Flask Framework with Python

Setting up the Environment

Objectives:

It's good practice to have a different python environment for different development setup. Here we're learning about how to setup an virtual environment for Flask Framework for our web application development.

By the end of this section, you should be able to:

- Install `virtualenv` , `virtualwrapper` and create virtual environments
- Use Flask to setup a simple server and respond with text

Getting set up in a virtual environment

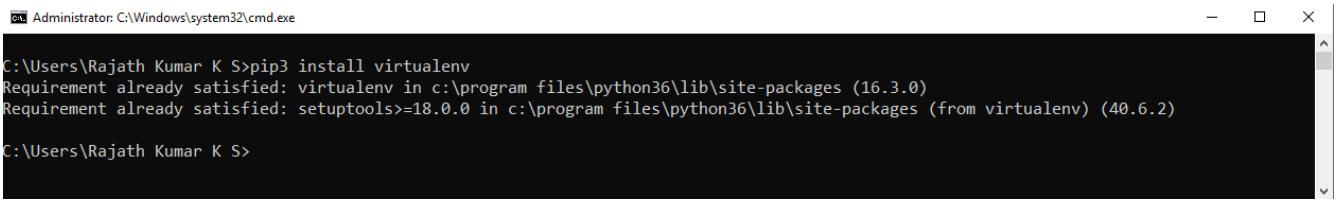
As you start working on more Python projects, you may find that managing dependencies becomes problematic. For example, you may start a project that uses version 3.5.x of some library, and later on you may start a new project and realize that version 3.7.x of the module is updated. But this new version breaks things in your old project, so you need to either deal with a broken old project, or a new project with an old version of the library.

It would be better if we could separate out distinct environments for each Python project we worked on. That way, we wouldn't have to worry about dependencies in one project potentially interfering with other projects.

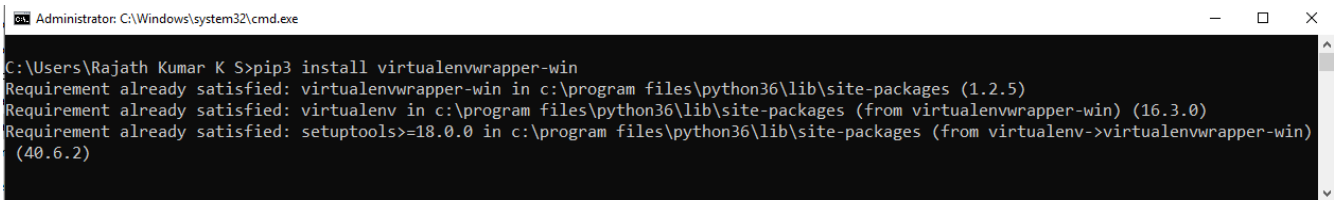
Thankfully, there's a way to set this up without much trouble in Python. The tool we'll be using is called `virtualenv` and `virtualwrapper` . Before we get started with any significant Python development, let's set up `virtualenv` and `virtualwrapper` so we can create separate environments for each project we'll be working on.

Installing `virtualenv` and `virtualwrapper`

```
pip3 install virtualenv
```




```
pip3 install virtualenvwrapper-win
```

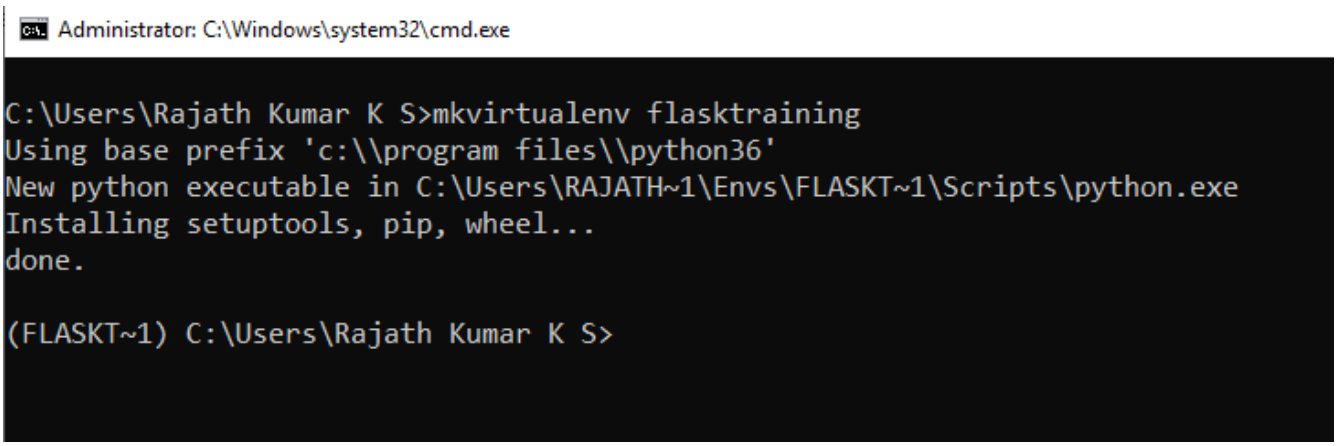


Creating Virtual Environment

To create virtual environment, you'll use `mkvirtualenv` command

```
mkvirtualenv flasktraining
```

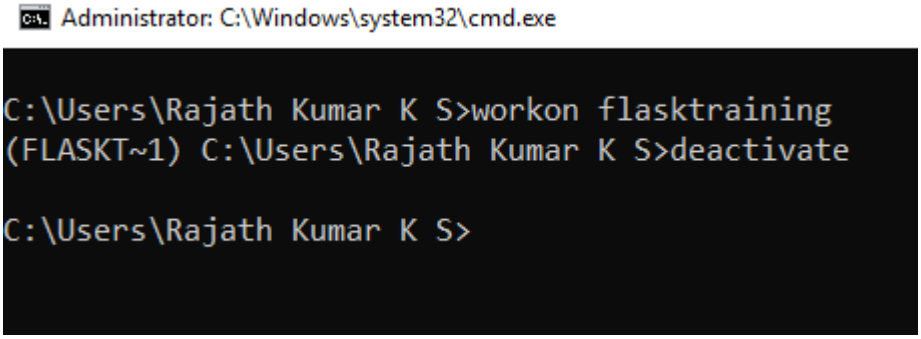
Here in this case i'm using my virtual environment name as **flasktraining**, you can use your own virtual environment name.



When it's created , it gets into virtual environment. To get out of your virtual environment use `deactivate`

To work in virtual environment you need to type in as follows,

```
workon flasktraining
```

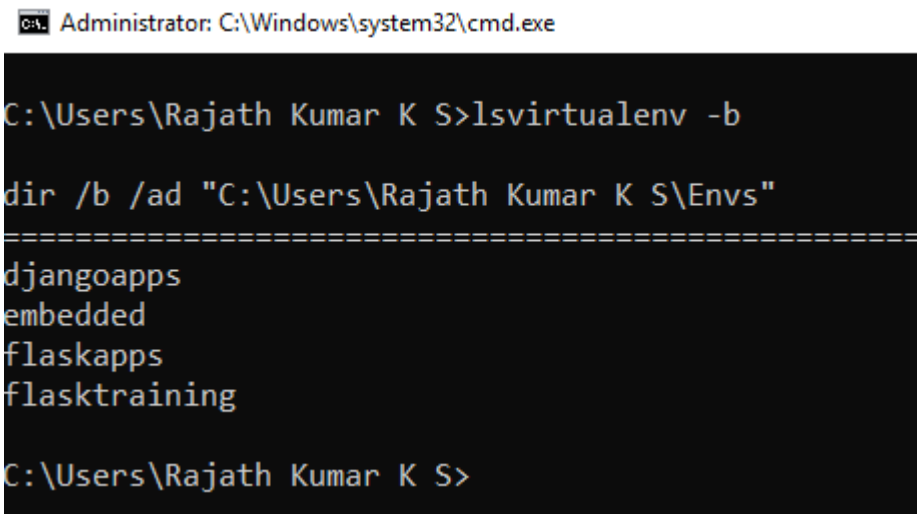


```
deactivate
```

to exit the virtual environment to your base machine.

To list the virtualenvironments that we've created in our system, you can use the `lsvirtualenv -b` command (`-b` stands for "brief").

```
lsvirtualenv -b
```



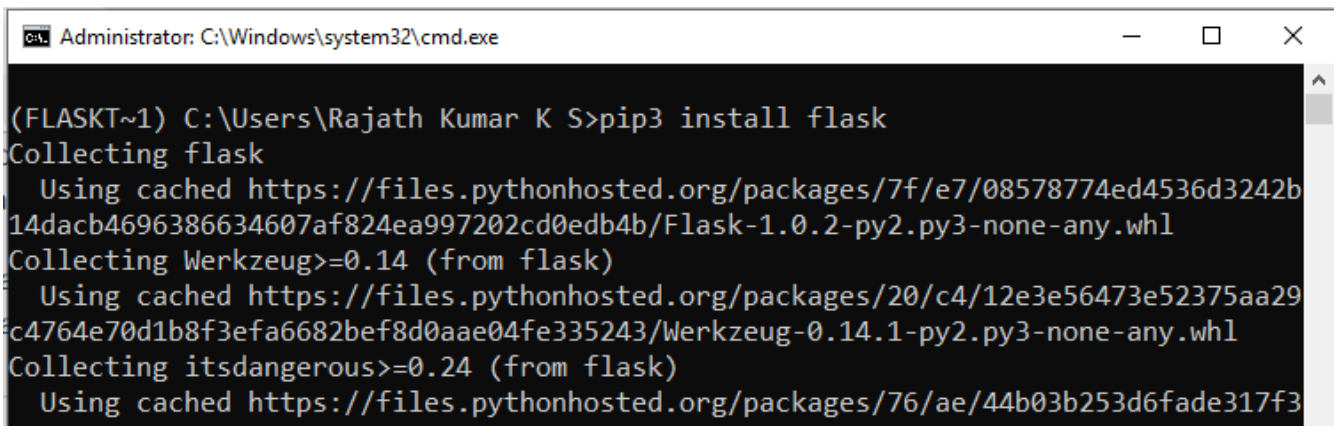
In my case i've installed with four virtual environments viz.,

1. djangoapps
2. embedded
3. flaskapps
4. flasktraining -- Virtual Environment that created now.

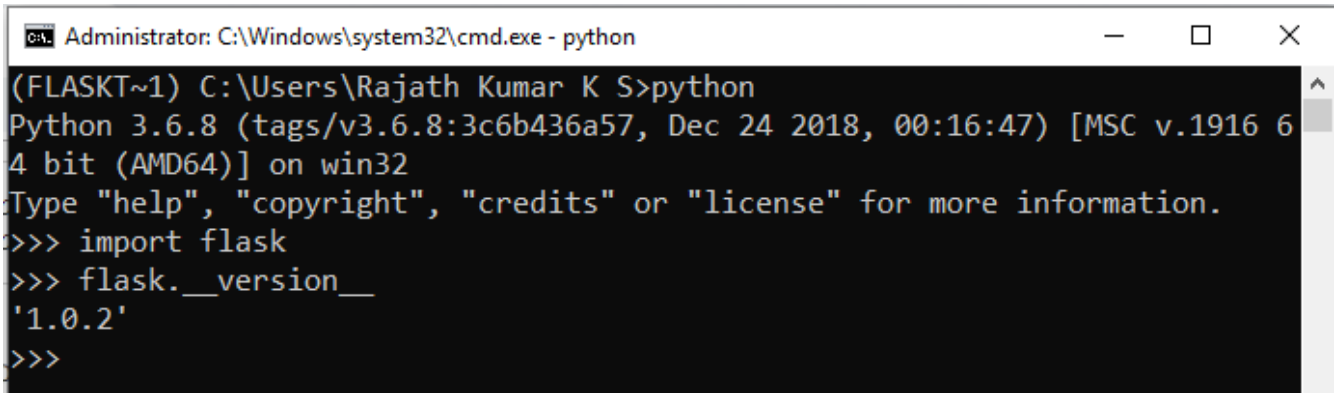
Let's install flask python package for the virtual environment that we've created . To install flask package i'm using PyPi (Python Package Index)

```
pip3 install flask
```

make sure that you're working inside the virtual environment you've created.



Let's check our installation happend we made has done properly, to check go to python interactive shell by typing `python` in your command prompt. Then type in `import flask`, if you don't see any error then you're installed flask on your virtual environment



Introduction to Flask Framework

> **Pirates use Flask, the Navy uses Django**

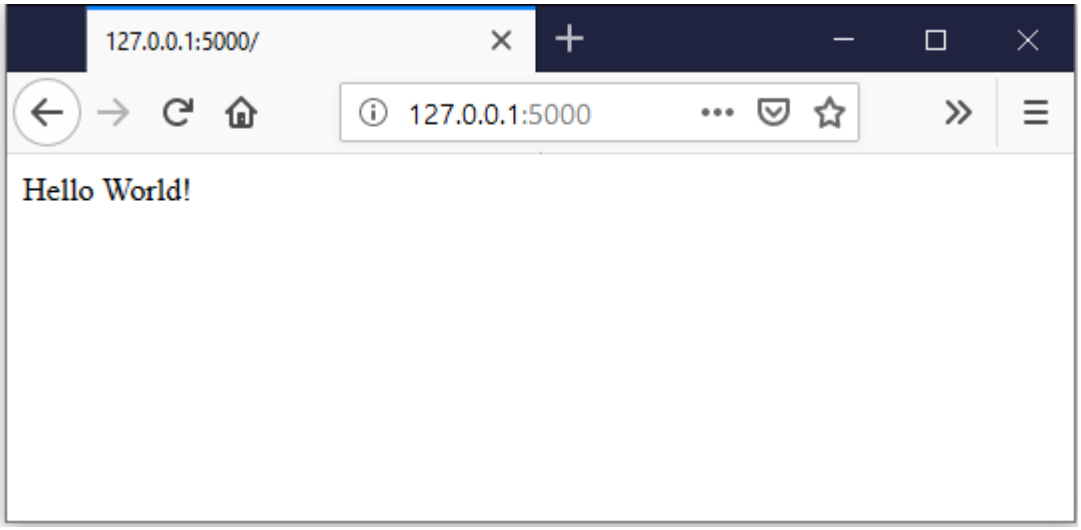
What is Flask ?

Flask is a micro-framework in Python. It allows us to easily start a server, and, when combined with other modules, build sophisticated applications. Flask is very easy to get started with so let's jump in!

Now let's create an main.py file in the desired location. To begin, we'll just use the below code.

```
# from the flask library import a class named Flask
from flask import Flask

# create an instance of the Flask class
app = Flask(__name__)

# listen for a route to `/` - this is known as the root route
@app.route('/')
def home():
    return "Hello World!"

if __name__ == '__main__':
    app.run()
```

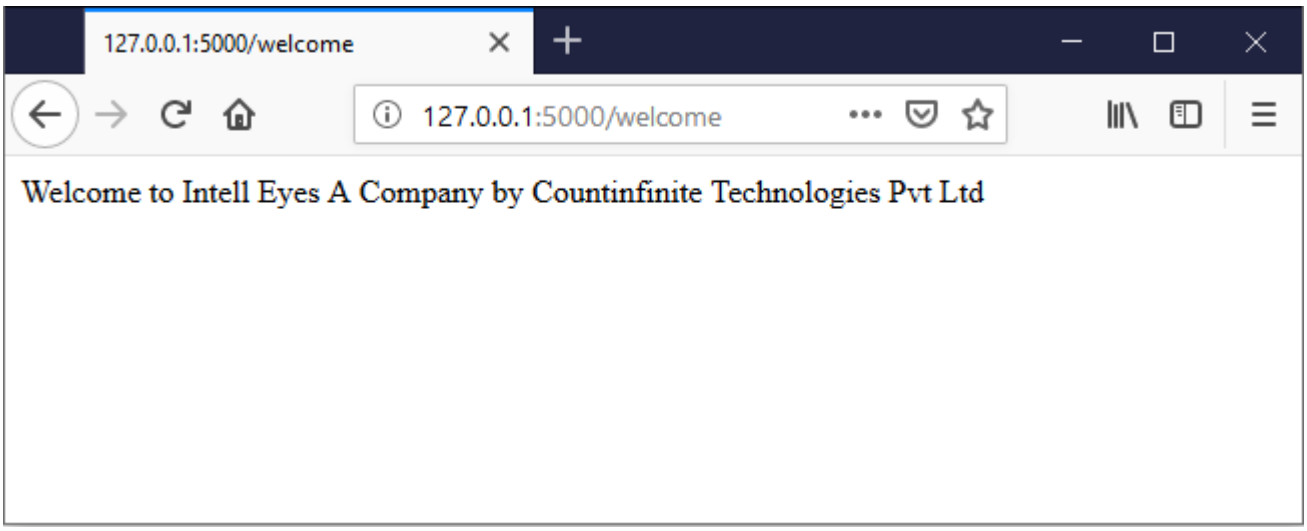
Make sure we saved our file. Next, in the terminal let's run `python main.py` and head over to `localhost:5000` / `127.0.0.1:5000` in the browser. You should see the text "Hello World" appear on the page!.



Now let's try to add another route to our application. When a user goes to `localhost:5000\welcome` we should return the text "Welcome to Intell Eyes A Company by Countinfinite Technologies Pvt Ltd." . We will need to include another route as well as a function to run when that route is reached.

```
# from the flask library import a class named Flask
from flask import Flask

# create an instance of the Flask class
app = Flask(__name__)

# listen for a route to `/` - this is known as the root route
@app.route('/')
def home():
    return "Hello World!"

@app.route('/welcome')
def welcome():
    return "Welcome to Intell Eyes A Company by Countinfinite Technologies Pvt Ltd"

if __name__ == '__main__':
    app.run()
```

Check back your link that you created has a new route by hitting `127.0.0.1:5000\welcome` on your favourite browser.



Flask Exercises

For thsi assignemtn you will be creating a very small flask appkication,. Your application should:

- have a route for `/welcome` , which responds with the string "Welcome to Flask from Intell Eyes".
- have a route for `/welcome/home`, which responds with the string "Welcome Home"

Bonus Exercise

- have a route for `/welcome/<yourname>`, which responds with string "Welcome `<yourname>`"