Welcome to lecture 2: Learning flask from CS50 web programming with python and javascript

By Amirhossein Safari
Telegram: @AmirhosseinSFR

Following times are times in the video that the master shows his code or run it:

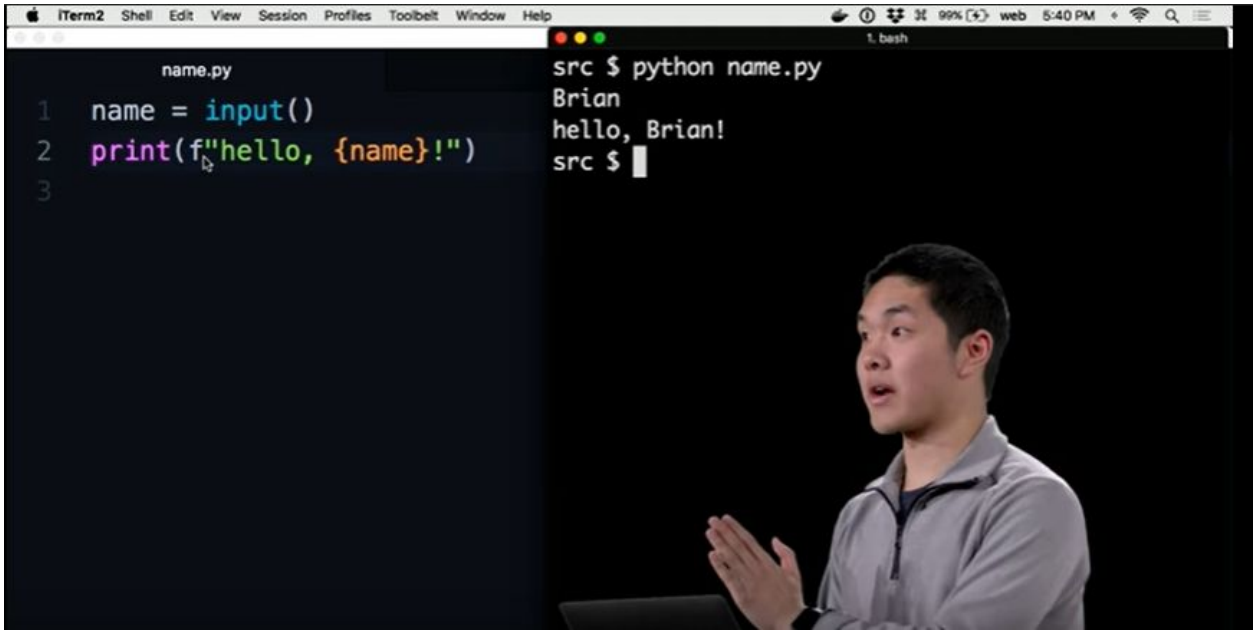
2:15
2:35
5:44 format string == f
9:19
12:09
12:20
12:26
14:50
15:34
16:05
17:36
17:59
19:32
20:02
21:16
23:51 showing data but not repeated data in a list
24:40
26:06 Mapping keys to values(data to another one)
27:19
29:35
30:27 and we should define functions first in Python
31:22
32:44
33:23
34:38 fixed the bug in 33:23
34:59
36:12
38:29
38:22
42:42
45:30
49:55
50:59
52:56
1:13:23
1:14:49
1:15:12
1:16:28
1:16:34
1:18:32
1:19:50
1:20:58
1:23:36
1:23:42
1:30:29
1:35:26
1:36:07
1:40:56
1:44:47

54:04
54:57
57:39
58:51
1:05:15
1:06:15
1:06:27
1:12:14
1:12:29

Ok, and some summary:)

This is some basics of python language:

1.Getting inputs and set it in a format string by{}



2.How to work with arrays and list items and error of indexing out of range(IndexError)

Sequences

- Strings:

```
name = "Alice"
print(name[0])
```

  - Strings are justs sequence of characters, and can be indexed as such.
- Tuples:

```
coordinates = (10.0, 20.0)
print(coordinates[1])
```

  - Tuples are immutable collections of values under a single name, which can be indexed positionally.
- Lists:

```
names = ["Alice", "Bob", "Charlie"]
print(names[2])
```

  - Lists are mutable collections of values under a single name, which can be indexed positionally.
  - Indexing out of range raises a Python 'exception'. In this case, an `IndexError`, because there is no fourth value in `names` for Python to return.

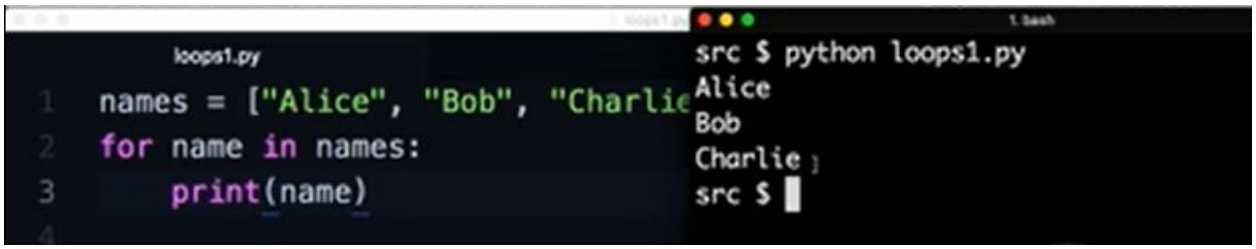
```
>>> names = ["Alice", "Bob", "Charlie"]
>>> names[0]
'Alice'
>>> names[1]
'Bob'
>>> names[2]
'Charlie'
>>> names[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>>
```

3.python interpreter

You can run codes in python interpreter by enter “python” command in terminal and write the code

```
>>> x = 28
>>> x
28
>>> if x > 0:
...     print("x is positive")
...
x is positive
>>>
```

4.For loops in Arrays



5.set()

The way we can remove repeated items

```
src $ python sets.py
{1, 3, 5}
src $
```

6.Mapping keys to values

dictionaries.py

```python
ages = {"Alice": 22, "Bob": 27}
ages["Charlie"] = 30
ages["Alice"] += 1

print(ages)
```

```
src $ python dictionaries.py
{'Alice': 23, 'Bob': 27, 'Charlie': 30}
src $
```

7.functions

functions.py

```python
def square(x):
    return x * x

for i in range(10):
    print("{} squared is {}".format(i, square(i)))
```

```
src $ python functions.py
0 squared is 0
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
6 squared is 36
7 squared is 49
8 squared is 64
9 squared is 81
src $
```

Note:functions should define before where we call them in python and if we don't...

```
src $ python functions.py
Traceback (most recent call last):
  File "functions.py", line 2, in <module>
    print("{} squared is {}".format(i, square(i)))
NameError: name 'square' is not defined
src $
```

8.Importing functions from other file

modules.py functions.py

```
from functions import square

print(square(10))
```

```
src $ python modules.py
0 squared is 0
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
6 squared is 36
7 squared is 49
8 squared is 64
9 squared is 81
100
src $
```

Wait…

There is a bug here, we just expect 100 here, what are the other numbers?

Well, we should put the for loop in a function to don't let it execute when we import the file in other file

So we have main() function!

```
modules.py        functions.py
def square(x):
    return x * x

def main():
    for i in range(10):
        print("{} squared is {}".format(i, square(i)))

if __name__ == "__main__":
    main()
```

Note: If statement just call the main() function, if we execute this file(functions.py here)

```
src $ python modules.py
100
src $
```

So bug fixed:))

9.Classes

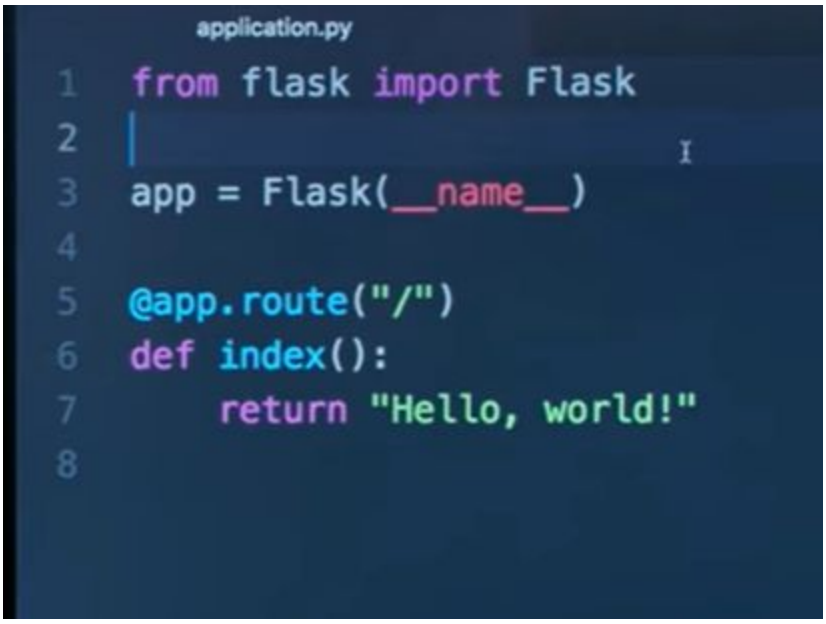
classes.py

```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

p = Point(3, 5)
print(p.x)
print(p.y)
```

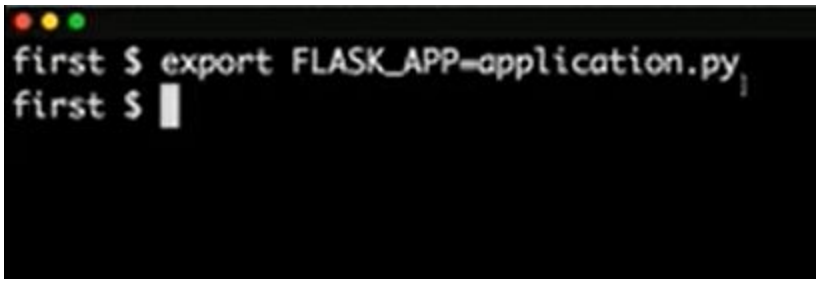
Note: __init__ function execute when we make an instance of class in this way

And here you go… Flask!

1.simple hello world



2.How to run flask app



And then “flask run”

3.Dynamic hello!!
Using url parameters

application.py

```
from flask import Flask

app = Flask(__name__)

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

@app.route("/<string:name>")
def hello(name):
    return f"Hello, {name}!"
```

4.Flask can return html tags

application.py

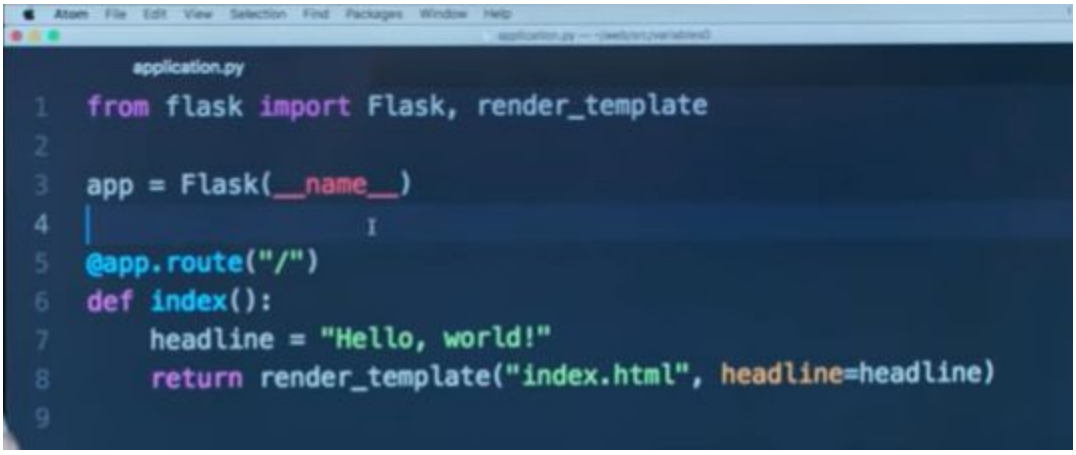
```python
from flask import Flask

app = Flask(__name__)

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

@app.route("/<string:name>")
def hello(name):
    name = name.capitalize()
    return f"<h1>Hello, {name}!</h1>"
```

5.Rendering html files

application.py

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")
```

6.Sending data in html file from flask

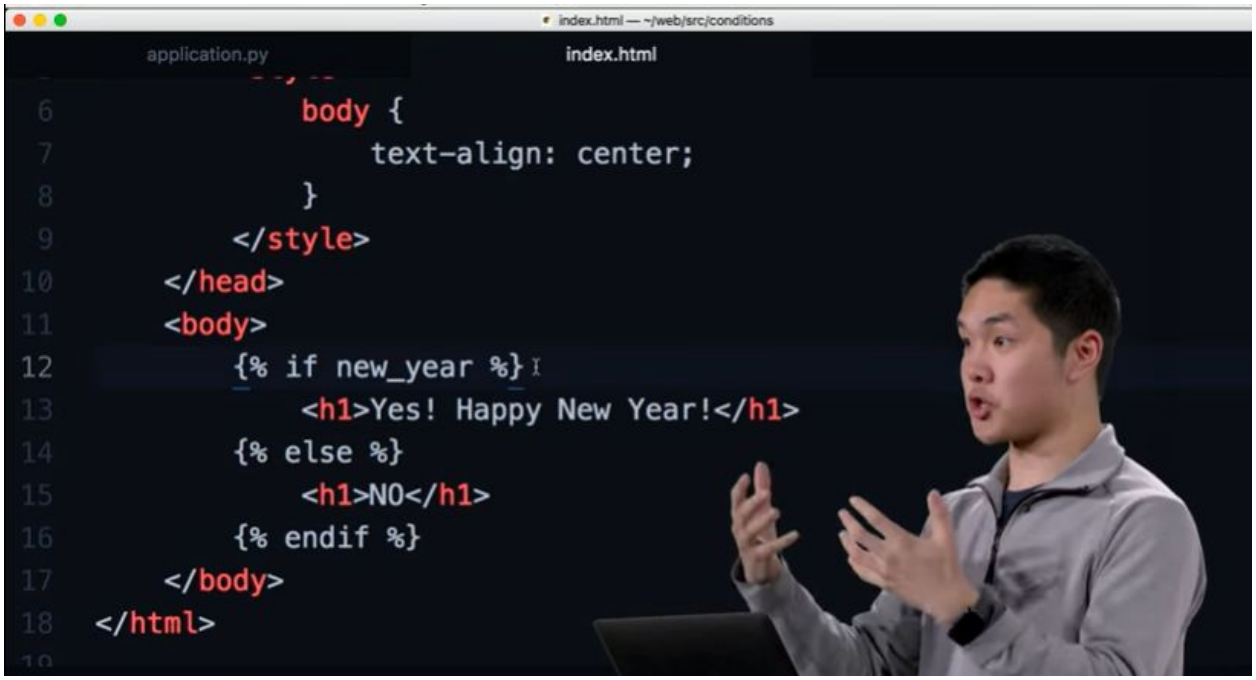


```html
<!DOCTYPE html>
<html>
    <head>
        <title>My Website!</title>
    </head>
    <body>
        <h1>{{ headline }}</h1>
    </body>
</html>
```

Dynamic variable inside {{}}

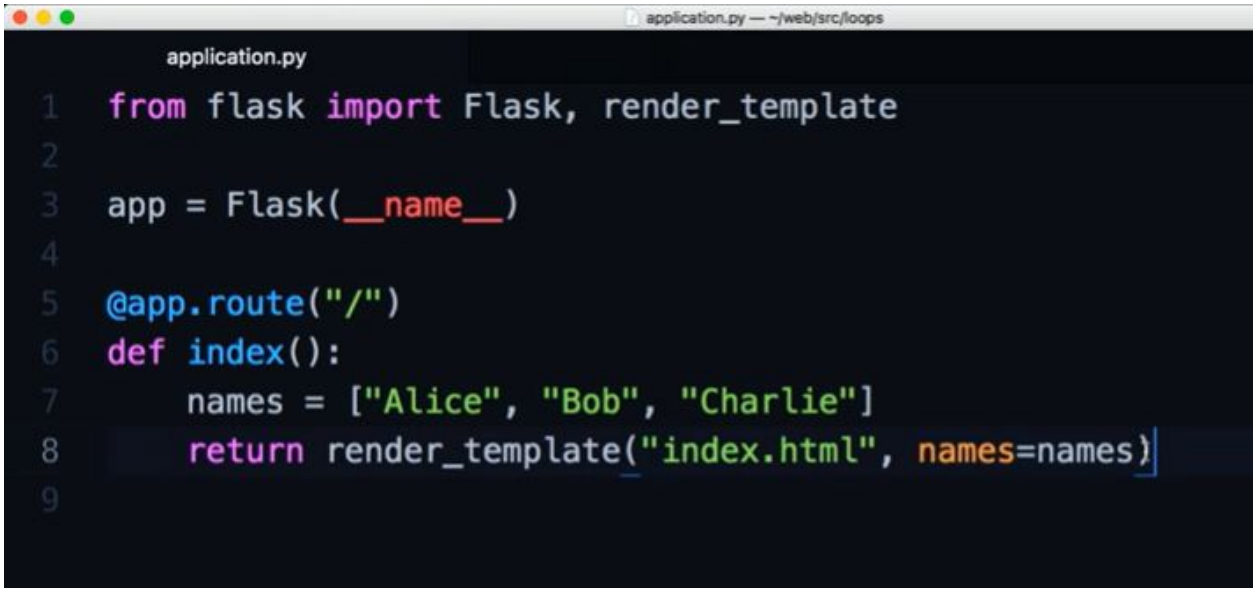
7.Using date parameters in flask

```python
import datetime

from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    now = datetime.datetime.now()
    new_year = now.month == 1 and now.day == 1
    return render_template("index.html", new_year=new_year)
```



Note: new_year is a Boolean

8.passing an array to html file



application.py index.html

```
3       <head>
4           <title>My Website!</title>
5       </head>
6       <body>
7           <h1>Names</h1>
8
9           <ul>
10              {% for name in names %}
11                  <li>{{ name }}</li>
12              {% endfor %}
13          </ul>
14      </body>
15  </html>
```

My Website!

127.0.0.1:5000

Names

- Alice
- Bob
- Charlie

9.url_for()

Well, this function get the name of function in flask file and route to the **url of that function**

And when page navigate to that url, the function automatically would be called

application.py — ~/web/src/urls

application.py

```
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4
5  @app.route("/")
6  def index():
7      return render_template("index.html")
8
9  @app.route("/more")
10 def more():
11     return render_template("more.html")
12
```

index.html — ~/web/src/urls

application.py | index.html

```
3     <head>
4         <title>My Website!</title>
5     </head>
6     <body>
7         <h1>First Page</h1>
8
9         <p>
10            Lorem ipsum dolor sit amet, consectetur adipiscing elit.
11        </p>
12
13        <a href="{{ url_for('more') }}">See more...</a>
14    </body>
15 </html>
```

more.html — ~/web/src/urls

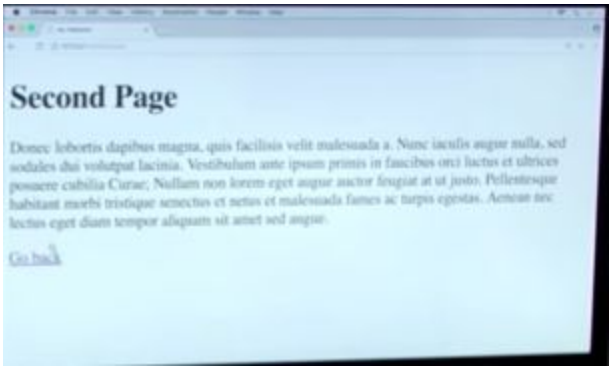
application.py | index.html | more.html

```
3     <head>
4         <title>My Website!</title>
5     </head>
6     <body>
7         <h1>Second Page</h1>
8
9         <p>
10            Donec lobortis dapibus magna, quis fac  velit malesuad
11        </p>
12
13        <a href="{{ url_for('index') }}">Go bac
14    </body>
15 </html>
```

First Page

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce placerat rutrum nisi at pellentesque. Fusce orci magna, pulvinar ut ultricies id, feugiat eu elit. Nam molestie eu lacus eu fermentum. Fusce eleifend tempus sapien eu maximus. Donec lobortis dapibus magna, quis facilisis velit malesuada a. Nunc iaculis augue nulla, sed sodales dui volutpat lacinia.

See more...



10.Avoiding of repeating code and create base file with jinja

application.py layout.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>My Website!</title>
    </head>
    <body>
        <h1>{% block heading %}{% endblock %}</h1>

        {% block body %}
        {% endblock %}
    </body>
</html>
```

application.py layout.html index.html

```jinja
{% extends "layout.html" %}

{% block heading %}
    First Page
{% endblock %}

{% block body %}
    <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce
    </p>

    <a href="{{ url_for('more') }}">See more...</a>
{% endblock %}
```

more.html — ~/web/src/inheritance

application.py layout.html index.html more.html

```
{% extends "layout.html" %}

{% block heading %}
    Second Page
{% endblock %}

{% block body %}
    <p>
        Donec lobortis dapibus magna, quis facilisis velit malesuada a.
    </p>

    <a href="{{ url_for('index') }}">Go back</a>
{% endblock %}
```

11.Working with Post and Get methods

application.py — ~/web/src/forms

application.py

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/hello", methods=["POST"])
def hello():
    name = request.form.get("name")
    return render_template("hello.html", name=name)
```

index.html — ~/web/src/forms

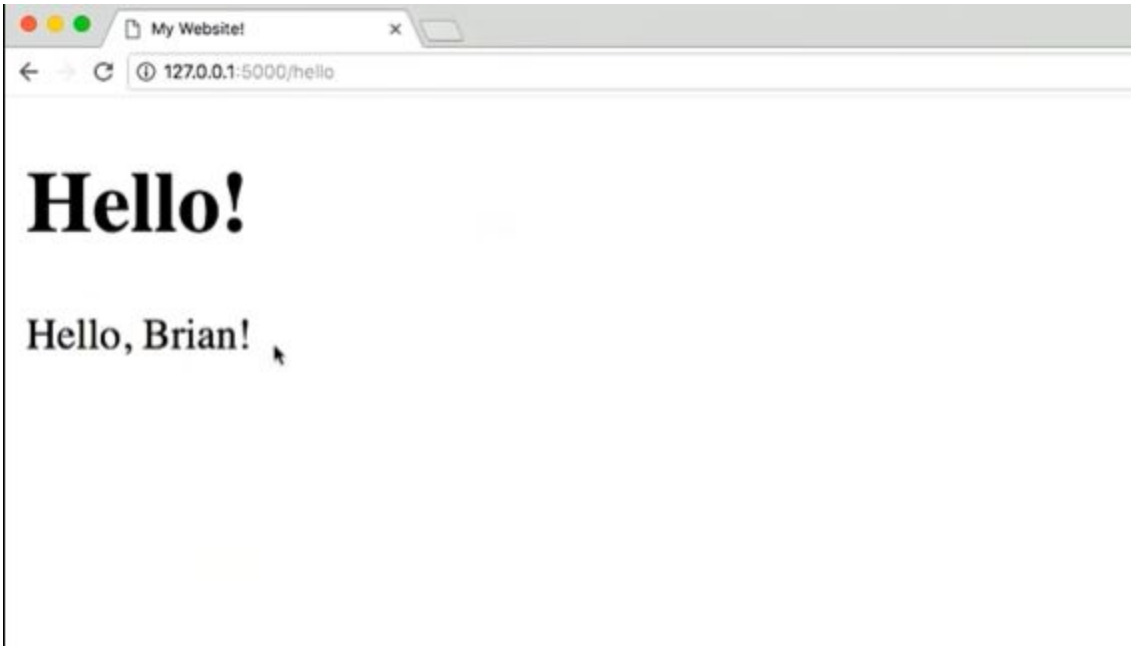
application.py | index.html

```
{% extends "layout.html" %}

{% block heading %}
    First Page
{% endblock %}

{% block body %}
    <form action="{{ url_for('hello') }}" method="post">
        <input type="text" name="name" placeholder="Enter Your Name">
        <button>Submit</button>
    </form>
{% endblock %}
```

application.py | hello.html | index.html

```
{% extends "layout.html" %}

{% block heading %}
    Hello!
{% endblock %}

{% block body %}
    Hello, {{ name }}!
{% endblock %}
```


My Website!
127.0.0.1:5000
First Page
Brian
Submit


My Website!
127.0.0.1:5000/hello
Hello!
Hello, Brian!

12.Seprating Get and Post request

```python
app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/hello", methods=["GET", "POST"])
def hello():
    if request.method == "GET":
        return "Please submit the form instead."
    else:
        name = request.form.get("name")
        return render_template("hello.html", name=name)
```

Note: and if you try to submit data by get method, the get data would be put into the url

And it's not good for security…

14.Storing data with arrays

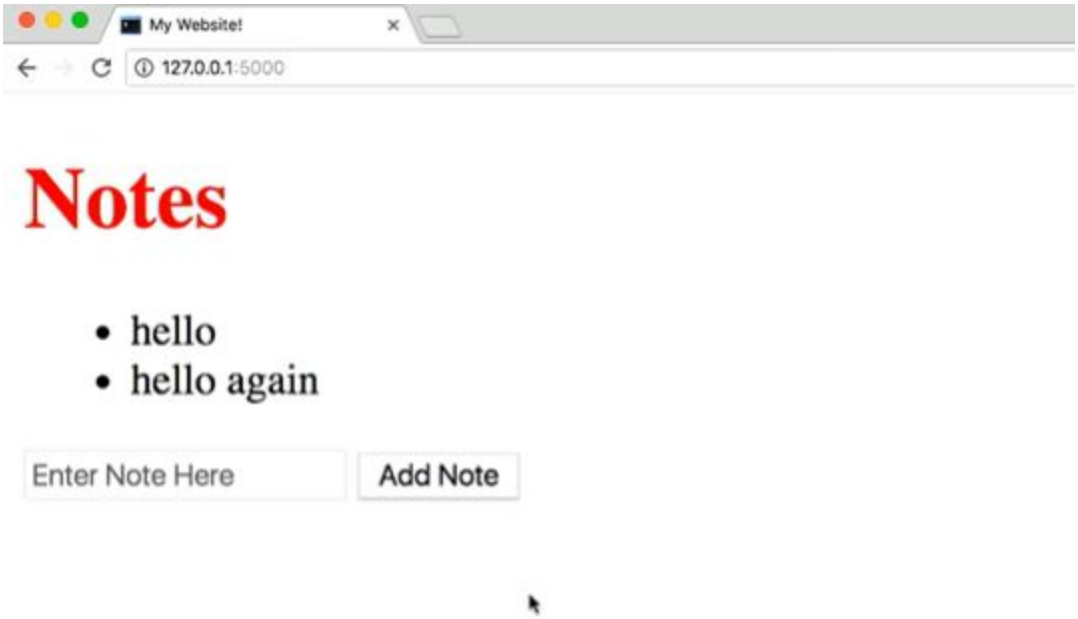
application.py — ~/web/src/notes

application.py

```python
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

notes = []

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        note = request.form.get("note")
        notes.append(note)

    return render_template("index.html", notes=notes)
```

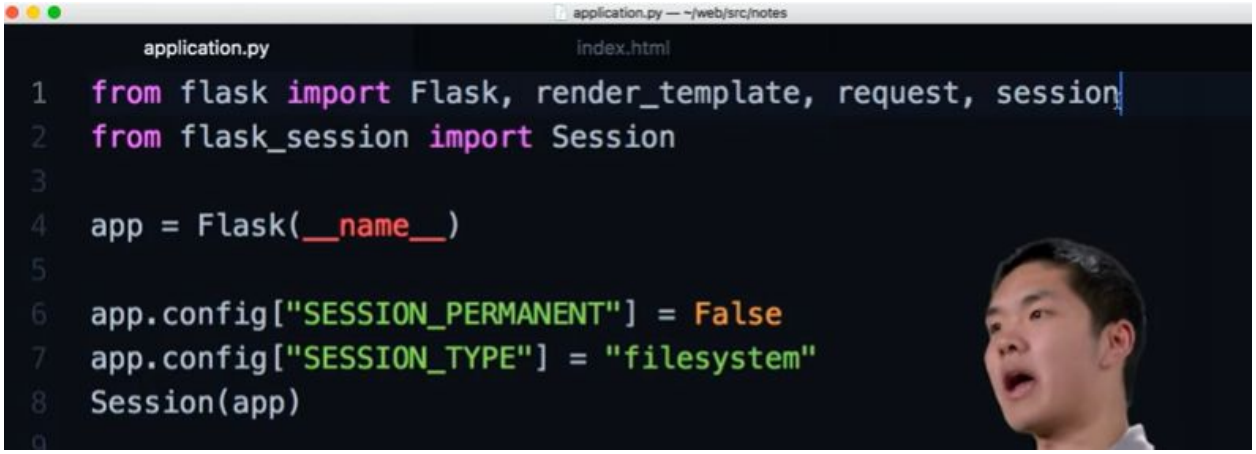
application.py index.html

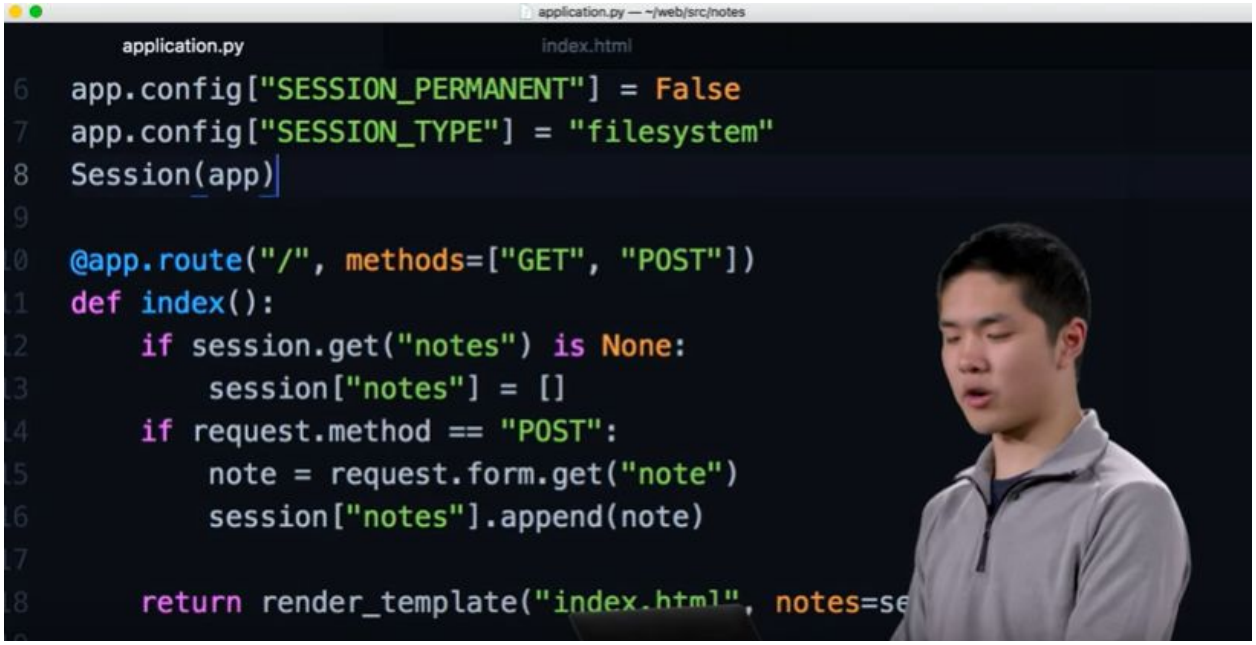
```html
{% block body %}

    <ul>
        {% for note in notes %}
            <li>{{ note }}</li>
        {% endfor %}
    </ul>

    <form action="{{ url_for('index') }}" method="post">
        <input type="text" name="note" placeholder="Enter Note Here">
        <button>Add Note</button>
    </form>
{% endblock %}
```

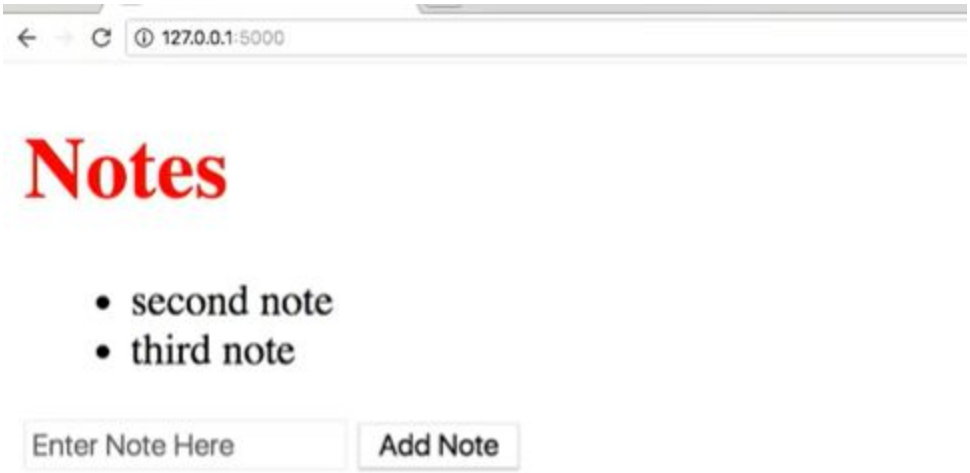

My Website!
127.0.0.1:5000
Notes
hello
hello again
Enter Note Here
Add Note

15.Storing data by sessions

This is necessary for separate different users' data







Ok now what is the difference? Well first if you close the tab your data wouldn't be removed, second every single user has his own data!

The End!!