Sending HTTP Requests:

In Python, the `requests` library provides us the ability to make HTTP requests to external APIs:

15/api.py

```
1  import requests
2
3  r = requests.get("https://www.colr.org/json/color/random")
4  print(f"Status Code: {r.status_code}")
5  print(f"Character Encoding: {r.encoding}")
```

- `requests.get(...)` sends a GET request,
- `requests.post(...)` sends a POST request,
- `requests.put(...)` sends a PUT request,
- ...etc...

The requests library is just a wrapper around the request and response from any HTTP web service:

15/api.py

```
7  print("== Headers ==")
8  for header in r.headers:
9    print(header + ": " + r.headers[header])
10
11 print("== Payload (text) ==")
12 print(r.text)
13
14 print("== Payload (json) ==")
15 data = r.json()
16 print(data["colors"][0]["hex"])
```

Note that:

- `r.text` returns the response as a string (at attribute).
- `r.json()` parses it for us into a dictionary for us to index into quickly (it's a function, requires the parameters)!

Receiving HTTP Requests:

The flask library allows us to receive HTTP requests:

15/app.py

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/', methods=["GET"])
5  def index():
6    return "index function!"
7
8  @app.route('/', methods=["POST"])
9  def post():
10   return "post function!"
11
12 @app.route('/hello', methods=["GET"])
13 def hello():
14   return "hello function!"
15
16 @app.route('/hello/<id>')
17 def with_id(id):
18   return f"with_id function: {id}"
19
20 @app.route('/hello')
21 def mystery():
22   return "mystery function!"
```

What happens with the following requests:

1. `GET /`
2. `POST /`
3. `PUT /`
4. `GET /hello/`
5. `GET /hello`
6. `POST /hello`
7. `PUT /hello`
8. `GET /hello/42`
9. `GET /hello/world`

Operating Systems: A Great Illusionist
Throughout this entire course, we have discussed how the operating system abstracts away the complexity of real systems:

- As a process, it appears that we have _____.
- ...and has _____!

Virtualization

Q: What is a "machine"?

Big Idea:
Map a _____ onto a _____.

- All states $\mathbf{S_X}$ can be represented on a host system $\mathbf{H(S_X)}$.
- For all sequences of transitions between $\mathbf{S_1} \Rightarrow \mathbf{S_2}$, there is a sequence of transitions between $H(\mathbf{S_1}) \Rightarrow H(\mathbf{S_2})$.

- Language Virtualization:

- Process Virtualization:

- System Virtualization:

Language Virtualization: Example w/ a JVM

Initial State ($\mathbf{S_1}$):		
Transition ($\mathbf{S_1} \Rightarrow \mathbf{S_2}$):		

System #1
`COPY r1 1`
`SHIFTL x 2`
`ADD x r1`
	System #2
	`COPY r1 x`
	`SHIFTL x`
	`SHIFTL x`
	`ADD x r1`
		System #3
		`COPY r1 x`
		`ADD r1 x`
		`ADD r1 x`
		`ADD r1 x`
		`ADD r1 x`

Final State ($\mathbf{S_2}$):		

System Virtualization: Containers
A commonly deployed form of visualization is _____.

- **As a developer of a Docker,** you build a `Dockerfile` that specifies the **snapshot** of the system you want to provide and then **build** that snapshot into a _____.

- Create a Dockerfile to specify how to build the image:

mp3/Dockerfile

```
1 FROM gcc:latest
2 COPY ./docker/entrypoint.sh /
3 RUN chmod +x entrypoint.sh
4 ENTRYPOINT ["/entrypoint.sh"]
```

To build it:

```
$ docker build --tag mp3-docker .
```

- As **a user of a container**, you specify the name of the docker image that you want to use to launch that image:

```
$ docker run -it --rm -v "pwd":/mp3  mp3-docker "make"
$ docker run --rm -it -p 27017:27017 mongo
```

System Virtualization: Hypervisor

...has this changed our industry??

Your CS 340 Virtual Machine:
The CS department has a "private cloud", containing cloud services for us to use!

- As part of being in CS 340, you have your very own VM!

- This machine is a "private cloud" solution to "Infrastructure as a Service" and is effectively identical to AWS EC2 or other compute services.