hello, section!

week 7

sandbox: http://bit.ly/2zegz2Y

pset 6 recap

(a preview of) MVC

MVC

why?

structural foundation

encapsulation → delegation → scalability

MVC

organization of a web application:

1. Model
2. View
3. Controller

MVC

organization of a web application:

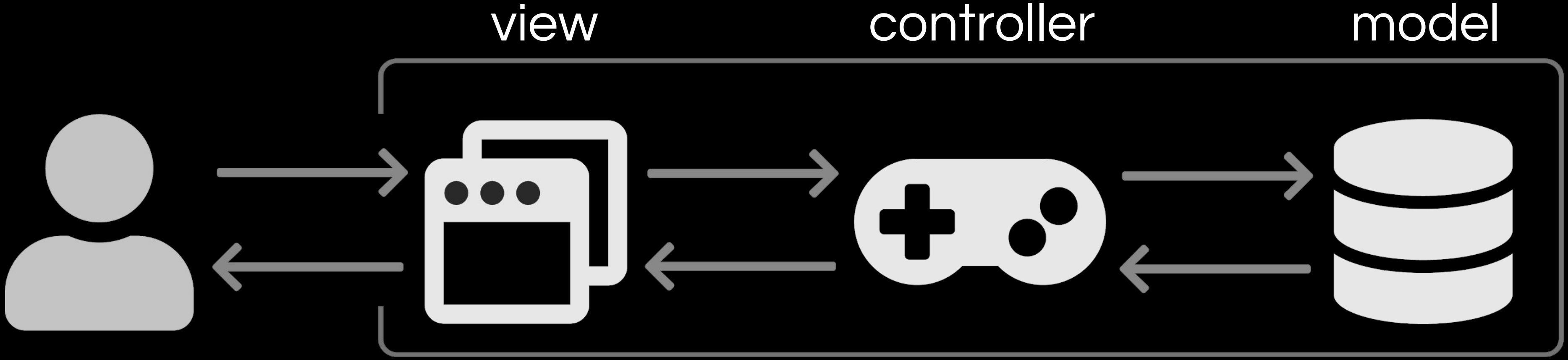
1. Model → Data 📈
2. View → Interface 🤗
3. Controller → Logic 🧠

MVC

organization of a web application:

1. Model → CSV 📈
2. View → HTML/CSS 🤗
3. Controller → Python 🧠

MVC



Flask

Flask

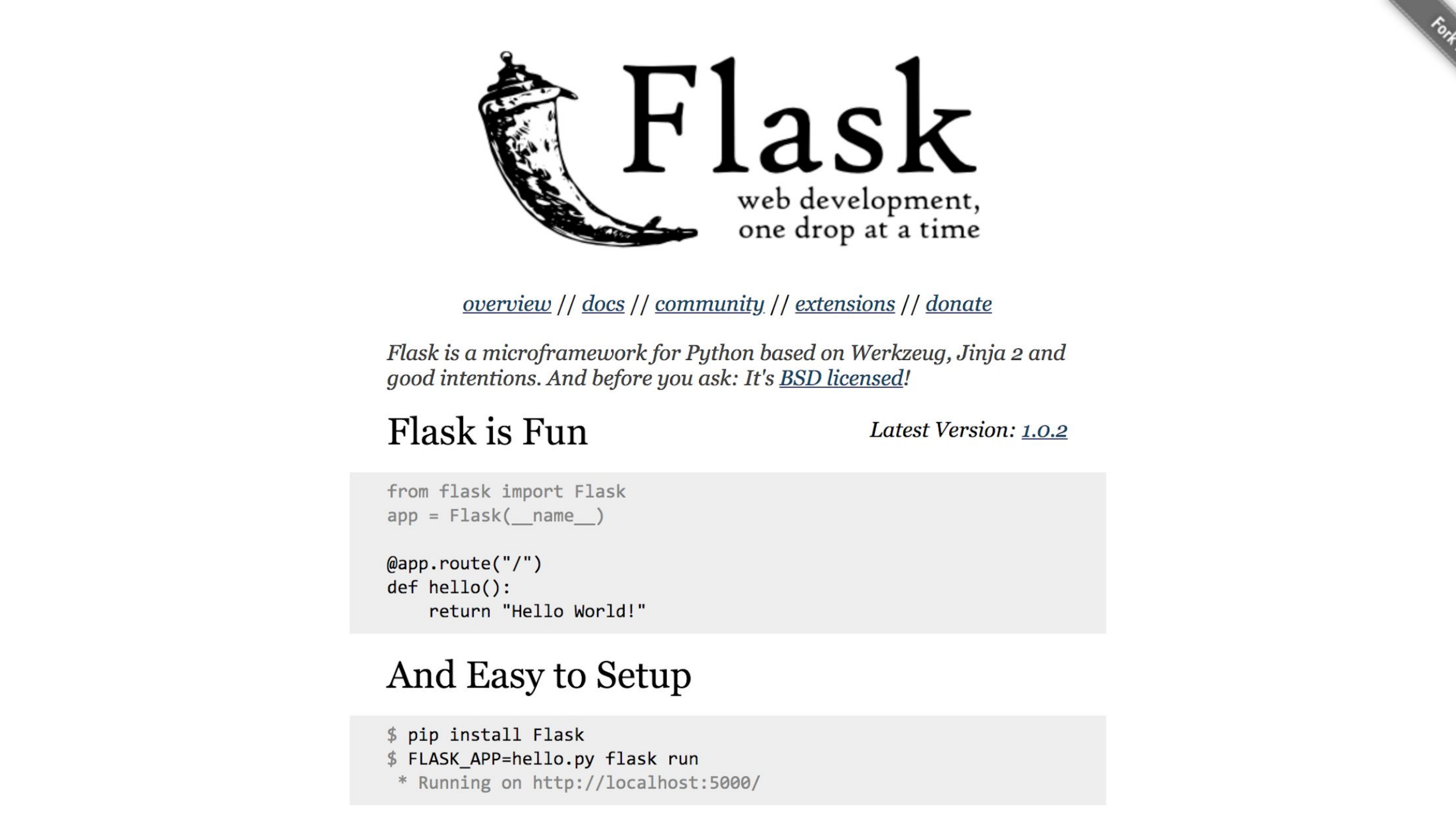
Python-based web microframework

Flask

Python-based web microframework

↓

handles the foundation of a web application for you!



overview // docs // community // extensions // donate

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's BSD licensed!

Flask is Fun

Latest Version: 1.0.2

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

@app.route("/")
def hello():
    return "Hello World!"
```

And Easy to Setup

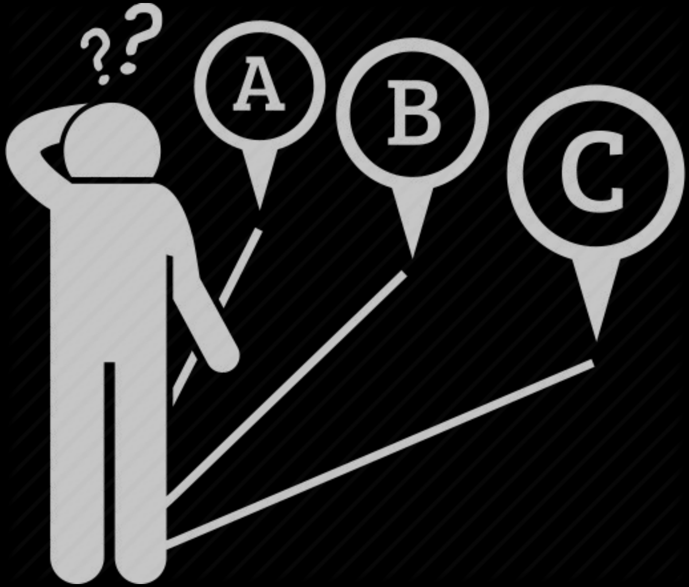
```
$ pip install Flask
$ FLASK_APP=hello.py flask run
 * Running on http://localhost:5000/
```

Flask

application.py

```
import CS50

def main():
    # python code
```



Flask

application.py

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "You are at the index!"
```

Flask

application.py

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "You are at the index!"
```

```
$ flask run
```

Flask

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "You are at the index!"

@app.route("/sample")
def sample():
    return "You have reached the sample page!"
```

Flask

HTML is *static*

Python is *dynamic*

```
<html>
   <head>
      <title>
         Current Time
      </title>
   </head>
   <body>
      The current time is 10:58
   </body>
</html>
```

```
<html>
   <head>
      <title>
         Current Time
      </title>
   </head>
   <body>
      The current time is 10:59
   </body>
</html>
```

```
<html>
   <head>
      <title>
         Current Time
      </title>
   </head>
   <body>
      The current time is 11:00
   </body>
</html>
```

```html
<html>
   <head>
      <title>
         Current Time
      </title>
   </head>
   <body>
      The current time is 11:01
   </body>
</html>
```

Flask

```
from flask import Flask
from datetime import datetime
from pytz import timezone

app = Flask(__name__)

@app.route("/")
def time():
    now = datetime.now(timezone('America/New_York'))
    return f"The current date and time is {now}."
```

decorators

decorators

a *function* that modifies the behavior of *other functions*

decorators

```
def override(func):
    def incr():
        return func() + 1
    return incr

@override
def one():
    return 1

print(one())
```

decorators

```
def override(func):
    def incr():
        return func() + 1
    return incr

@override
def one():
    return 1

print(one())
```

prints 2

decorators

`@app.route()`

- specify URL associated with a function and which methods (GET, POST) to apply

`@login-required`

- written by CS50
- only allow usage of certain functions if the user is logged in

decorators

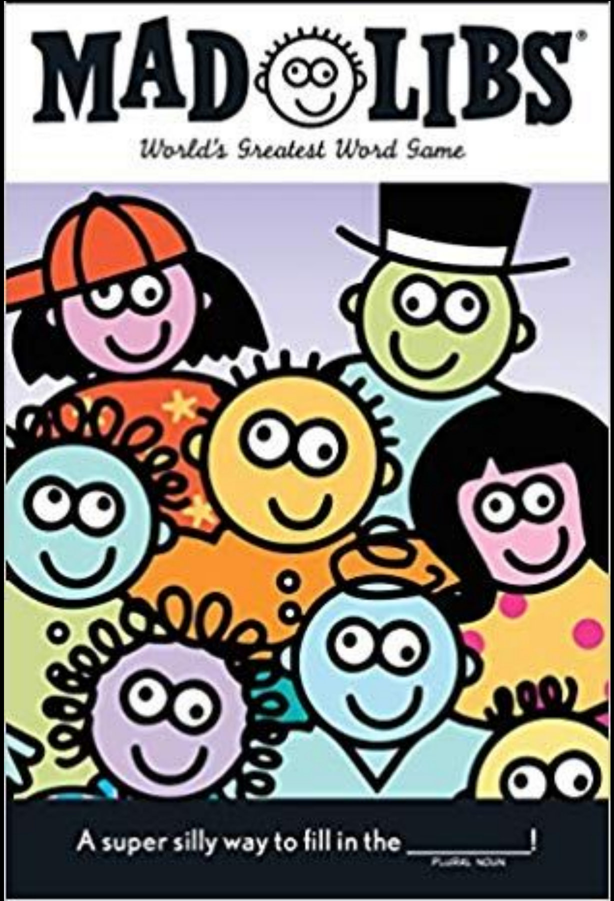
```
@app.route("/show/<number>")
def show(number):
   return f"You passed in {number}."
```

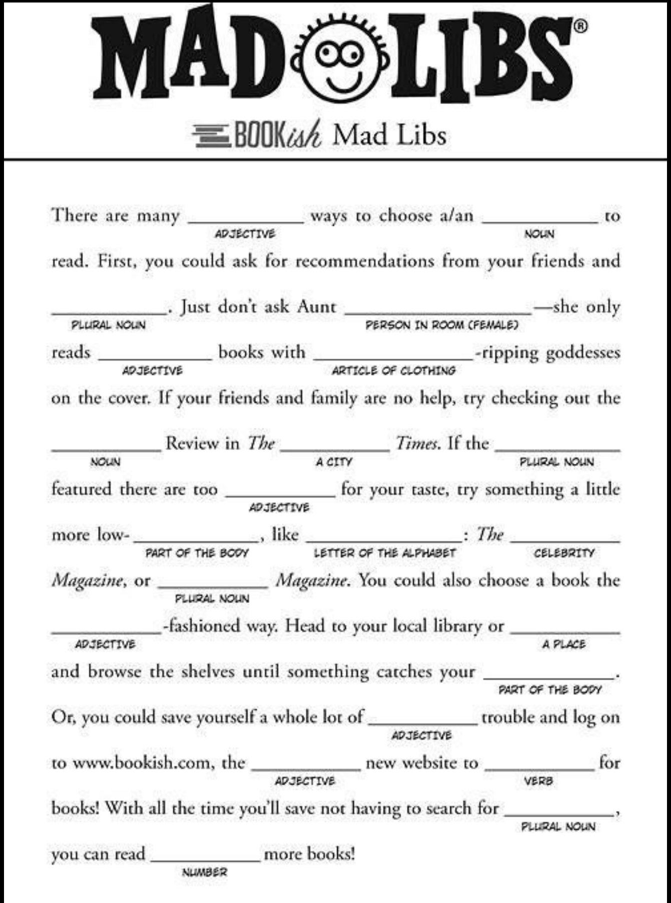
decorators

```
@app.route("/login", methods=['GET', 'POST'])
def login():
  if request.method == "POST":
    # do one thing
  else:
    # do a different thing
```

Jinja

Jinja


MAD LIBS®
World's Greatest Word Game
A super silly way to fill in the __________!
PLURAL NOUN



MAD LIBS®

BOOKish Mad Libs

There are many ____ADJECTIVE____ ways to choose a/an ____NOUN____ to read. First, you could ask for recommendations from your friends and ____PLURAL NOUN____. Just don't ask Aunt ____PERSON IN ROOM (FEMALE)____—she only reads ____ADJECTIVE____ books with ____ARTICLE OF CLOTHING____-ripping goddesses on the cover. If your friends and family are no help, try checking out the ____NOUN____ Review in *The* ____A CITY____ *Times*. If the ____PLURAL NOUN____ featured there are too ____ADJECTIVE____ for your taste, try something a little more low-____PART OF THE BODY____, like ____LETTER OF THE ALPHABET____: *The* ____CELEBRITY____ *Magazine*, or ____PLURAL NOUN____ *Magazine*. You could also choose a book the ____ADJECTIVE____-fashioned way. Head to your local library or ____A PLACE____ and browse the shelves until something catches your ____PART OF THE BODY____. Or, you could save yourself a whole lot of ____ADJECTIVE____ trouble and log on to www.bookish.com, the ____ADJECTIVE____ new website to ____VERB____ for books! With all the time you'll save not having to search for ____PLURAL NOUN____, you can read ____NUMBER____ more books!

Jinja

templating language

Jinja

templating language

↓

adding logic to HTML

Jinja

- template inheritance
- Python-like logic & variables in HTML

Jinja

layout.html

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <meta name="viewport" content="initial-scale=1, width=device-width">
        <title>hello</title>
    </head>
    <body>
        {% block body %}{% endblock %}
    </body>
</html>
```

Jinja

layout.html

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <meta name="viewport"
         content="initial-scale=1,
         width=device-width">
        <title>hello</title>
    </head>
    <body>
        {% block body %}{% endblock %}
    </body>
</html>
```

index.html

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

{% block body %}
    hello, world!
{% endblock %}
```

Jinja

`{% . . . %}` → control flow, logic

`{{ … }}` → "print" as HTML

Jinja

- application.py
- form.html
- table.html

Jinja

form.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Multiplication Table
        </title>
    </head>
    <body>
        <form action="/" method="post">
            <input name="size" type="number" placeholder="dimension"/>
            <input name="submit" type="submit" />
        </form>
    </body>
</html>
```

Jinja

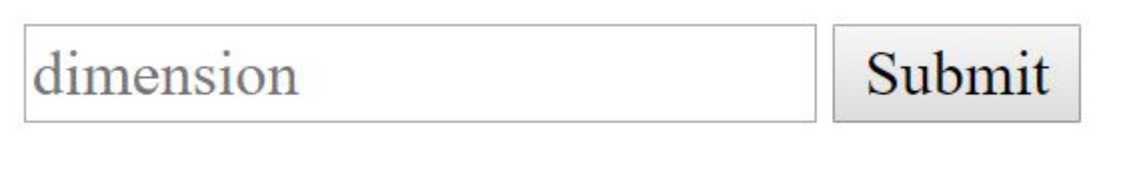
form.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Multiplication Table
        </title>
    </head>
    <body>
        <form action="/" method="post">
            <input name="size" type="number" placeholder="dimension"/>
            <input name="submit" type="submit" />
        </form>
    </body>
</html>
```

Jinja

form.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Multiplication Table
        </title>
    </head>
    <body>
        <form action="/" method="post">
            <input name="size" type="number" placeholder="dimension"/>
            <input name="submit" type="submit" />
        </form>
    </body>
</html>
```



Jinja

application.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

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

Jinja

application.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])

def mult_table():
    return render_template("form.html")
```

Jinja

application.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])

def mult_table():
    if request.method == "GET":
        return render_template("form.html")
```

Jinja

application.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])

def mult_table():
    if request.method == "GET":
        return render_template("form.html")
    # our form is set up to submit via POST
    elif request.method == "POST":
        return render_template("table.html")
```

Jinja

table.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
  <body>
        <table>

          <tr>

              <td>

              </td>

          </tr>

        </table>
  </body>
</html>
```

Jinja

application.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])

def mult_table():
    if request.method == "GET":
        return render_template("form.html")
    # our form is set up to submit via POST
    elif request.method == "POST":
        return render_template("table.html", dim=request.form.get("size"))
```

Jinja

table.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
  <body>
        <table>
         // loop to repeat “dim” times (“dim” # of rows)
          <tr>
         // loop to repeat “dim” times (“dim” # of columns)
              <td>
         // print out that value of the cell between <td>s
              </td>

          </tr>

        </table>
  </body>
</html>
```

Jinja

table.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
  <body>
        <table>
         {% for i in range(dim) %}
          <tr>
              {% for j in range(dim) %}
              <td>
         // print out that value of the cell between <td>s
              </td>
              {% endfor %}
          </tr>
          {% endfor %}
        </table>
  </body>
</html>
```

Jinja

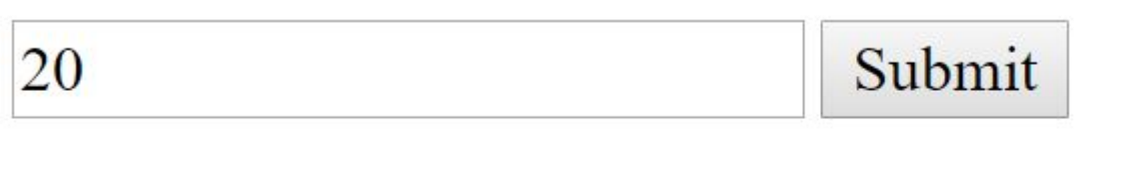
table.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
  <body>
        <table>
         {% for i in range(dim) %}
          <tr>
              {% for j in range(dim) %}
              <td>
                  {{ (i + 1) * (j + 1) }}
              </td>
              {% endfor %}
          </tr>
          {% endfor %}
        </table>
  </body>
</html>
```

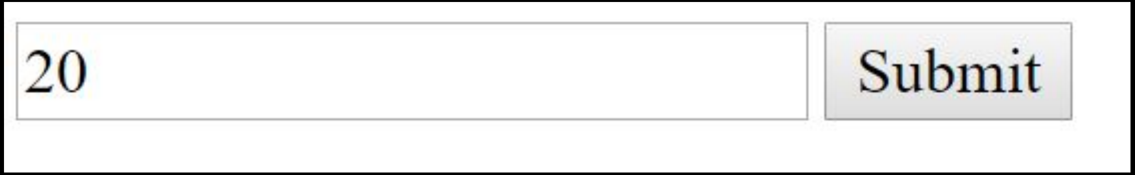
Jinja

table.html

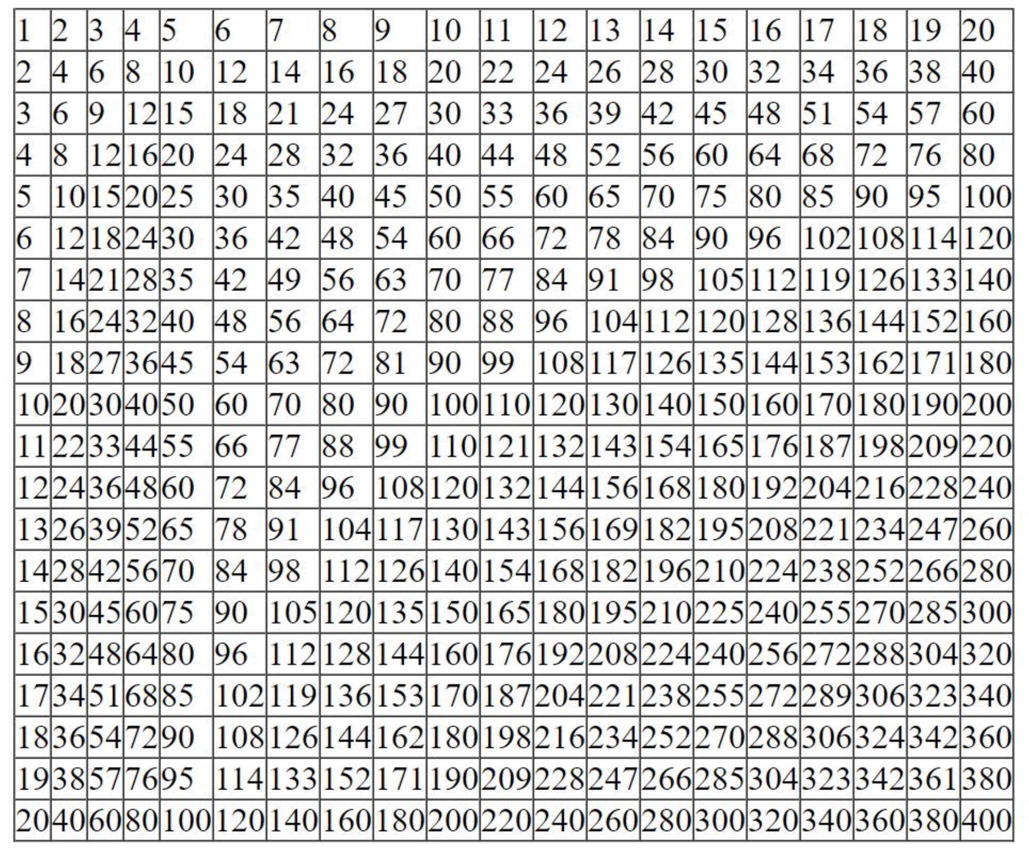
```
<!DOCTYPE html>
<html>
  <head>
    <title>Table</title>
  </head>
  <body>
        <table>
         {% for i in range(dim|int) %}
          <tr>
              {% for j in range(dim|int) %}
              <td>
                  {{ (i + 1) * (j + 1) }}
              </td>
              {% endfor %}
          </tr>
          {% endfor %}
        </table>
  </body>
</html>
```



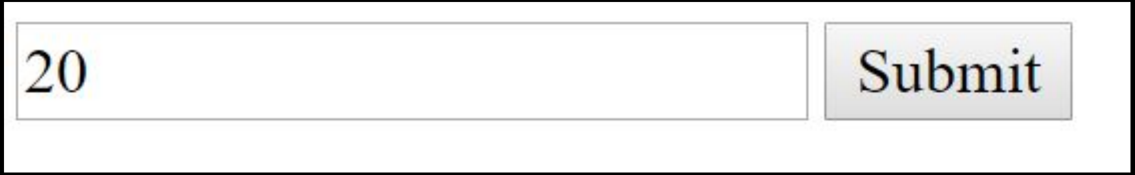
20



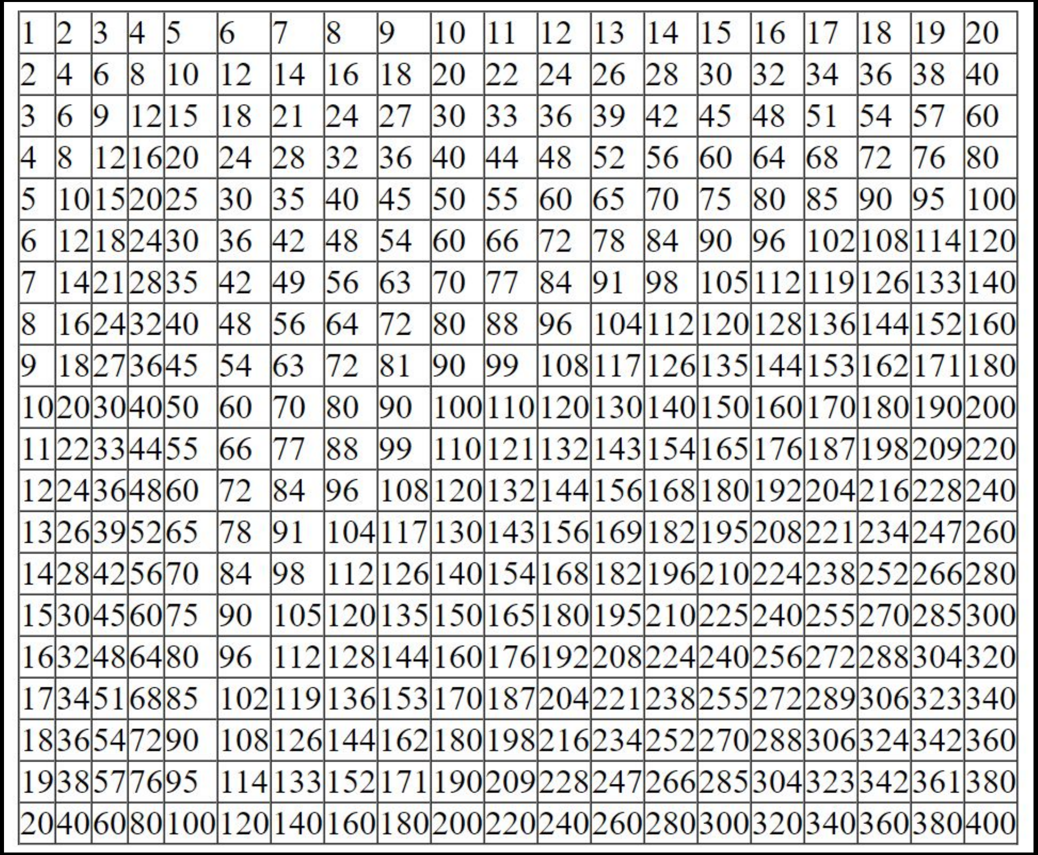
20



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133	140
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171	180
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187	198	209	220
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247	260
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238	252	266	280
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270	285	300
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289	306	323	340
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360
19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323	342	361	380
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400



20



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133	140
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171	180
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187	198	209	220
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247	260
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238	252	266	280
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270	285	300
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289	306	323	340
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360
19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323	342	361	380
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400

It may not look beautiful… but that’s what CSS is for!

Ajax

Ajax

asynchronous Javascript and XML

Ajax

asynchronous Javascript and XML

↓

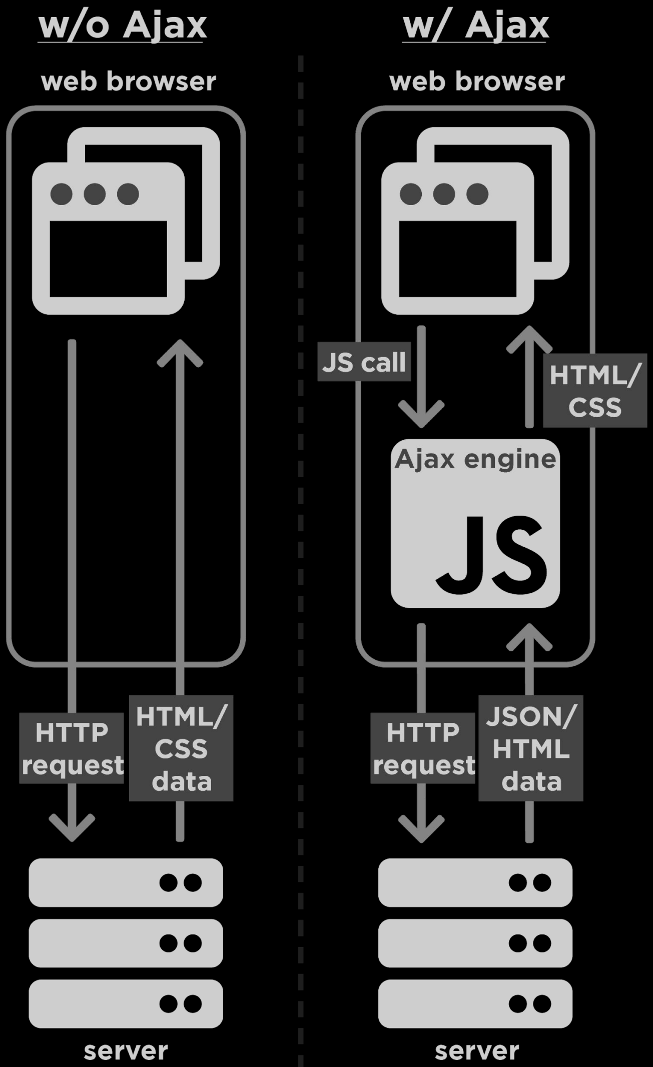
constantly updating webpage w/o needing to refresh

examples

sports scores

emails

what are other examples?



w/o Ajax

web browser

HTTP request

HTML/ CSS data

server

w/ Ajax

web browser

JS call

HTML/ CSS

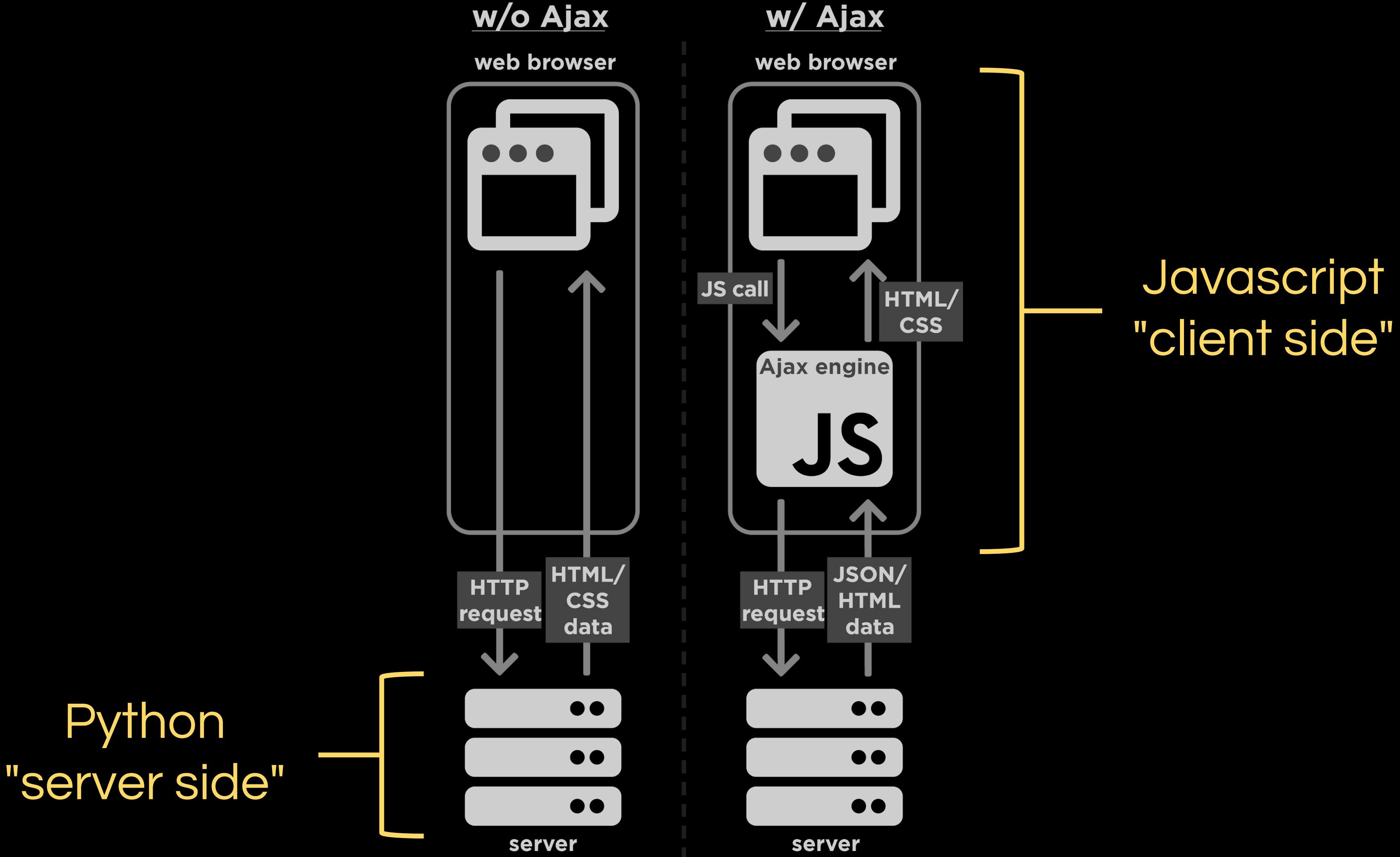
Ajax engine

JS

HTTP request

JSON/ HTML data

server



w/o Ajax

web browser

HTTP request

HTML/ CSS data

server

w/ Ajax

web browser

JS call

HTML/ CSS

Ajax engine

JS

HTTP request

JSON/ HTML data

server

Javascript "client side"

Python "server side"

Ajax

```
let xhttp = new XMLHttpRequest();
```

Ajax

- `readyState`
  - 0 (no request yet)
  - 1, 2, 3
  - 4 (request finished, response ready)
- `status` property will (hopefully) be 200
- `open()` (defines the request)
- `send()` (sends the request)

Ajax

```
function ajax_request(argument)
{
    let aj = new XMLHttpRequest();
    aj.onreadystatechange = function() {
        if (aj.readyState == 4 && aj.status == 200)
        {
            // do something to the page
        }
    };

    aj.open("GET", <the URL>, true);
    aj.send();
}
```

pset 7 tips & tricks

- Google Python set & list functions!
- use `querySelector()` for JavaScript validation

final project to-dos

- ➔ Preproposals due by 11:59pm on Tue 11/6
- ➔ If collaborating with 1 or 2 classmates, each of you should submit a preproposal, even if identical.