

Project 3: You can work alone or in pairs and this will be due at the start of the final on Tuesday, December 3rd.

Write your application-layer product, using the [REST API Architectural structure \(Links to an external site\)](#) and Python Flask. You will create a server that you can use POST command to add users. See below my example:

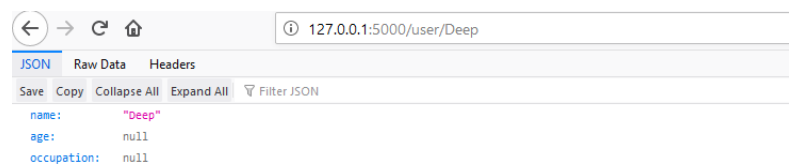
```
C:\WINDOWS\system32>curl --request GET --url http://127.0.0.1:5000/user/Deep
"User not found"

C:\WINDOWS\system32>curl --request POST --url http://127.0.0.1:5000/user/Deep
{
  "name": "Deep",
  "age": null,
  "occupation": null
}

C:\WINDOWS\system32>curl --request GET --url http://127.0.0.1:5000/user/Deep
{
  "name": "Deep",
  "age": null,
  "occupation": null
}

C:\WINDOWS\system32>
```

```
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 249-954-604
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [20/Nov/2019 11:02:03] "GET /user/Jass HTTP/1.1" 200 -
127.0.0.1 - - [20/Nov/2019 11:02:16] "POST /user/George HTTP/1.1" 201 -
127.0.0.1 - - [20/Nov/2019 11:04:02] "GET /user/George HTTP/1.1" 200 -
127.0.0.1 - - [20/Nov/2019 11:04:57] "GET /user/Deep HTTP/1.1" 404 -
127.0.0.1 - - [20/Nov/2019 11:11:45] "POST /user/Deep HTTP/1.1" 201 -
127.0.0.1 - - [20/Nov/2019 11:11:54] "GET /user/Deep HTTP/1.1" 200 -
```



Since we are not storing the data in actual database (if you prefer, you can do this on your project) once you close the program *post* data will get removed from the memory.

I also recommend that responses comes back from your server in JSON.

Testing the REST api servers is easiest if you use Chrome and the [Postman Chrome App \(Links to an external site\)](#) because it allows you to choose which of the commands (e.g., POST or GET) you are sending. In the example I'm using curl to post a command to create a user on my server. ***curl --request POST --url http://127.0.0.1:5000/user/Deep***

RESTful approaches to building client-server interactions are becoming the dominant approach to providing services. This involves commands similar to a web protocol (POST, GET, and DELETE). The GET command never causes a state change; POST or DELETE are used for operations that change state. (As you can read in the linked Wikipedia page, POST and DELETE should be idempotent, meaning you can call them multiple times and end up with the same result each time.) Clients connect to the service using http and the commands look like URLs.

To give you an example of a RESTful structure, you need to follow the instruction below to setup your virtual environment. I only test this in windows so you might have to work around for Linux and MAC. [MAC guide](#). Make sure to run your program as root in a terminal. Update your Python to the latest version.

Steps to get your system ready:

Install Pip:

As of Python Pip is installed automatically and will be available in your Scripts folder.

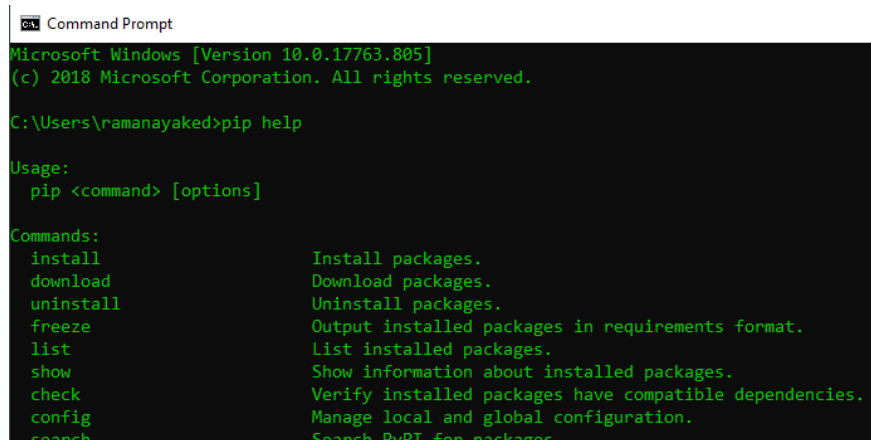
If you install a later version of Python I would recommend installing it according to this helpful [stackoverflow post](#).

Pip is a Package manager for python which we will use to load in modules/libraries into our environments.

An example of one of these libraries is VirtualEnv which will help us keep our environments clean from other Libraries. This sounds really confusing but as you start using it you'll begin to understand how valuable this encapsulation of modules/libraries can be.

To test that Pip is installed open a command prompt (win+r->'cmd'->Enter) and try 'pip help'

You should see a list of available commands including install, which we'll use for the next part:



```
Command Prompt
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

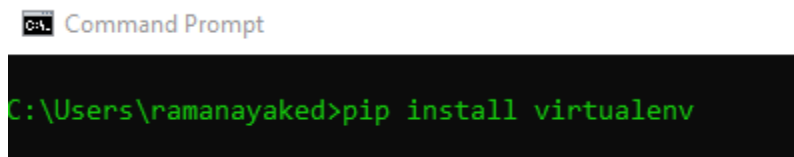
C:\Users\ramanayaked>pip help

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show            Show information about installed packages.
  check           Verify installed packages have compatible dependencies.
  config          Manage local and global configuration.
  search          Search PyPI for packages.
```

Install virtualenv:

Now that you have pip installed and a command prompt open installing virtualenv to our root Python installation is as easy as typing 'pip install virtualenv'
Like so:



```
Command Prompt

C:\Users\ramanayaked>pip install virtualenv
```

Now we have virtualenv installed which will make it possible to create individual environments to test our code in. But managing all these environments can become cumbersome. So we'll pip install another helpful package...

Install virtualenvwrapper-win:

This is the kit and caboodle of this guide.

Just as before we'll use pip to install virtualenvwrapper-win. 'pip install virtualenvwrapper-win' Like so:

```
C:\Users\ramanayaked>pip install virtualenvwrapper-win
```

Excellent! Now we have everything we need to start building software using python! Now I'll show you how smooth it is to use these awesome tools!

Make a Virtual Environment:

Let's call it HelloWoldDeep (you can call whatever you want here). All we do in a command prompt is enter 'mkvirtualenv HelloWoldDeep'

This will create a folder with python.exe, pip, and setuptools all ready to go in its own little environment. It will also activate the **Virtual** Environment which is indicated with the (HelloWoldDeep) on the left side of the prompt.

```
C:\Users\ramanayaked>mkvirtualenv HelloWoldDeep
Using base prefix 'c:\users\ramanayaked\appdata\local\programs\python\python37'
New python executable in C:\Users\ramanayaked\Envs\HelloWoldDeep\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
(HelloWoldDeep) C:\Users\ramanayaked>
```

Anything we install now will be specific to this project. Nowhere else and available to the projects we connect to this environment.

Connect our project with our Environment:

Now we want our code to use this environment to install packages and run/test code.

First let's create a directory with the same name as our **virtual environment** in our preferred development folder. In this case mine is 'dev'

```
(HelloWoldDeep) C:\Users\ramanayaked>mkdir dev
```

```
(HelloWoldDeep) C:\Users\ramanayaked\dev>cd dev
```

Dev will be the root folder of our first project!

Set Project Directory:

Now to bind our virtualenv with our current working directory we simply enter 'setprojectdir .' Like so:

```
(HelloWoldDeep) C:\Users\ramanayaked\dev>setprojectdir .  
  
"C:\Users\ramanayaked\dev" is now the project directory for  
virtualenv "C:\Users\ramanayaked\Envs\HelloWoldDeep"  
  
"C:\Users\ramanayaked\dev" added to  
C:\Users\ramanayaked\Envs\HelloWoldDeep\Lib\site-packages\virtualenv_path_extensions.pth  
(HelloWoldDeep) C:\Users\ramanayaked\dev>
```

Now next time we activate this environment we will automatically move into this directory!

Deactivate:

Let say you're content with the work you've contributed to this project and you want to move onto something else in the command line. Simply type 'deactivate' to deactivate your environment.

```
(HelloWoldDeep) C:\Users\ramanayaked\dev>deactivate  
  
C:\Users\ramanayaked\dev>
```

Notice how the parenthesis disappear.

You don't have to deactivate your environment. Closing your command prompt will deactivate it for you. As long as the parenthesis are not there you will not be affecting your environment. But you will be able to impact your root python installation.

Workon:

Now you've got some work to do. Open up the command prompt and type 'workon HelloWoldDeep' to activate the environment and move into your root project folder.

```
(HelloWoldDeep) C:\Users\ramanayaked\dev>deactivate  
  
C:\Users\ramanayaked\dev>workon HelloWoldDeep  
(HelloWoldDeep) C:\Users\ramanayaked\dev>
```

Pip Install:

To use flask we need to install the packages and to do that we can use pip to install it into our HelloWoldDeep virtual environment.

Make sure (HelloWoldDeep) is to the left of your prompt and enter 'pip install flask'

```
(HelloWoldDeep) C:\Users\ramanayaked\dev>pip install flask
Collecting flask
  Using cached https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9641f4caf13540e2cdec85276964ff8f43b
ny.whl
Collecting Jinja2>=2.10.1
  Using cached https://files.pythonhosted.org/packages/65/e0/eb35e762802015cab1ccee04e8a277b03f1d8e53da3ec3106882ec
-any.whl
Collecting click>=5.1
  Using cached https://files.pythonhosted.org/packages/fa/37/45185cb5abb30d7257104c434fe0b07e5a195a6847506c074527a
.whl
Collecting Werkzeug>=0.15
  Using cached https://files.pythonhosted.org/packages/ce/42/3aeda98f96e85fd26180534d36570e4d18108d62ae36f87694b476
ne-any.whl
Collecting itsdangerous>=0.24
  Using cached https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b8
-none-any.whl
Collecting MarkupSafe>=0.23
  Using cached https://files.pythonhosted.org/packages/65/c6/2399700d236d1dd681af8aebff1725558cddf6e43d7a5184a675f
m-win_amd64.whl
Installing collected packages: MarkupSafe, Jinja2, click, Werkzeug, itsdangerous, flask
Successfully installed Jinja2-2.10.3 MarkupSafe-1.1.1 Werkzeug-0.16.0 click-7.0 flask-1.1.1 itsdangerous-1.1.0
```

You also need pip install flask-restful. Make sure to inside you virtual environment.

```
(HelloWoldDeep) C:\Users\ramanayaked\dev>pip install flask-restful
Collecting flask-restful
  Using cached https://files.pythonhosted.org/packages/17/44/6e490150ee443ca81d5f88b61bb4bbb133
3-none-any.whl
Collecting aniso8601>=0.82
  Using cached https://files.pythonhosted.org/packages/eb/e4/787e104b58eadc1a710738d4e418d7e599
ne-any.whl
Collecting six>=1.3.0
  Using cached https://files.pythonhosted.org/packages/65/26/32b8464df2a97e6dd1b656ed26b2c19460
y.whl
Collecting pytz
  Using cached https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d2
ny.whl
Requirement already satisfied: Flask>=0.8 in c:\users\ramanayaked\envs\helloworlddeep\lib\site-p
Requirement already satisfied: Werkzeug>=0.15 in c:\users\ramanayaked\envs\helloworlddeep\lib\si
Requirement already satisfied: Jinja2>=2.10.1 in c:\users\ramanayaked\envs\helloworlddeep\lib\si
Requirement already satisfied: click>=5.1 in c:\users\ramanayaked\envs\helloworlddeep\lib\site-p
Requirement already satisfied: itsdangerous>=0.24 in c:\users\ramanayaked\envs\helloworlddeep\li
.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\ramanayaked\envs\helloworlddeep\lib\
-restful) (1.1.1)
Installing collected packages: aniso8601, six, pytz, flask-restful
Successfully installed aniso8601-8.0.0 flask-restful-0.3.7 pytz-2019.3 six-1.13.0
(HelloWoldDeep) C:\Users\ramanayaked\dev>
```

This will bring in all the tools required to write your web server application! Now that you have flask installed in your virtual environment you can start coding! Open up your favorite text editor and create a new file called hello.py and save it in your HelloWoldDeep directory. I've simply taken the sample code from Flask's website to create a very basic 'Hello World!' server.

I've named the file hello.py.

```

hello.py - C:/Users/ramanayaked/dev/hello.py (3.7.3)
File Edit Format Run Options Window Help
#!/flask/bin/python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return "Hello, World!"

if __name__ == '__main__':
    app.run(debug=True)

```

Make sure to save it the **Dev** Directory we created:

Once the code is in place I can start the server using 'python hello.py' this will run the python instance from your virtual environment that has flask.

You can now navigate with your browser to <http://127.0.0.1:5000/> and see your new site!

```

Command Prompt - python hello.py

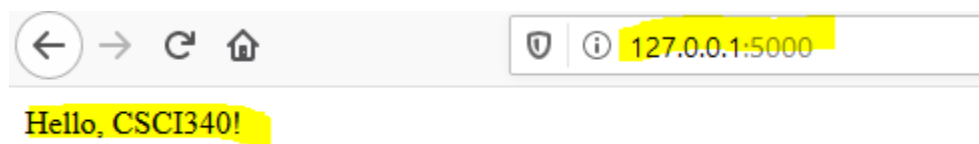
(HelloWoldDeep) C:\Users\ramanayaked\dev>dir
Volume in drive C is OSDisk
Volume Serial Number is 4C33-EA7F

Directory of C:\Users\ramanayaked\dev

11/20/2019  11:40 AM  <DIR>          .
11/20/2019  11:40 AM  <DIR>          ..
11/20/2019  11:40 AM                186 hello.py
               1 File(s)                186 bytes
               2 Dir(s)  325,876,883,456 bytes free

(HelloWoldDeep) C:\Users\ramanayaked\dev>python hello.py
* Serving Flask app "hello" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 171-957-534
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```



You have everything you need to start working on your project on Flask without worrying about gunking up your Python installations or your OS because we run everything in virtual environment. Make sure to work on the same folder and your terminal window is (HelloWoldDeep). Noticed the missing r 😊. Credit goes on creating these steps to:

I would read this article here first before you writing your code get an idea of how send requests to a server. <https://blog.miguelgrinberg.com/post/designing-a-restful-api-with->

[python-and-flask](#) . You can use this as a guidance to create your application to add users to database. <https://kite.com/blog/python/flask-restful-api-tutorial/> or easily modify the code here create users instead of what they have in here:
<https://www.bogotobogo.com/python/python-REST-API-Http-Requests-for-Humans-with-Flask.php>

Another good example: <https://www.thegeekstuff.com/2019/04/rest-basic-concepts/>

Or write your own application.