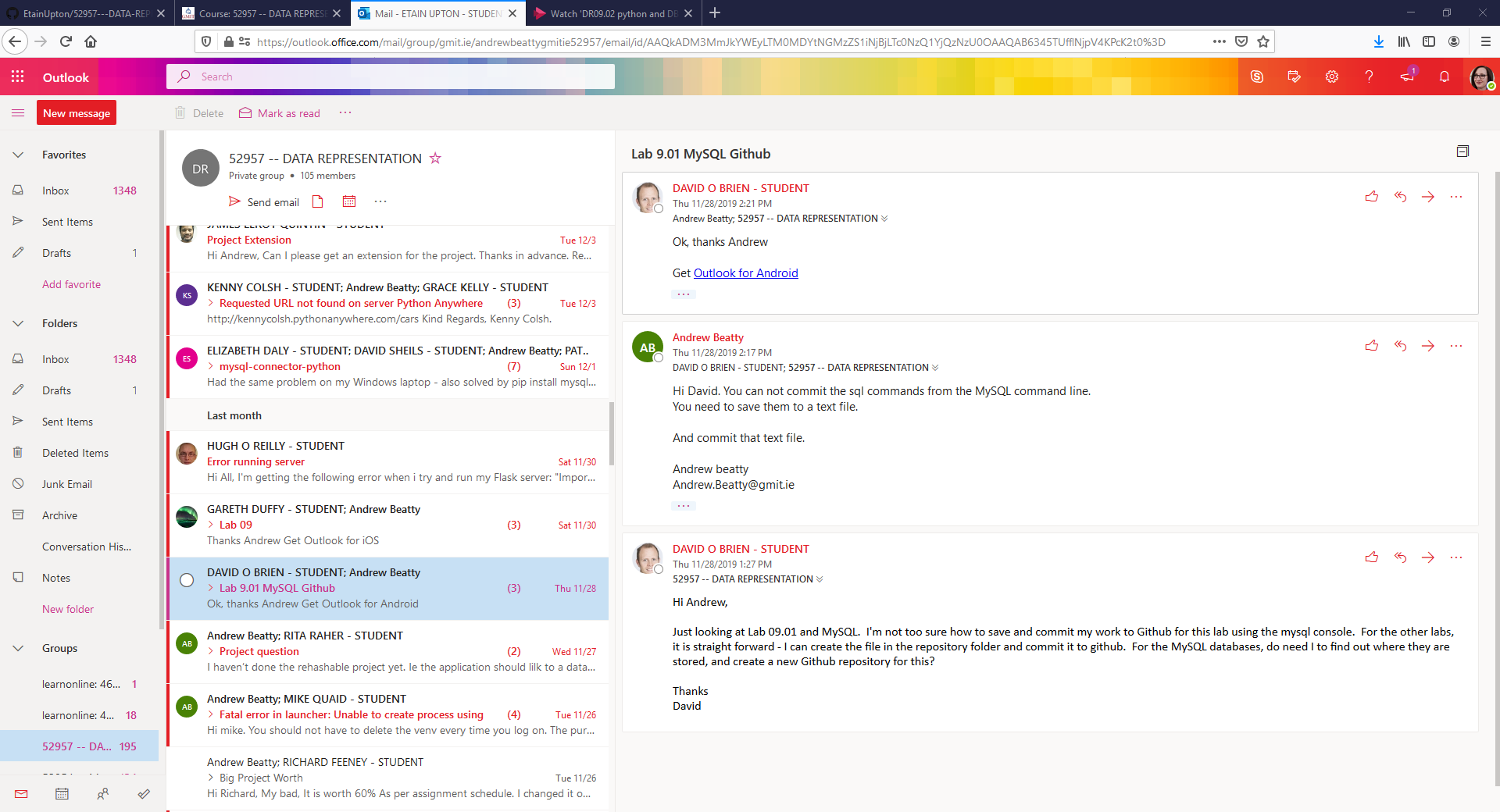
1. Python and DB

Lecture – 17 mins – watched this but encountered an issue following Andrew’s instructions at 6mins.

Slides – Downloaded

Lab – I should try this as it may help me figure out how to complete the steps shown by Andrew in the video.

Note: Insert root and blank as the password into my configuration file.



**Week 4:**

**I decided to go back to week 4 as there is a ‘Quick assignment’ in week 5 based on content from week 4 and 5 which I may be able to complete and pick up some marks.**

|  |
| --- |
| JSON and RESTful API – Week 4 DONE |
| Consuming the API: - Week 5  AJAX and JQUERY |

Week 4;

Lab mainly based on 3rd lecture (AJAX)

**Lectures**

1. HTTP: L[ecture](https://web.microsoftstream.com/video/e7827ff1-1f27-45f7-b831-198d426d06f7?list=studio)       DONE     [Slides](https://learnonline.gmit.ie/pluginfile.php/124716/mod_label/intro/DR4.2%20HTTP2%20methods%20and%20response%20codes%20slides.pptx)  DOWNLOADED
2. JSON: [Lecture](https://web.microsoftstream.com/video/4a51391c-856a-4237-964e-12c185410dfc?list=studio)       DONE          [Slides](https://learnonline.gmit.ie/pluginfile.php/124716/mod_label/intro/DR4.3%20JSON%20slides.pptx) DOWNLOADED
3. AJAX:  [Lecture](https://web.microsoftstream.com/video/fab4adbc-f990-4164-b731-75317fc891e4?list=studio)     DONE            [Slides](https://learnonline.gmit.ie/pluginfile.php/124716/mod_label/intro/DR4.4%20AJAX%20JQuery%20slides.pptx) DOWNLOADED
4. REST: [Lecture](https://web.microsoftstream.com/video/830aa515-058c-4a95-a14d-95518e511fcf?list=studio)       DONE                 [Slides](https://learnonline.gmit.ie/pluginfile.php/124716/mod_label/intro/DR4.5%20Rest%20slides.pptx) DOWNLOADED
5. HTTP notes

An Example of a ‘Get’ : You tube search

Example of a ‘Post’ – uploading something to my respository on Github.

1. JSON

Mentions the lab we will need to do to extract the euro value for bitcoin

1. AJAX

Used to get information from a web server.

You will need this magic line of code in the header of any page that uses Jquery

This loads the JQuery library into the browser for the page;

<head>

        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

  </head>

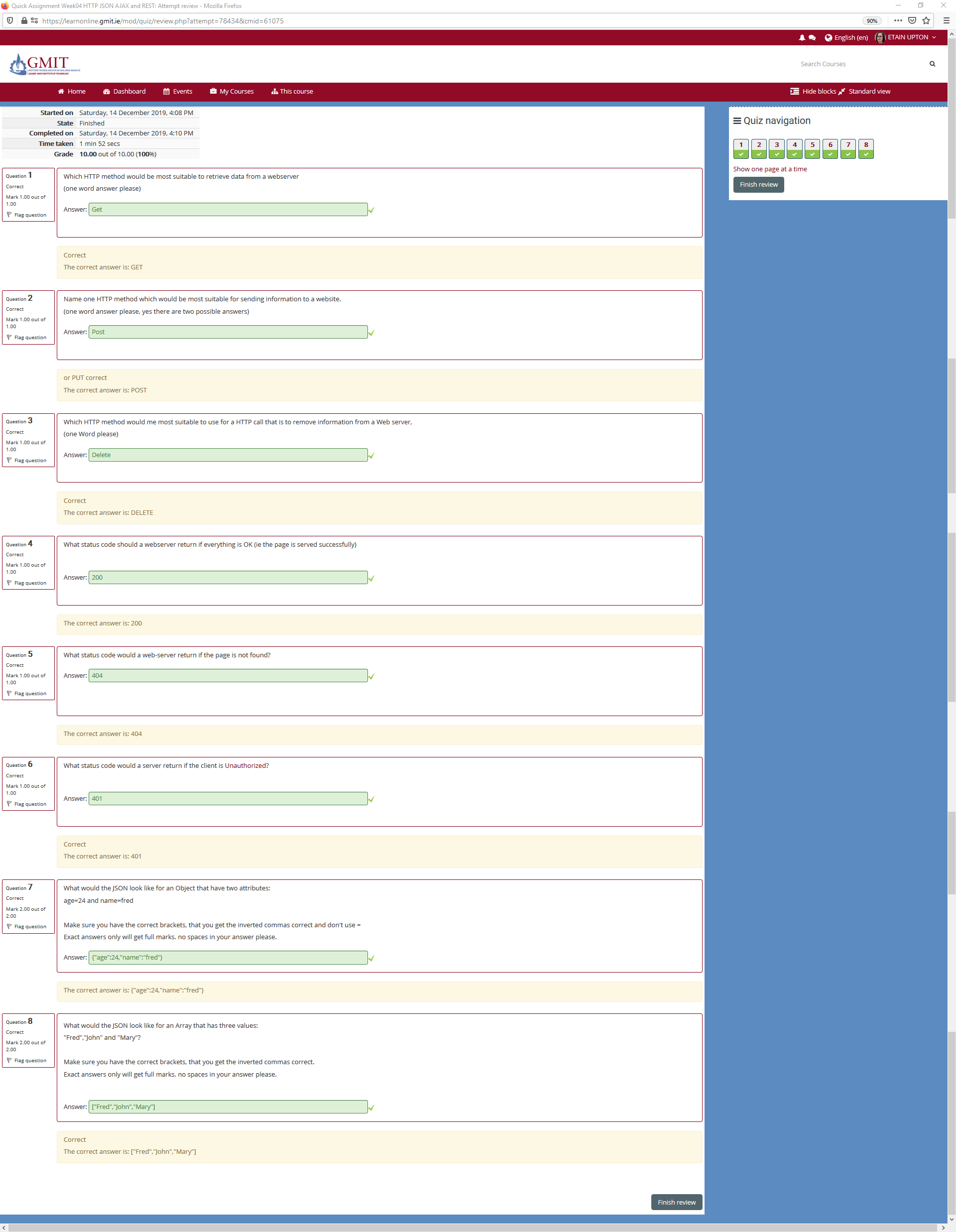
Lab mentioned at 5.59mins

1. REST:

An example of a rest API – getting emails from a server.

Stateless – each request should be treated as a separate transaction.

I completed the Lab (Lab 04) & uploaded it to my GitHub, I also completed the Quick assignment;



Next week: create a web page that consumes a RESTful API.

**Week 5**

AJAX and REST - Using AJAX to consume a RESTful API

NB I need to create a web page that consumes a RESTful API for my project.

Week 2 – car viewer… is mentioned – I may need to review this. I should be able to link the lab from week 2 Carviewer.html (which I haven’t done) to an app-server.

I have watched ‘Overview of week05’ video

#### Lectures and Labs

**Curl**:             [Updated Lecture](https://web.microsoftstream.com/video/849a0f66-f734-4e3c-b5f2-9680b6eb36eb?list=studio)    DONE  [Slides](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/DR5.2%20Curl.pptx)  DOWNLOADED   [Lab 05.1](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/Lab05.1%20CURL.pdf) DONE

Note on calling CURL: The windows console does not handle ', it only has ". This means that when we are passing data it needs to be in "".

Any inverted commas inside these inverted commas need to be escaped, so JSON would look something like.

"\"name\":\"value\""

YUCK!!!!!!

**Flask**:            [Lecture](https://web.microsoftstream.com/video/74148de3-6fc7-4140-b0bc-b0db2c3d896b?list=studio)      DONE   [Slides](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/DR5.3%20Flask.pptx)  DOWNLOADED       [Lab 05.2](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/Lab05.2%20flask1.pdf)      need to do this (demo’d in video0029

Note: FLASK is does not come with anaconda, so you will have to install it using PIP.      pip install flask     [click here if that does not work for you](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/Lab05-if%20you%20are%20having%20issues%20with%20Flask.pdf)

**AJAX**:             [Lecture](https://web.microsoftstream.com/video/85608962-2580-4b7f-86ac-d62026f20a0e?list=studio)                          [Slides](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/DR5.4%20AJAX%20and%20REST.pptx)    DOWNLOADED      [Lab 05.3](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/Lab05.3%20AJAX%20calls%20to%20REST.pdf?time=1571859434326)

**Challenge**:   [Lab Requirements](https://learnonline.gmit.ie/pluginfile.php/128937/mod_label/intro/Lab05.4%20connect%20carviewer%20to%20API.pdf)       [Video looking at requirements](https://web.microsoftstream.com/video/e1522fdc-c520-4262-b6fe-c22c4298f4a7?list=studio) (poor sound)      Solution on GitHub

**Curl** – allows you to get URL’s down from the internet.

Curl + URL

-X is a ‘put’ method

**Flask – stopped here**

Flask is an app server.

An App Server provides dynamic content.

.py in VS Code

I suspect what Andrew does in exe I can do in VS Code

Here Andrew has prepared the code – created an app-server to store cars that can be used with carviewer.html from week 02

<https://github.com/andrewbeattycourseware/dataRepresentation/blob/master/code/week05-REST/server/b_restserver.py>

I got stuck here.

I decided to go back and complete labs 1 & 2.

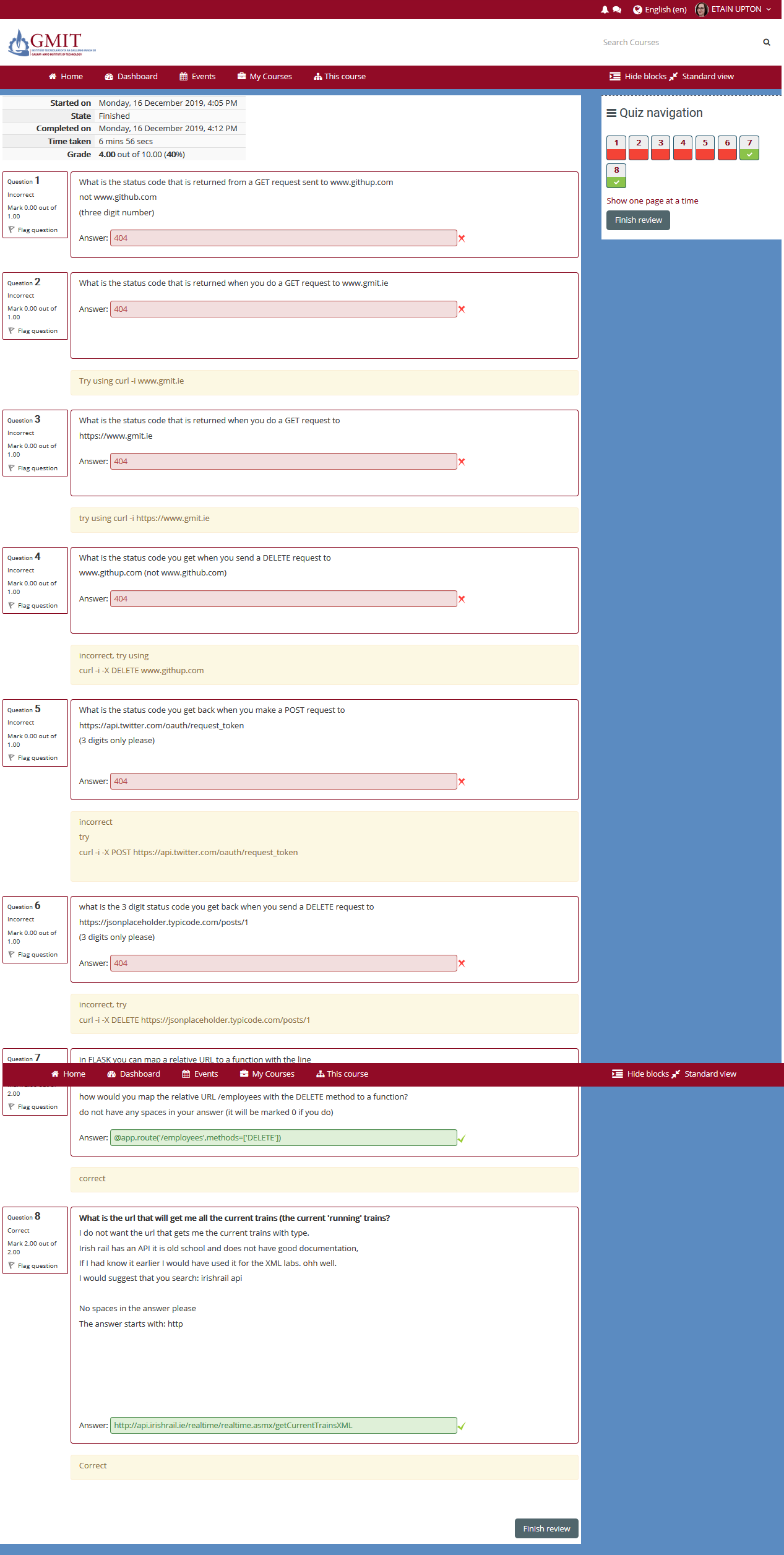
1 – completed

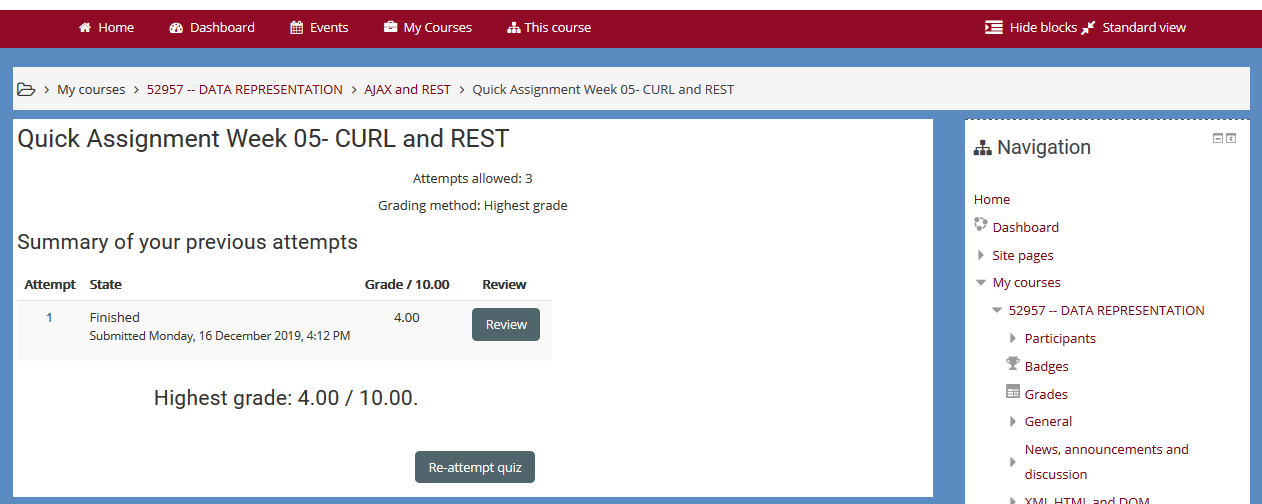
2 – in progress.

Tomorrow: Try to complete lab 2 and the rest of week 5 including the quick assignment

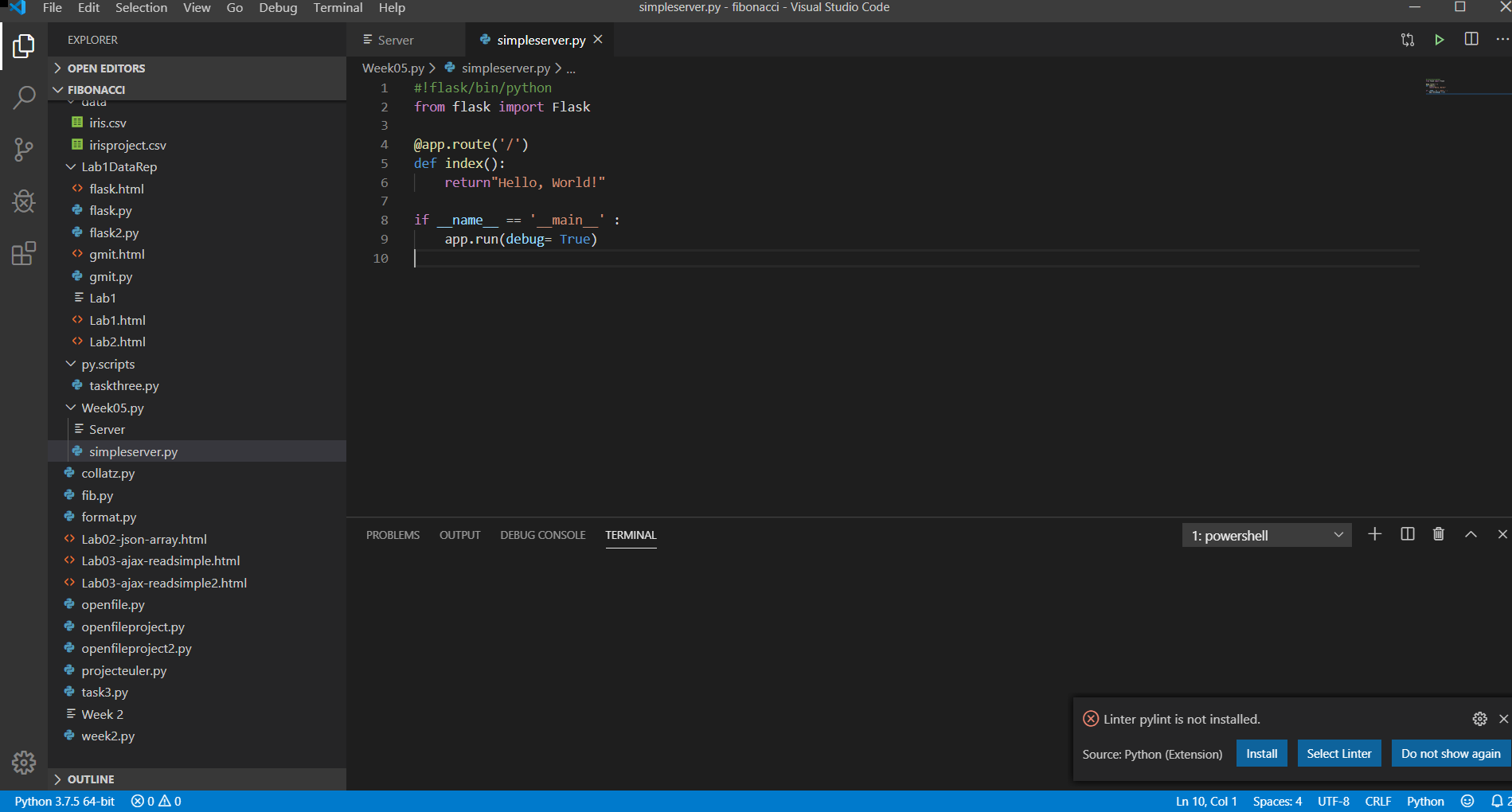
NB for Lab 2: <https://learnonline.gmit.ie/pluginfile.php/114853/mod_resource/content/2/exercises.pdf> and answers; <https://github.com/andrewbeattycourseware/dataRepresentation/tree/master/code/week02/exercise>

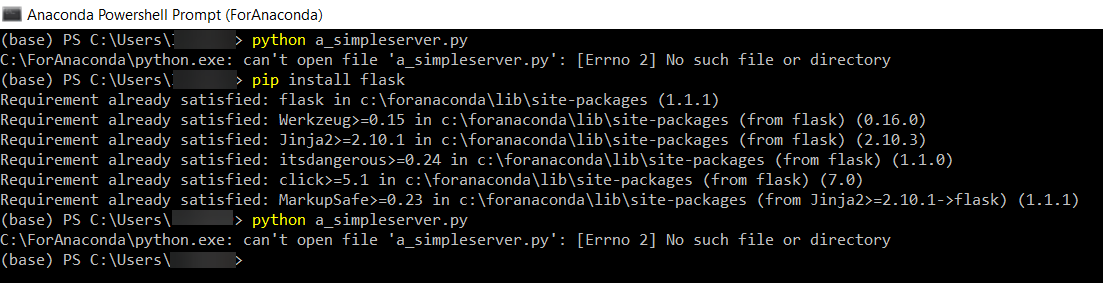
Quick assessment;





I ran into issues following the *Lab 05.2 Flask* instructions;





As shown above, I continued to get the error ‘C:\ForAnaconda\python.exe: can't open file 'a\_simpleserver.py': [Errno 2] No such file or directory’.

I struggled to progress past this.

I moved on to **week 8** as I am progressing slower than I anticipated due to the issue mentioned, and the deadline is approaching so I wanted to do as much as possible to fulfil the requirements for this project.

This lecture describes how to create the Rest API & write the code for the server.

Andrew mentioned some requirements for this;

***Step 1: design the API***

* *An app server that has a RESTful interface to provide CRUD operations for one database table.*
* *Crud* = Create, Read, Update, Delete.
* Andrew instructed that for this demo he would pick something at random like a book, and that I should choose my own entity.
* *A Book will have*
  1. *An id (Integer, auto increment) KEY i.e. this will be the unique identifier*
  2. *A title*
  3. *An Author*
  4. *A price (integer, the price in cent)*
  5. *There are other attributes it could have ISBN (this could have been the unique identifier, I choose to use an ID instead to make this as general as possible.*
* For my project, instead of a book I chose to base my example on a dog.
  1. Instead of an ID = A microchip ID number (example 12345). Serve as Unique Identifier.
  2. Instead of a title = Name, e.g. Fluffy.
  3. Author = dog owner’s name, e.g. John Doe.
  4. Price = Value of dog.

These are the 4 entities I will use.

* *This is just a CRUD interface so this will be similar to app server we created in week05.*

Note: I struggled with the code in week 05 so I am hoping I will be able to follow Andrew’s steps from this vdeo posted to Moodle by Andrew without issue (<https://web.microsoftstream.com/video/f9835a3d-8628-43ee-bb64-d00a3f55fcc7?list=studio&referrer=https:%2F%2Flearnonline.gmit.ie%2Fcourse%2Fview.php%3Fid%3D1318>).

* *It will need to allow us*
  + *Get all books*
  + *Get a book by id (might not actually be needed, but I am putting it in anyway)*
  + *Create a book*
  + *Update a book*
  + *Delete a book*
* *Other applications may require other functionality in the interface. And we can always add functionality later.*
* *NOTE: no code yet*

I copied Andrew’s template for an interface and altered it to fit the purposes for my project submission (highlighted in yellow);

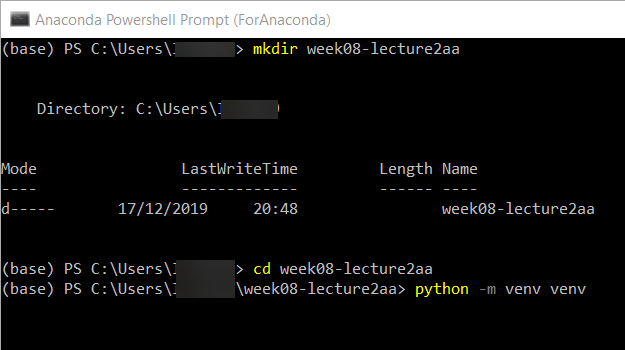
***Proposed Interface***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action** | **Method** | **URL** | **Sample params** | **Sample return** |
| Get all | GET | /books  /dogs | none | [  {…},{…},{…}  ] |
| Find by id | GET | /books/id  /dogs/id | none | { ”id":”1", ”title":”xxx",  ”author":”xxx", "price":3000  }  { ”id":”1", ”name":”xxx",  ”owner":”xxx", "value":3000  } |
| Create | POST | /books  /dogs | { ”title":”xxx",  ”author":”xxx", "price":3000  }  { ”name":”xxx",  ”owner":”xxx", "value":3000  } | { ”id":”1", ”title":”xxx",  ”author":”xxx", "price":3000  }  { ”id":”1", ”name":”xxx",  ”owner":”xxx", "value":3000  } |
| Update | PUT | /books/id  /dogs/id | {  "price":3000  }  {  "value":3000  } | { ”id":”1", ”title":”xxx",  ”author":”xxx", "price":3000  }  { ”id":”1", ”name":”xxx",  ”owner":”xxx", "value":3000  } |
| delete | DELETE | /books/id  /dogs/id | none | {  "done":true  }  {  "done":true  } |

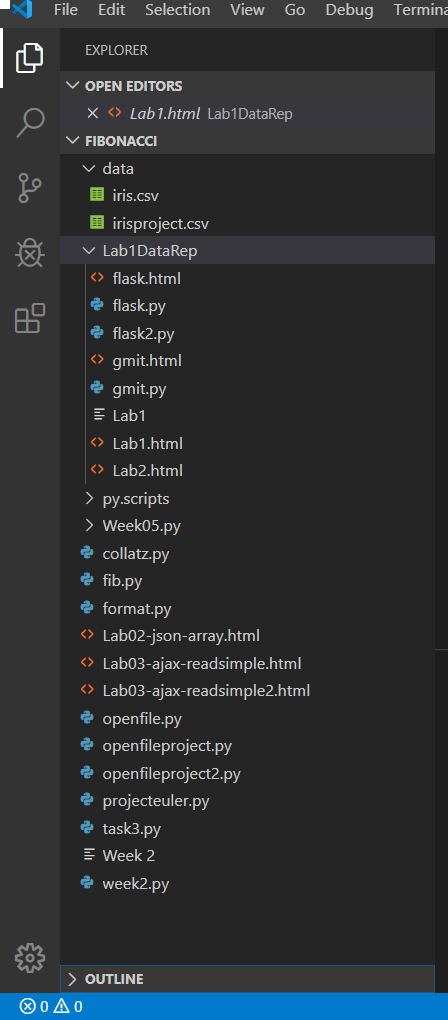
***Step2 : make app-server with skeleton functions***

* *OK Now we start to code*
* *Make a very basic appserver, test it.*
* *Add a function (from the table above) and URL map for each of the functions we require in our interface. Each function should just return text saying what they are. Test them using CURL*

I tried to follow Andrew’s instruction in the video, but I run into issues when I try to create a directory using Cmder;

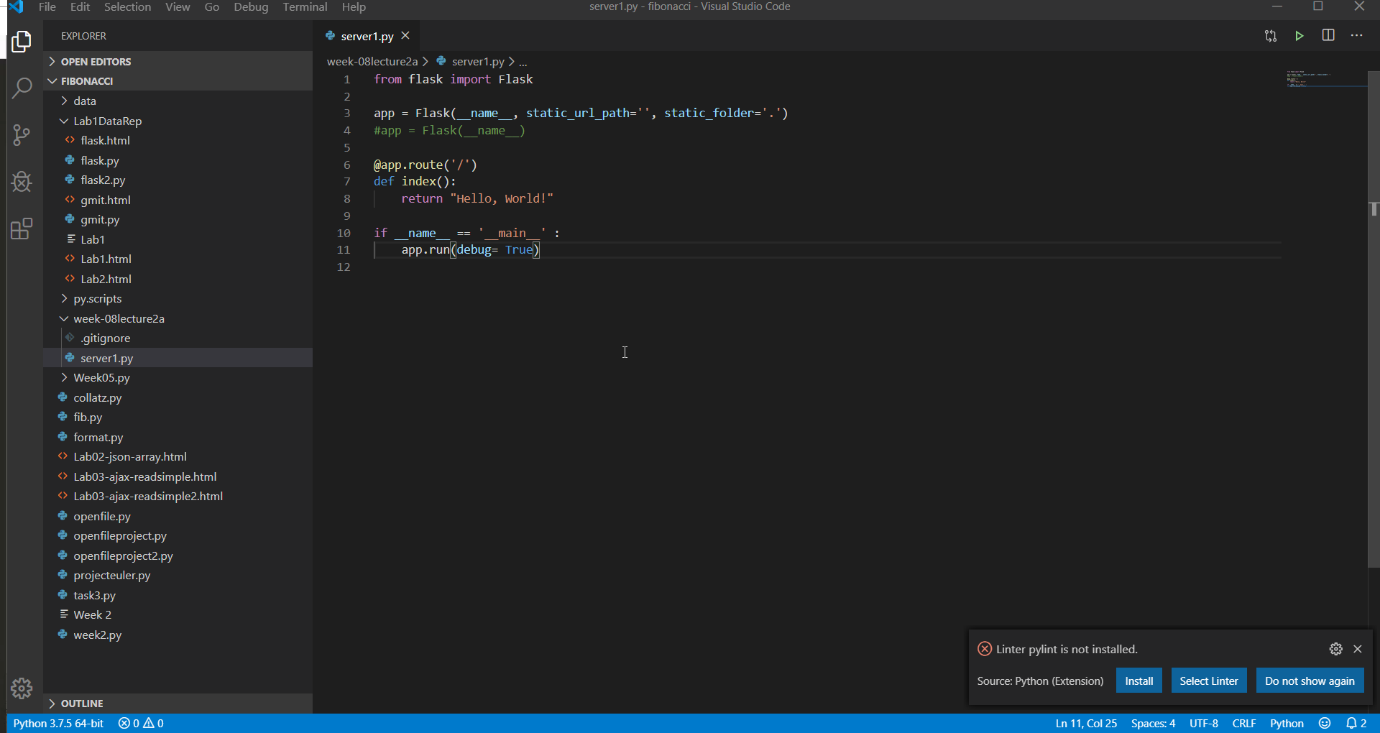


This does not successfully create the directory in my VS code;



I decide to create it myself instead in VS code.

I follow Andrew’s code;



from flask import Flask

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

#app = Flask(\_\_name\_\_)

@app.route('/')

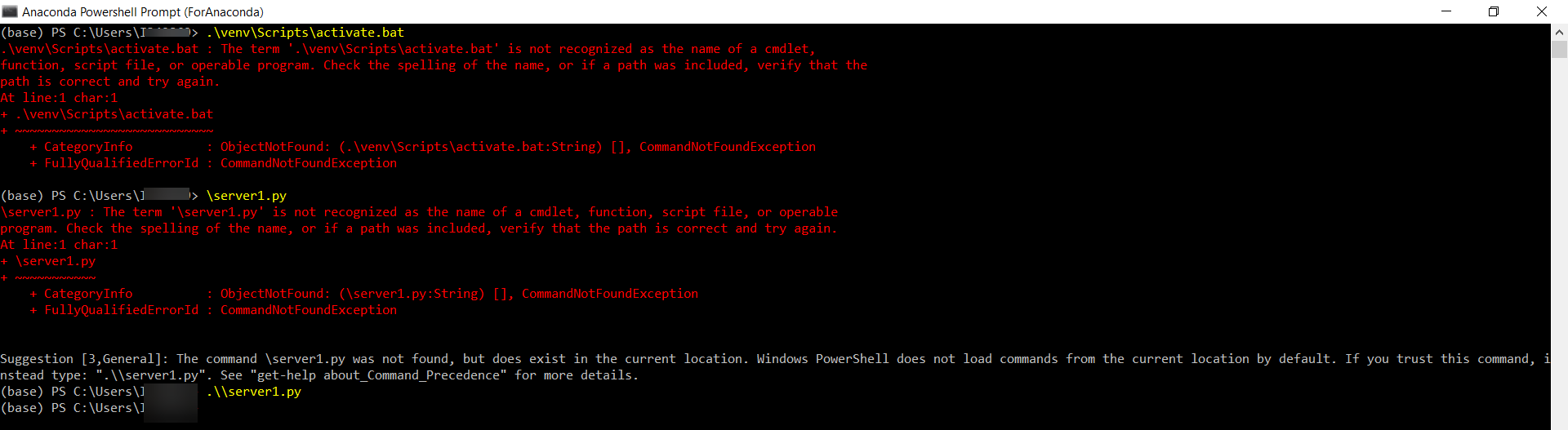
def index():

    return "Hello, World!"

if \_\_name\_\_ == '\_\_main\_\_' :

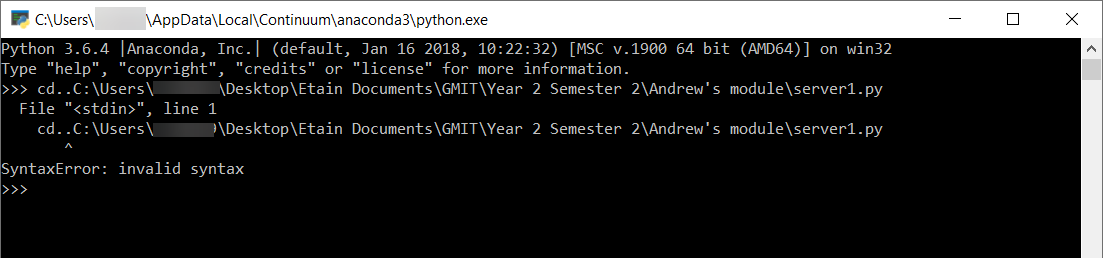
    app.run(debug= True)

However, I encounter issues once again;



I decide to try and continue on with VS code.

More errors;



I copied Andrew’s code example to create the interface;

from flask import Flask

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return "in getAll"

@app.route('/dogs/<int:id>')

def findById(id):

    return "in find By ID for id "+str(id)

@app.route('/dogs', methods=['POST]')

def create():

    return "in create "

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Continuing to follow Andrew’s steps on the PowerPoint and in his video;

***STEP3: write the code for each of the functions***

* *For this stage we will not link to a database, we will just store the books* ***(or dogs)*** *in a list like we did in week 05.*
* *Do the get all first that should be the easiest, TEST if with CURL*
* *Do find by id, TEST IT*
* *Do create, TEST IT*
* *Do update, TEST IT*
* *Do delete, TEST IT*

from flask import Flask, jsonify

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    return "in find By ID for id "+str(id)

@app.route('/dogs', methods=['POST]')

def create():

    return "in create "

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

from flask import Flask, jsonify

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    return "in create "

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Note: I could not test as I was going along, so I was unable to copy my versions of #curl <http://127.0.0.1:5000/dogs/2> to paste into my code.

18.12.2019;

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

In Cmder Andrew demonstrated that as this is in windows he needed to escape the double quotes. He also needed to change the header: curl -i -H “Content- Type:application/json”

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    global nextId

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Final;

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    global nextId

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Final 2;

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    global nextId

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    foundDogs = list(filter(lambda t: t['id']== id, dogs))

    if (len(foundDogs) == 0):

        abort(404)

    foundDog = foundDogs[0]

    if not request.json:

        abort(400)

    reqJson = request.json

    if reqJson['Name']:

        foundBook['Name'] = reqJson['Name']

    if reqJson['Owner']:

        foundBook['Owner'] = reqJson['Owner']

    if reqJson['Value']:

        foundBook['Value'] = reqJson['Value']

    return jsonify(foundDog)

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Last;

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    global nextId

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    foundDogs = list(filter(lambda t: t['id']== id, dogs))

    if (len(foundDogs) == 0):

        abort(404)

    foundDog = foundDogs[0]

    if not request.json:

        abort(400)

    reqJson = request.json

    if 'Name' in reqJson:

        foundDog['Name'] = reqJson['Name']

    if 'Owner' in reqJson:

        foundDog['Owner'] = reqJson['Owner']

    if 'Value' in reqJson:

        foundDog['Value'] = reqJson['Value']

    return jsonify(foundDog)

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Final;

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    global nextId

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    foundDogs = list(filter(lambda t: t['id']== id, dogs))

    if (len(foundDogs) == 0):

        abort(404)

    foundDog = foundDogs[0]

    if not request.json:

        abort(400)

    reqJson = request.json

    if 'Value' in reqJson and type(reqJson['Value']) is not int:

        abort(400)

    if 'Name' in reqJson:

        foundDog['Name'] = reqJson['Name']

    if 'Owner' in reqJson:

        foundDog['Owner'] = reqJson['Owner']

    if 'Value' in reqJson:

        foundDog['Value'] = reqJson['Value']

    return jsonify(foundDog)

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    return "in delete for id "+str(id)

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Last final;

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    global nextId

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    foundDogs = list(filter(lambda t: t['id']== id, dogs))

    if (len(foundDogs) == 0):

        abort(404)

    foundDog = foundDogs[0]

    if not request.json:

        abort(400)

    reqJson = request.json

    if 'Value' in reqJson and type(reqJson['Value']) is not int:

        abort(400)

    if 'Name' in reqJson:

        foundDog['Name'] = reqJson['Name']

    if 'Owner' in reqJson:

        foundDog['Owner'] = reqJson['Owner']

    if 'Value' in reqJson:

        foundDog['Value'] = reqJson['Value']

    return jsonify(foundDog)

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    foundDogs = list(filter(lambda t: t['id']== id, dogs))

    if (len(foundDogs) == 0):

        abort(404)

    dogs.remove(foundDogs[0])

    return jsonify({"done":True})

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

Note to Etain: Stopped at 30mins on video on 17.12.19

<https://web.microsoftstream.com/video/f9835a3d-8628-43ee-bb64-d00a3f55fcc7?list=studio&referrer=https:%2F%2Flearnonline.gmit.ie%2Fcourse%2Fview.php%3Fid%3D1318>

Slide 6 (step 3)

**Week 06: 18.12.2019**

**requests**:                    [Lecture](https://web.microsoftstream.com/video/0b26c2ed-9eac-48bb-90e9-62c13a0b8a6d?list=studio)                   [Slides](https://learnonline.gmit.ie/pluginfile.php/133025/mod_label/intro/DR6.2%20python%20and%20APIs.pptm)   Downloaded   [Lab 6.01](https://learnonline.gmit.ie/pluginfile.php/133025/mod_label/intro/Lab06.1%20use%20python%20to%20access%20server.pdf)

**Authentication**:         [Lecture](https://web.microsoftstream.com/video/b29b3e98-c429-4ff8-8f03-851b61ab0ef4?list=studio)                   [Slides](https://learnonline.gmit.ie/pluginfile.php/133025/mod_label/intro/DR6.3%20API%20keys%20and%20OAuth.pptm)                [lab 6.02](https://learnonline.gmit.ie/pluginfile.php/133025/mod_label/intro/Lab06.2%20API%20keys.pdf?time=1572969954446)

**APIs and packages:  L**[**ecture**](https://web.microsoftstream.com/video/3d9a9603-a7cb-4bcc-b7b8-85b25a8370c4?list=studio)[**Slides**](https://learnonline.gmit.ie/pluginfile.php/133025/mod_label/intro/DR6.4%20using%20packages.pptx)[**Lab 6.03**](https://learnonline.gmit.ie/pluginfile.php/133025/mod_label/intro/Lab06.3%20using%20the%20package.pdf)

Python consuming APIs

* We can use python to do anything that we can do in CURL.
* There are many web services that have APIs, that we can consume.
  + Github
  + Gmail
  + The server we made for the carviewer

What is needed;

* Read and write JSON (usually to DICT)

import json

json.dump()

json.load()

* Send HTTP requests (with requests package)

import requests

requests.get(url)

* Read and write to files

f = open("../../week02/carviewer2.html", "r")

html = f.read()

* Write to excel file

from xlwt import \*

w = Workbook()

w.save('cars.xls')

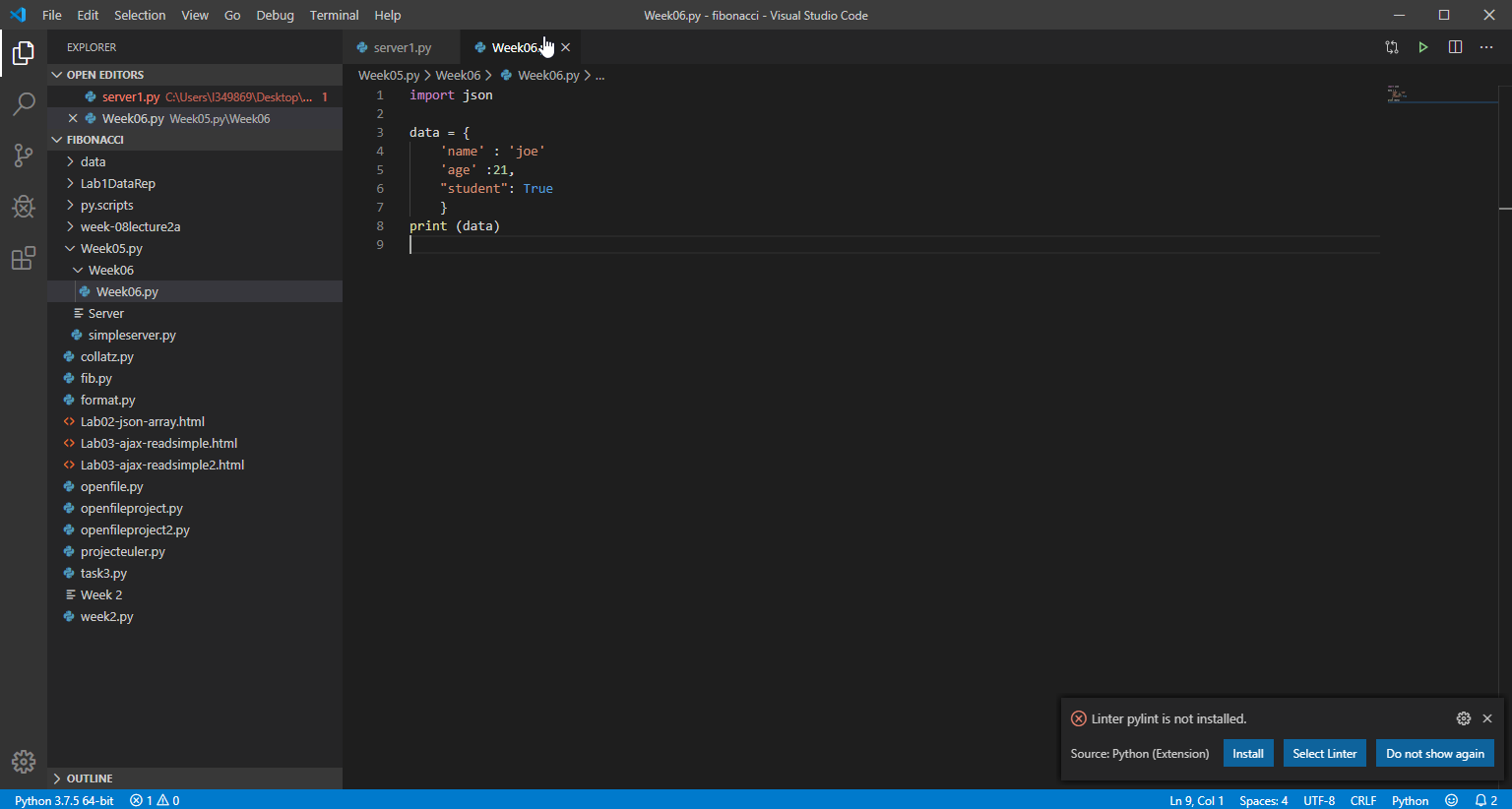
JSON package;

* In simple terms
* dump: converts Python object to JSON string and outputs to a file (dumps returns a string)
* load: read a JSON string from a file and converts to Python object (loads: read string directly

Mapping;

|  |  |
| --- | --- |
| JSON | Python |
| object | dict |
| array | list |
| string | str |
| number (int) | int |
| number (real) | float |
| true | True |
| false | False |
| null | None |

I follow Andrew’s steps in the [video](https://web.microsoftstream.com/video/0b26c2ed-9eac-48bb-90e9-62c13a0b8a6d?list=studio&referrer=https:%2F%2Flearnonline.gmit.ie%2Fcourse%2Fview.php%3Fid%3D1318) on Moodle;



Code:

import json

data = {

    'name' : 'joe'

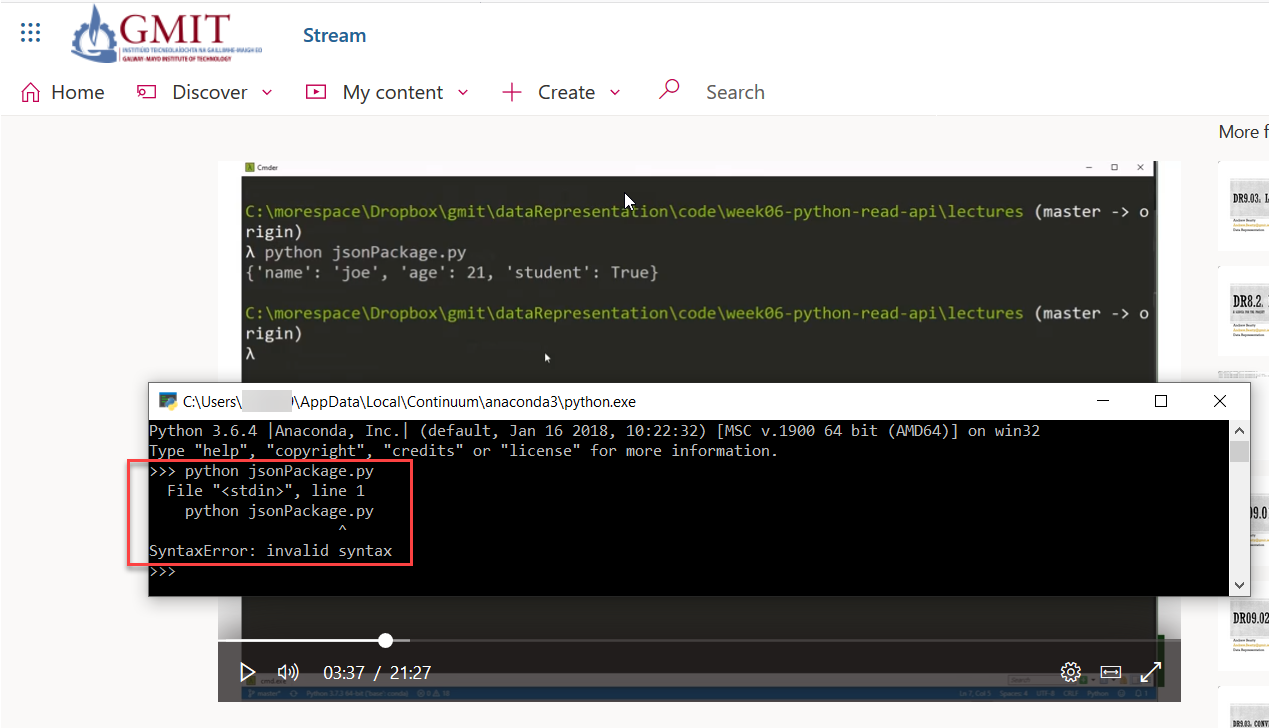
    'age' :21,

    "student": True

    }

print (data)

However, when I try to complete the below step I encounter an error;



I continued on with VS Code, although I couldn’t successfully run anything in commander;

import json

data = {

    'name' : 'joe'

    'age' :21,

    "student": True

    }

#print (data)

file = open("simple.json", 'w')

json.dump(data, file)

The below code should ensure the file appears neatly, and creates a nice indented file which will be easier to view;

import json

data = {

    'name' : 'joe'

    'age' :21,

    "student": True

    }

#print (data)

file = open("simple.json", 'w')

json.dump(data, file, indent=4)

To get a string and pass this up to a server;

import json

data = {

    'name' : 'joe'

    'age' :21,

    "student": True

    }

#print (data)

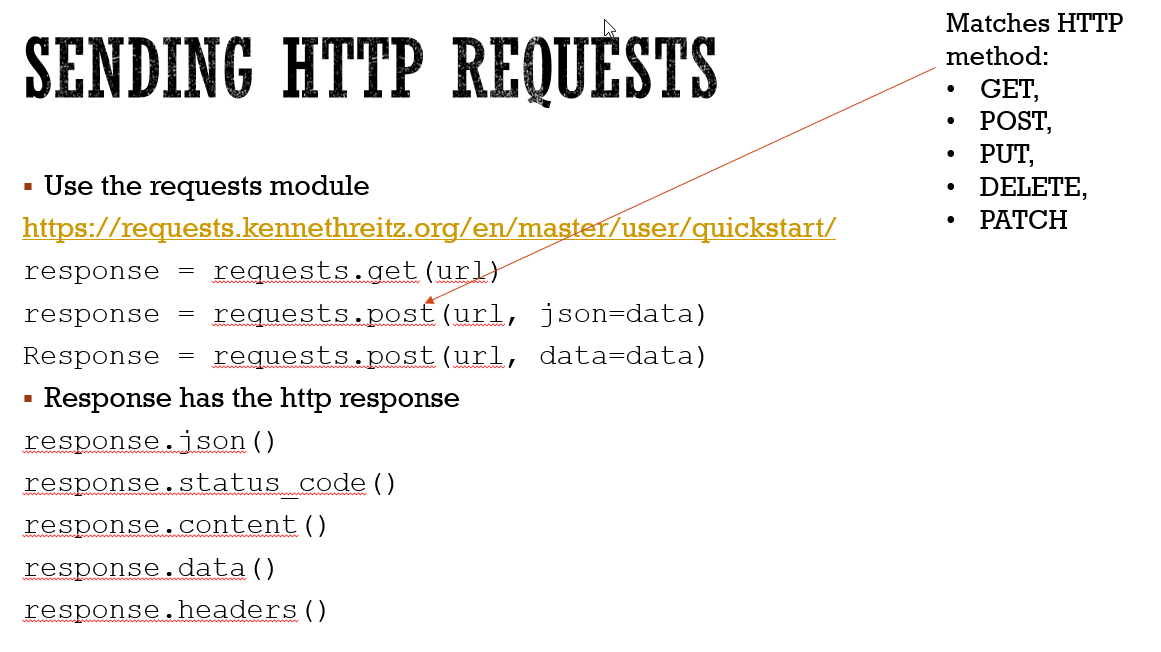
file = open("simple.json", 'w')

#json.dump(data, file, indent=4)

jsonString = json.dumps(data)

print(jsonString)

Sending HTTP requests;



Andrew demonstrated the code to print the status code, text and header from the GMIT website. I copied this;

import requests

url = 'https://www.gmit.ie'

response = requests.get(url)

print (response.status\_code)

print (response.text)

print (response.headers)

Code to do a post;

import requests

#url = 'https://www.gmit.ie'

#response = requests.get(url)

#print (response.status\_code)

#print (response.text)

#print (response.headers)

url = 'http://127.0.0.1:5000/cars'

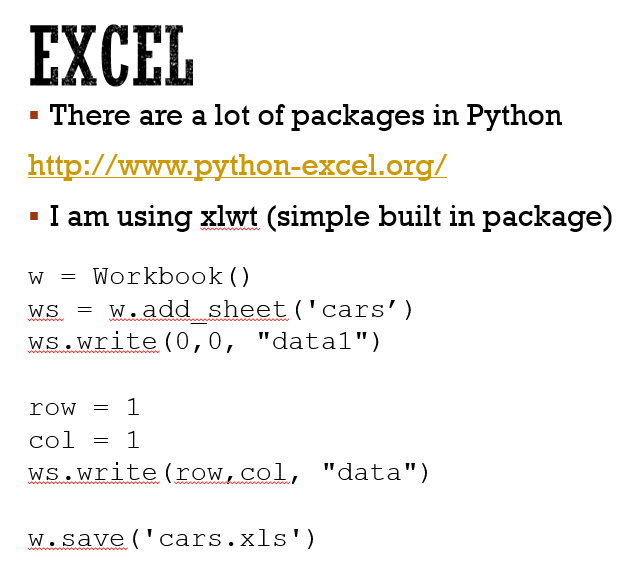
data = {'reg':'123','make':'blah','model':'blah','price':1234}

response = requests.post(url, json=data)

print(response.status\_code)

prit(response.json())





Etain: Stopped at 16.50 <https://web.microsoftstream.com/video/0b26c2ed-9eac-48bb-90e9-62c13a0b8a6d?list=studio&referrer=https:%2F%2Flearnonline.gmit.ie%2Fcourse%2Fview.php%3Fid%3D1318>

import requests

import json

from xlwt import \*

url = "http://217.0.0.1:5000/cars"

response = requests.get(url)

Video 2: API Keys & Oauth

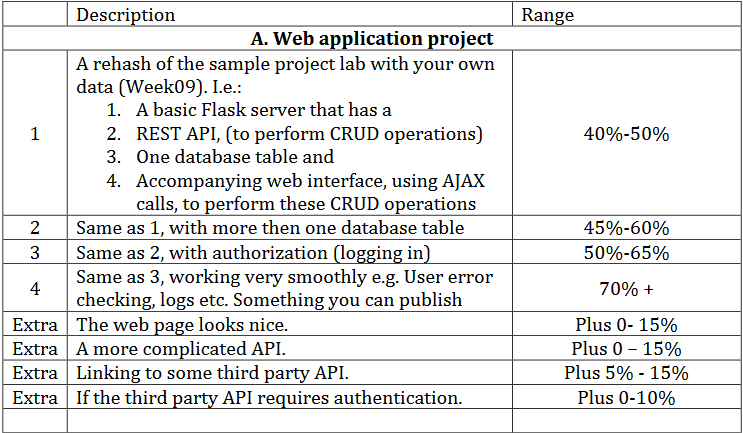
Python packages exist that make interacting with API’s easier.

**Etain’s final submission for the ‘big project’ as part of module 52957 - DATA-REPRESENTATION.**

**Lecturer: Andrew Beatty**

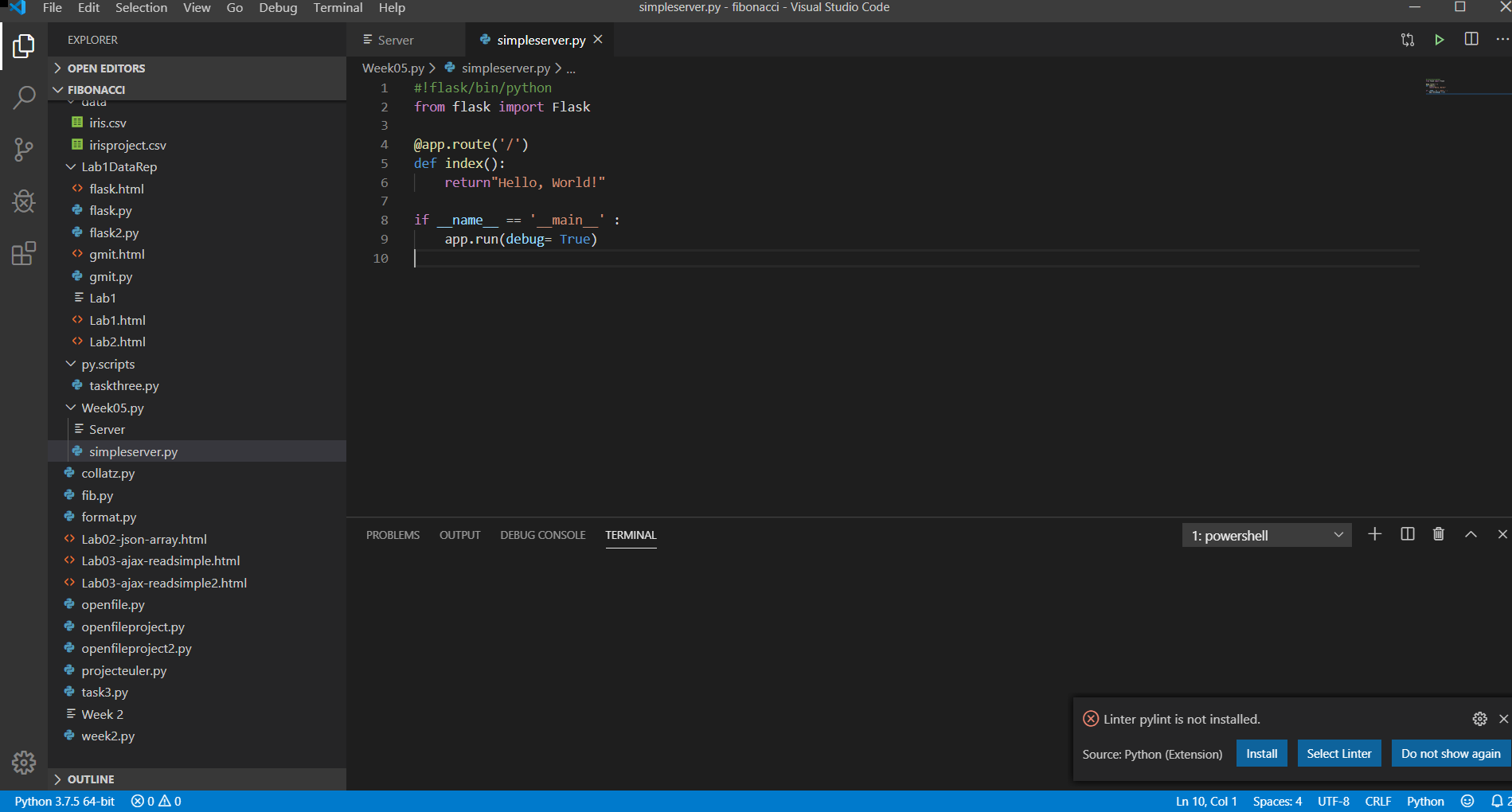
**Student: Etain Upton G00318876**

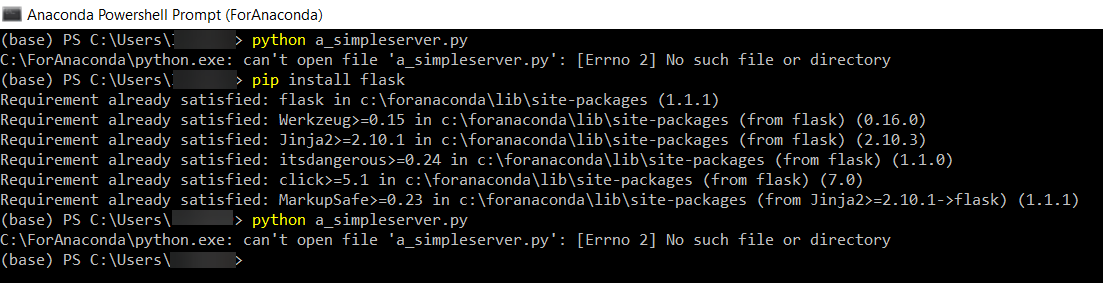
**I chose to complete Option (A) for this assessment – a Web Application based on;**



I found this module challenging. I successfully managed to complete Lab 01 but struggled with Lab 02 onwards.

I ran into issues following the *Lab 05.2 Flask* instructions;





As shown above, I continued to get the error ‘C:\ForAnaconda\python.exe: can't open file 'a\_simpleserver.py': [Errno 2] No such file or directory’.

I struggled to progress past this.

I have included my code below for review.

**An app server that has a RESTful interface to provide CRUD operations for one database table;**

I followed the steps completed by Andrew Beatty in his lecture *‘DR8.2a REST for project’* to write the following code;

from flask import Flask, jsonify, request, abort

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='.')

dogs=[

    { "id":123, "Name":"Fluffy", "Owner":"John Doe", "Value": 10000},

    { "id":456, "Name":"Bob", "Owner":"Jane Doe", "Value": 20000},

    { "id":789, "Name":"Rex", "Owner":"Jim Doe", "Value": 30000}

]

nextId=101

#app = Flask(\_\_name\_\_)

#@app.route('/')

#def index():

#    return "Hello, World!"

@app.route('/dogs')

def getAll():

    return jsonify(dogs)

@app.route('/dogs/<int:id>')

def findById(id):

    foundDogs = list(filter(lambda b: b['id'] == id, dogs))

    if len(foundDogs) == 0:

        return jsonify ({}) , 204

    return jsonify(foundDogs [0])

@app.route('/dogs', methods=['POST]')

def create():

    global nextId

    if not request.json:

        abort(400)

    #other checking

    dog = {

        "id": nextId,

        "Name": request.json['Name'],

        "Owner": request.json['Owner'],

        "Value": request.json['Value']

    }

    nextId += 1

    dogs.append(dog)

    return jsonify(dog)

@app.route('/dogs/<int:id>', methods=['PUT'])

def update(id):

    foundDogs = list(filter(lambda t: t['id']== id, dogs))

    if (len(foundDogs) == 0):

        abort(404)

    foundDog = foundDogs[0]

    if not request.json:

        abort(400)

    reqJson = request.json

    if 'Value' in reqJson and type(reqJson['Value']) is not int:

        abort(400)

    if 'Name' in reqJson:

        foundDog['Name'] = reqJson['Name']

    if 'Owner' in reqJson:

        foundDog['Owner'] = reqJson['Owner']

    if 'Value' in reqJson:

        foundDog['Value'] = reqJson['Value']

    return jsonify(foundDog)

    return "in update for id "+str(id)

@app.route('/dogs/<int:id>', methods=['DELETE'])

def delete(id):

    foundDogs = list(filter(lambda t: t['id']== id, dogs))

    if (len(foundDogs) == 0):

        abort(404)

    dogs.remove(foundDogs[0])

    return jsonify({"done":True})

if \_\_name\_\_ == '\_\_main\_\_' :

    app.run(debug= True)

**To make background colour 07.27 in** [**https://web.microsoftstream.com/video/4fa87e2f-32cb-449e-b728-a1a1ffc6cbbf?list=studio**](https://web.microsoftstream.com/video/4fa87e2f-32cb-449e-b728-a1a1ffc6cbbf?list=studio)

**Project**

**I have decided to complete (Option A) a Web Application project based on;**

There is flexibility in what you do for this project, but, as a minimum, your project should have code that demonstrates:

* Consuming a RESTful API, either in Python (your server) or javascript (web page) (week 5 lab 3)
* Running a Server.
* Web frontend to display data and interact with server (final lab in week 5 – note I need lad 2 to be completed for this). Lab ran through in video <https://web.microsoftstream.com/video/85608962-2580-4b7f-86ac-d62026f20a0e?list=studio>.

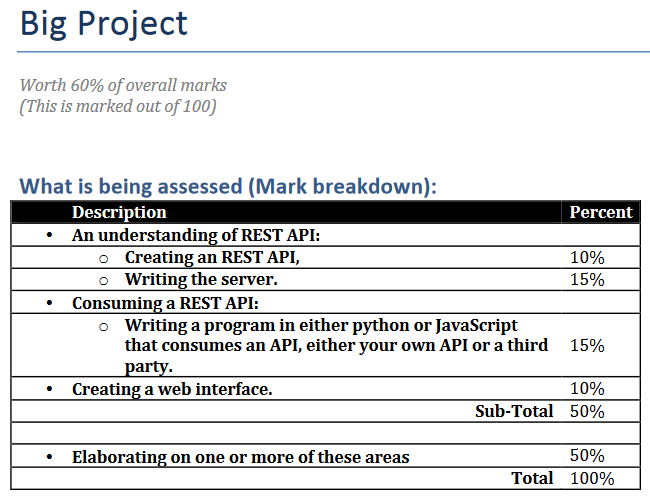
Andrew’s instruction;

Mandatory Handup (put link it the free text part):

* Link to GitHub that contains the server code (and requirements.txt)

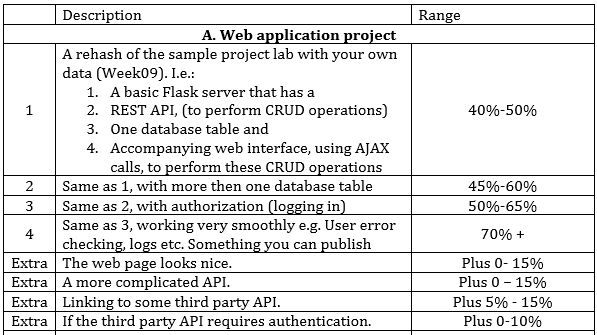
Optional (as files)

* Copy of .sql file that has a create command for your table(s)
* Link to a hosted version of your application
* Copy of an [overview](https://learnonline.gmit.ie/mod/url/view.php?id=58277) file that has any other information you think I need



Assessment strategy:

There are marks (50%)for each of the areas (basic)and optional extra marks(50%)for elaborating on any of the areas. You have flexibility as to what you do for this project. I understand that this can cause confusion as to what you should do, so here is a table of indicative grade ranges, for the kind of project you hand up.



*The project should be well laid out and easy for me to run. Marks may be deducted for poorly formatted code.*

*Handup:*

* 1. *A link to the GitHub repository directory that contains the project.*
  2. *A “ReadMe”file if there is anything (complicated) I need to do to run this code.*
  3. *You do not need to host the server on a cloud hosting site (azure) but if you do, please provide the link.*
  4. *Any other documentation you feel is appropriate.*

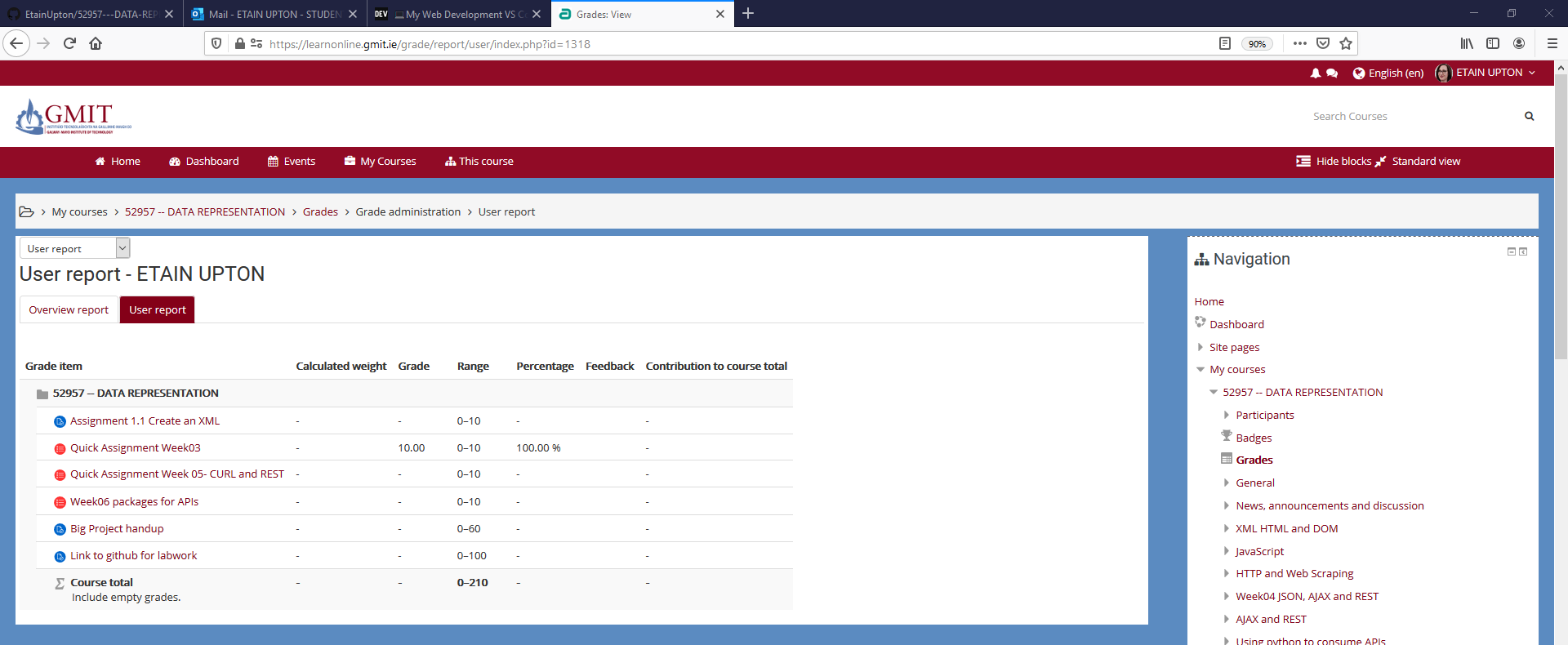
**Worth 60% of overall marks (This is marked out of 100).**

Note: Check the website I submitted as part of a previous module. (Web Application Development module).

I need a minimum of 70% in this project to pass as I’m unsure whether I’ve done the exercises…i.e. I need to complete as far as step 4.

**Note:**

**Don’t forget to upload my labs etc here;**



Lab walkthroughs; <https://web.microsoftstream.com/video/33832c4f-2cc0-4e9a-9bdf-9707c1b6710d?list=studio>