Flask

Module 5

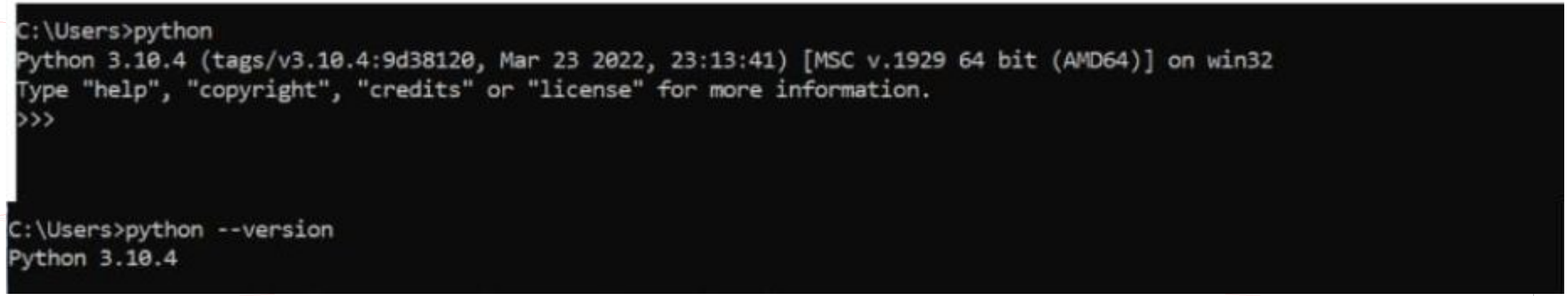
- What is Flask?
  - Open Source Python based **Web Application Framework**
    - Collection of library, modules to write web application without having to bother about low level details such and protocol, thread management etc
  - Large Community collaborators
- Inventor: Armin Ronacher with Team of Python enthusiast called Pocco
- Flask is built on **WSGI** (Web Server Gateway Interface, standard web development) Toolkit and **Jinja2** Template Engine (to render the HTML pages)
- Adopted by Pinterest, LinkedIn
- Micro Framework:Lightweight framework, provides essential components - routing, request handling, sessions

Features of Flask

- Fast Debugging
- RESTful request dispatching.
- Application Programming Interface
- Flexible configuration
- Integrated Unit Testing

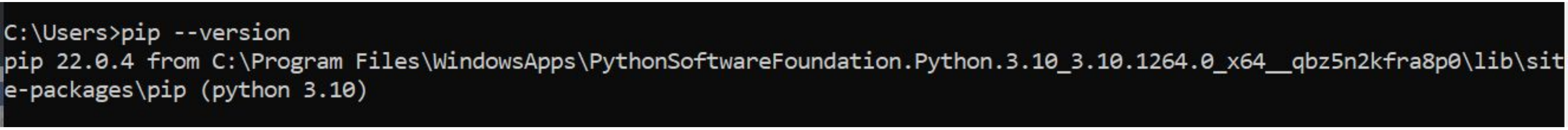
Installation

- Prerequisite:
  - Download Python 3.7 or newer
  - https://www.python.org/downloads/ (don’t forget to set PATH)



```
C:\Users>python
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

C:\Users>python --version
Python 3.10.4
```

  - pip (comes along with python)



```
C:\Users>pip --version
pip 22.0.4 from C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.1264.0_x64__qbz5n2kfra8p0\lib\sit
e-packages\pip (python 3.10)
```

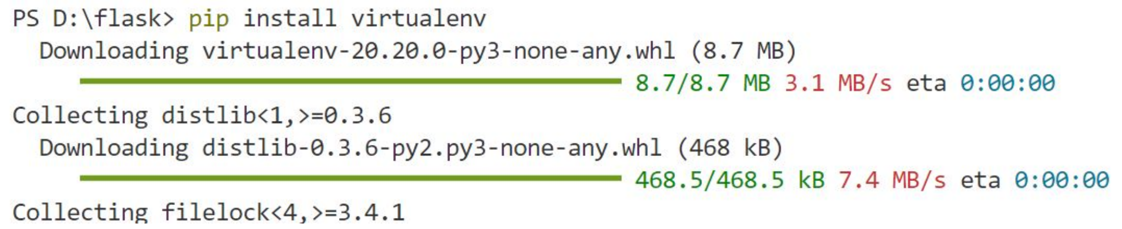
Using virtual environment

Make a directory and move into the directory

mkdir flask

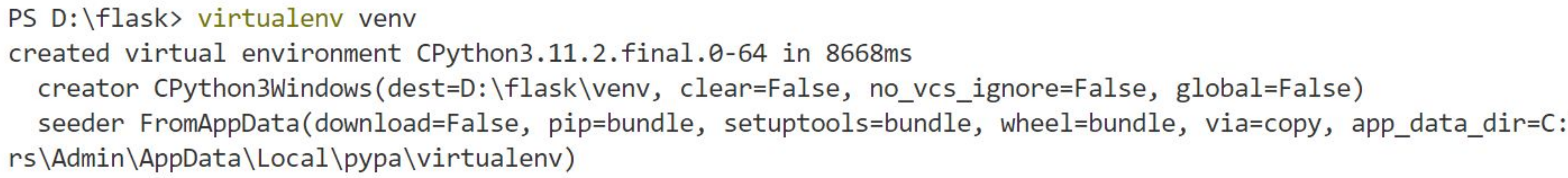
cd flask

Install virtualenv



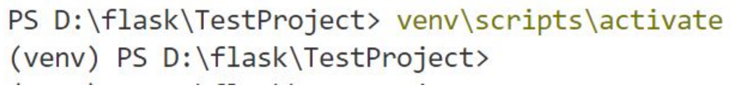
```
PS D:\flask> pip install virtualenv
  Downloading virtualenv-20.20.0-py3-none-any.whl (8.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 3.1 MB/s eta 0:00:00
Collecting distlib<1,>=0.3.6
  Downloading distlib-0.3.6-py2.py3-none-any.whl (468 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 468.5/468.5 kB 7.4 MB/s eta 0:00:00
Collecting filelock<4,>=3.4.1
```

Create a virtual environment



```
PS D:\flask> virtualenv venv
created virtual environment CPython3.11.2.final.0-64 in 8668ms
  creator CPython3Windows(dest=D:\flask\venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:
rs\Admin\AppData\Local\pypa\virtualenv)
```

Activate virtual environment

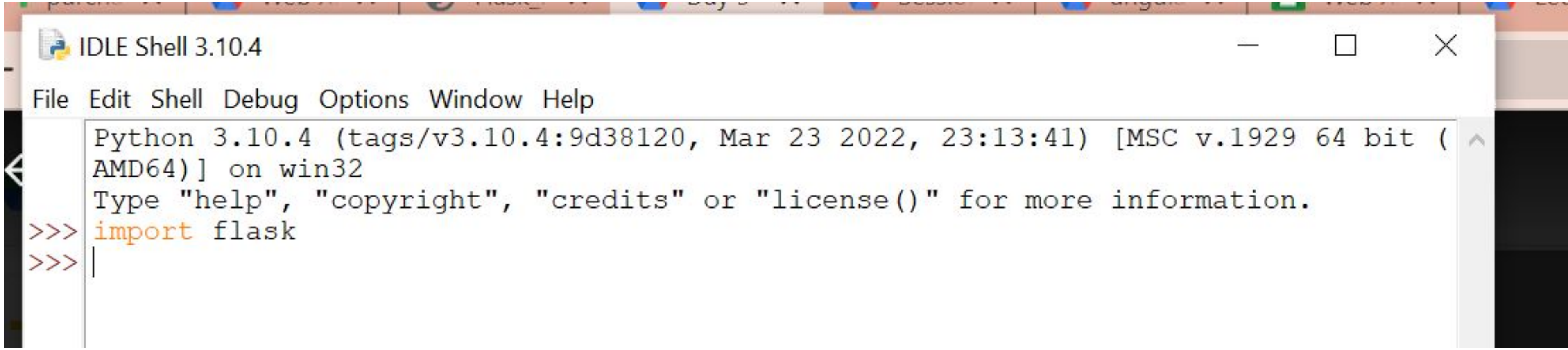


```
PS D:\flask\TestProject> venv\scripts\activate
(venv) PS D:\flask\TestProject>
```

install flask environment within virtual environment

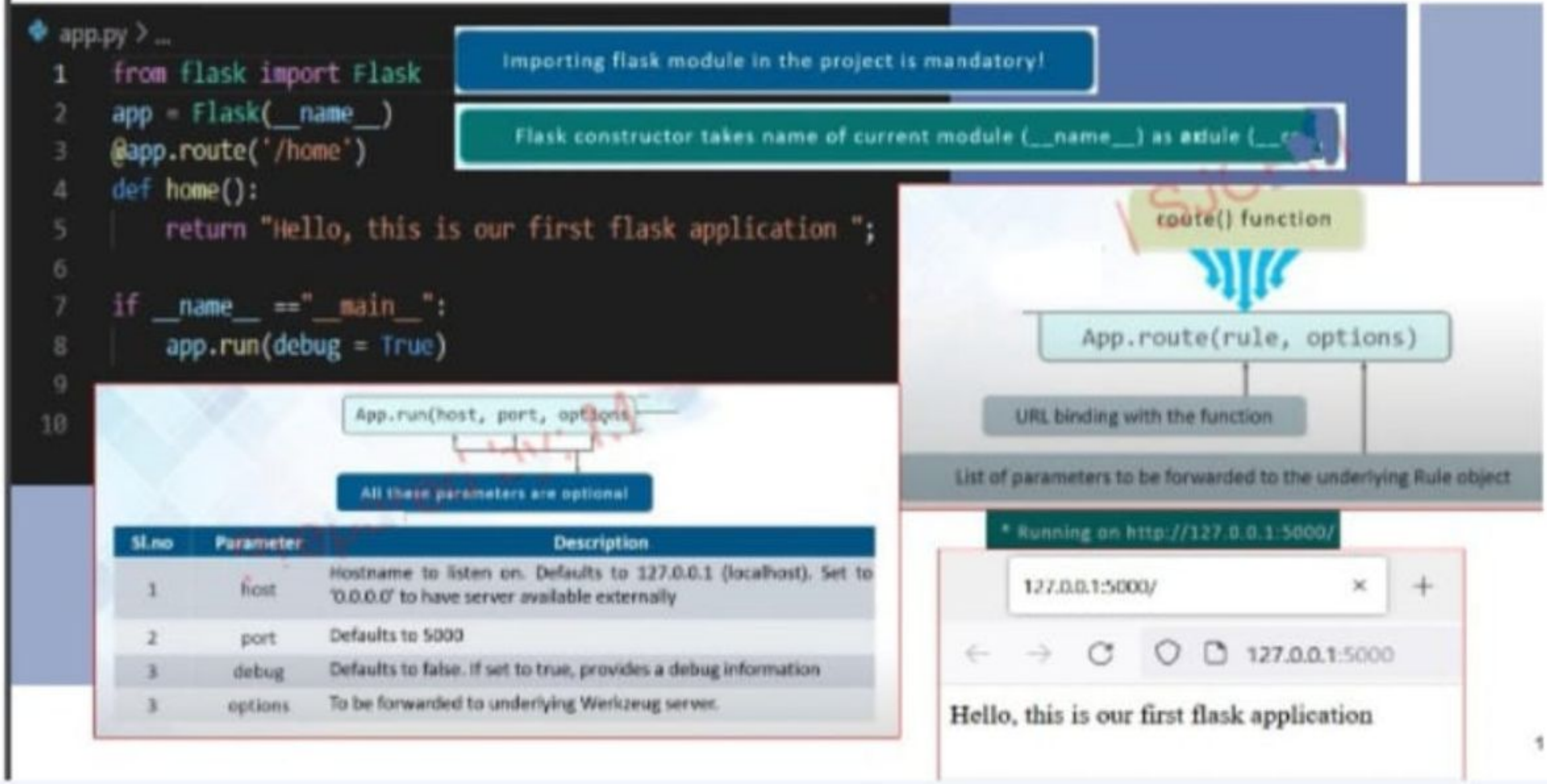
>pip install flask

- Verify if flask has been installed property
  - Open IDLE
  - Import flask command should not through an error



```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import flask
>>> 
```

First App



app.py > ...

```python
from flask import Flask
app = Flask(__name__)
@app.route('/home')
def home():
    return "Hello, this is our first flask application ";

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

Importing flask module in the project is mandatory!

Flask constructor takes name of current module (__name__) as a module (__[illegible]

route() function
App.route(rule, options)
URL binding with the function
List of parameters to be forwarded to the underlying Rule object
App.run(host, port, options)
All these parameters are optional

Sl.no	Parameter	Description
1	host	Hostname to listen on. Defaults to 127.0.0.1 (localhost). Set to '0.0.0.0' to have server available externally
2	port	Defaults to 5000
3	debug	Defaults to false. If set to true, provides a debug information
3	options	To be forwarded to underlying Werkzeug server.

```
* Running on http://127.0.0.1:5000/
```

Hello, this is our first flask application

- To run the app

```
python -m flask --app .\app.py run
```

Or just

```
python -m flask  run
```

App Routing

- This helps the user remember the URLs.
- Routes in Flask are mapped to Python functions

```
@app.route('/')
  def index():
```

- The route() decorator, @app.route(), binds a URL to a function.
- If you want the route /hello, you can bind it to the hello_world() function like this:

```
@app.route('/hello')
def hello_world():
  return "hello world"
```

- Parameters can be used when creating routes. A parameter can be a string (text) like this: /product/cookie. (make URL Dynamic)

```
@app.route('/product/<name>')
def get_product(name):
 return "The product is " + str(name)
```

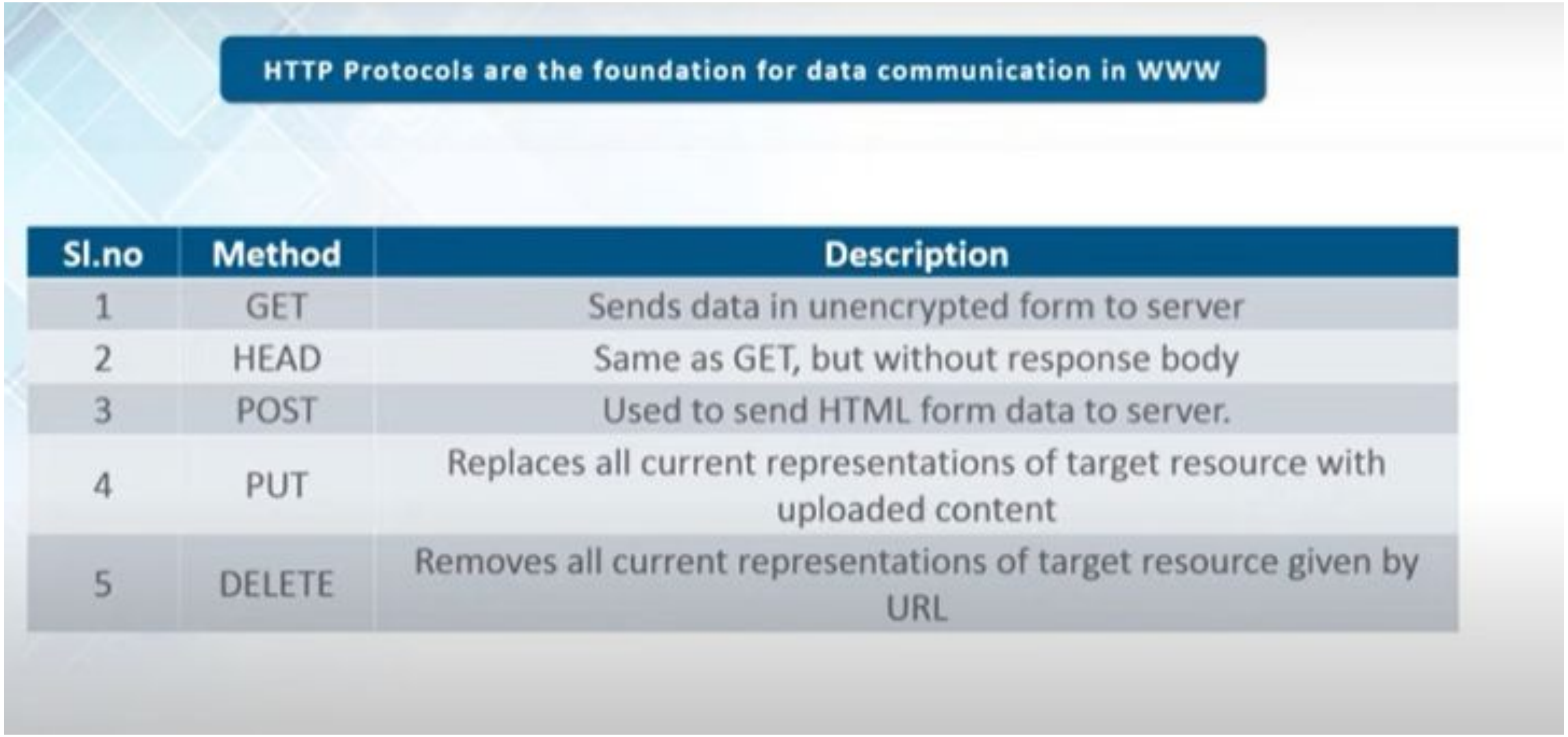
- Alternative for app routing is addURL
- This approach is mainly used in case we are importing the view function from another module
- Don’t need decorator
- Just add functions
- add_url_rule(<url rule>, <endpoint>, <view function>)

```
def show_user(username):
    # Greet the user
    return f'Hello {username} !'

app.add_url_rule('/user/<username>', 'show_user', show_user)

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

URL Building

- The url_for() function is used to build a URL to the specific function dynamically.
- The first argument is the name of the specified function, and then we can pass any number of keyword argument corresponding to the variable part of the URL.
- This function is useful in the sense that we can avoid hard-coding the URLs into the templates by dynamically building them using this function.

Flask HTTP Method



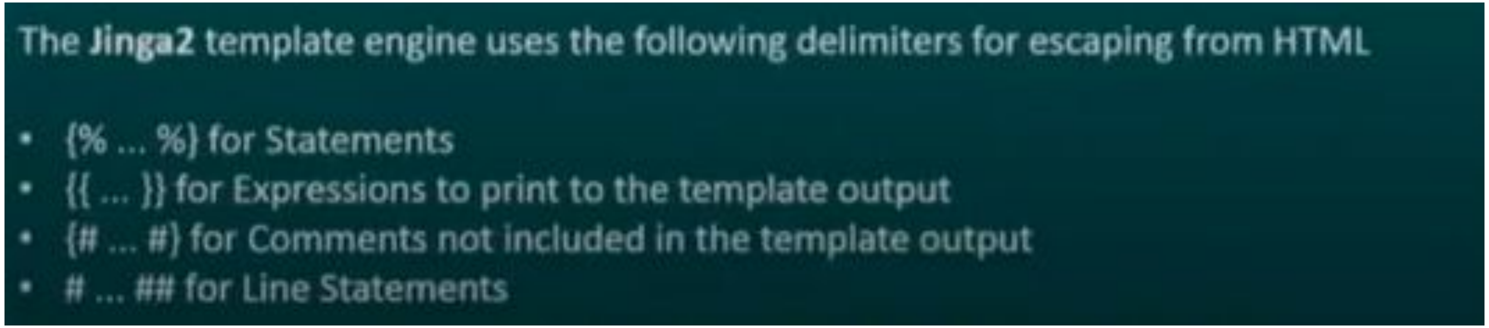
HTTP Protocols are the foundation for data communication in WWW

Sl.no	Method	Description
1	GET	Sends data in unencrypted form to server
2	HEAD	Same as GET, but without response body
3	POST	Used to send HTML form data to server.
4	PUT	Replaces all current representations of target resource with uploaded content
5	DELETE	Removes all current representations of target resource given by URL

Templates - Jinja

- Flask uses templates to expand the functionality of a web application while maintaining a simple and organized file structure.
- Templates are enabled using the Jinja2 template engine and allow data to be shared and processed before being turned in to content and sent back to the client.
- Jinja template library to render templates
- Integrate some datasource to render it as a HTML page
- Feature
    - Sandbox execution mode.
    - Powerful automatic HTML escaping system for cross site scripting prevention.
    - Template inheritance which makes it possible to use similar layout for all templates.
    - Easy to debug with debugging system that integrates compile and runtime errors into standard Python traceback system.
    - Optional Ahead of time template compilation.

- flask will try to find the HTML file In the template folder, in the same folder in which the script is present
- render_template() function renders HTML files for display in the web browser
- Handy when you need to create dynamic pages

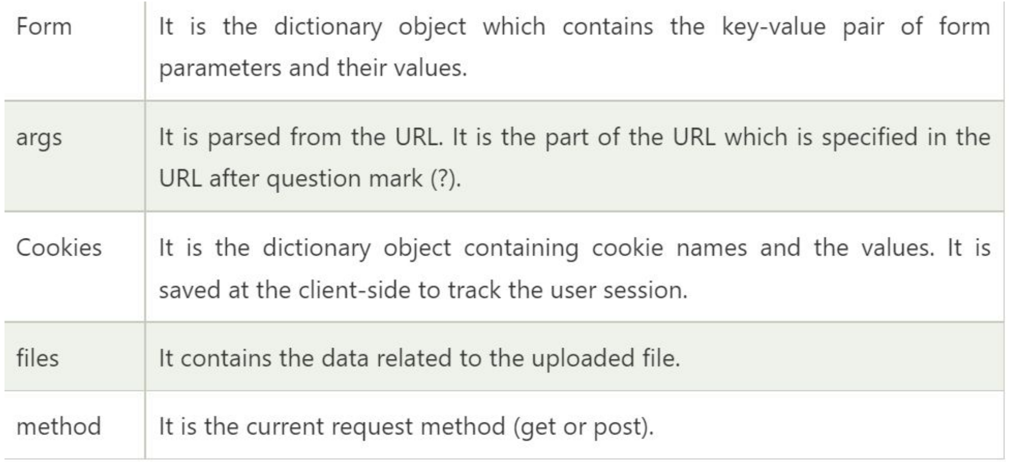


The **Jinga2** template engine uses the following delimiters for escaping from HTML

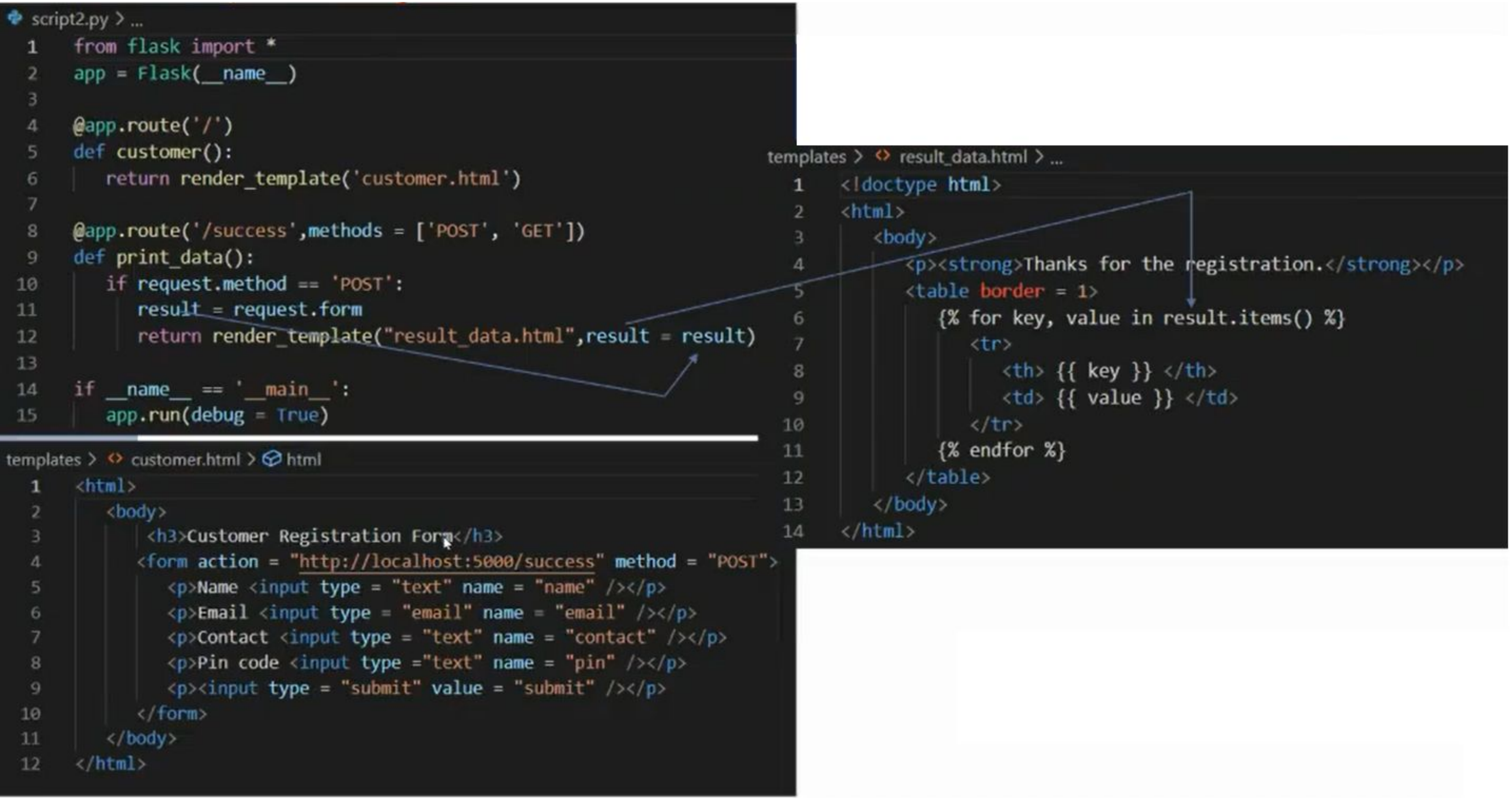
- {% ... %} for Statements
- {{ ... }} for Expressions to print to the template output
- {# ... #} for Comments not included in the template output
- # ... ## for Line Statements

Flask Request Object

- Web applications frequently require processing incoming request data from users.
- This payload can be in the shape of query strings, form data, and JSON objects.
- Flask, like any other web framework, allows you to access the request data.



Form	It is the dictionary object which contains the key-value pair of form parameters and their values.
args	It is parsed from the URL. It is the part of the URL which is specified in the URL after question mark (?).
Cookies	It is the dictionary object containing cookie names and the values. It is saved at the client-side to track the user session.
files	It contains the data related to the uploaded file.
method	It is the current request method (get or post).



```python
from flask import *
app = Flask(__name__)

@app.route('/')
def customer():
    return render_template('customer.html')

@app.route('/success',methods = ['POST', 'GET'])
def print_data():
    if request.method == 'POST':
        result = request.form
        return render_template("result_data.html",result = result)

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

templates > customer.html > html

```html
<html>
    <body>
        <h3>Customer Registration Form</h3>
       <form action = "http://localhost:5000/success" method = "POST">
          <p>Name <input type = "text" name = "name" /></p>
          <p>Email <input type = "email" name = "email" /></p>
          <p>Contact <input type = "text" name = "contact" /></p>
          <p>Pin code <input type ="text" name = "pin" /></p>
          <p><input type = "submit" value = "submit" /></p>
       </form>
    </body>
</html>
```

templates > result_data.html > ...

```html
<!doctype html>
<html>
   <body>
      <p><strong>Thanks for the registration.</strong></p>
      <table border = 1>
         {% for key, value in result.items() %}
            <tr>
               <th> {{ key }} </th>
               <td> {{ value }} </td>
            </tr>
         {% endfor %}
      </table>
   </body>
</html>
```

Flask cookies

- Cookies are stored on the client’s computer as text files.
- Aim is to remember and track data that is relevant to customer usage for better visitor experience and website statistics.
- The Flask Request object contains the properties of the cookie.
- It is a dictionary object for all cookie variables and their corresponding values, and the client is transferred.In addition to this, cookies also store the expiration time, path, and domain name of its website.
- Cookies are set on the response object.
    - Server sends the Cookie to the user along with the response. It is done using the make_response() function.
    - Once the response is set, we use the set_cookie() function to attach the cookie to it.

- The cookie takes the attributes:

```
response.set_cookie('<Title>','<Value>','<Expiry Time>')
```

- The cookie is sent back along with the Request to the server.
    - to get the Cookie back from the user, use request.cookies.get() function.

File Uploading

- File uploading is the process of transmitting the binary or normal files to the saver.
- It requires an HTML form whom enctype property is set to "multipart/form-data" to publish the file to the URL.
- The server side flask fetched the file from the request object using request.files[ ] Object.
- On successfully uploading the file, it is saved to the desired location on the server