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

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

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

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