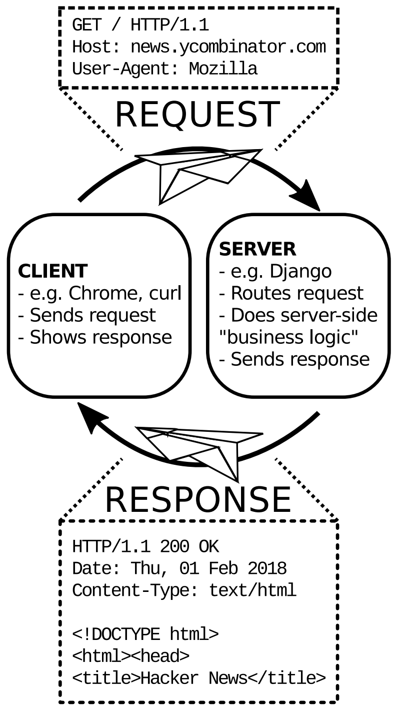
Requests & Responses



```
GET / HTTP/1.1
Host: news.ycombinator.com
User-Agent: Mozilla
```

```
HTTP/1.1 200 OK
Date: Thu, 01 Feb 2018
Content-Type: text/html

<!DOCTYPE html>
<html><head>
<title>Hacker News</title>
```

Python requests

```
import requests
url = ("https://api.openaq.org"
     "/v1/measurements"
     "?location=Oakland")
response = requests.get(url)
d = response.json()
print(d) # See response data
print(d["results"][0]["value"])
```

API Terminology

API The interface that software uses to communicate with other computers or software. **REST APIs** are the most popular, and use HTTP requests and responses to exchange data or perform actions.

API Key / Secret "Username" and "password" for APIs that need it.

HTTP Methods

GET Retrieve information

POST Create a new item

PUT Update an existing item

DELETE Delete an item

Terminology

Protocol Agreed-upon "routine" for computers talking to each other

IP address Internet Protocol address, a "phone number" for computers

Domain name E.g. `google.com`, a mnemonic that's turns into the IP address of a server

TCP The way computers "call each other up" and send data to each other over the internet (*transmission control protocol*)

HTTP A protocol where a *client* uses TCP to connect and send a *request*, containing a particular *method* and *path* to a *server*, which in turn replies with a *response*.

Headers & body Requests and responses both are split into extra info (headers), and data (body).

Method The first text in the request, conventionally is one of 4 words in all caps. The backend gets to decide how to process different HTTP methods differently.

Advanced Python

Try / except exceptions Handle or ignore errors.

```
a = [1, 2]
try:
    lucky_13 = a[13]
except Exception as e:
    print("It broke:", e)
```

List comprehension Like a for-loop, but also creates a new list

```
names = ["John", "Paul", "G"]
long_names = [
    n.lower() for n in names
    if len(n) > 2
]   # = ["john", "paul"]
```

Unpacking assignment Can assign to two or more at once, in both loops and elsewhere

```
pairs = [(10, 5), (8, 100)]
for left, right in pairs:
    print(left * right)
x, y = [35, 15]
```

Variable length arguments "Catch-alls" for positional or named args.

```
def do_all(*args, **kwargs):
    print(args, kwargs)
    return sum(args)
do_all(3, 5, b=3)
```

Django routing

```
from django.urls import path
from django.http import HttpResponse

def hi_world(request):
    # Simplest view: Just respond
    # with string, no templating
    return HttpResponse("Hi world!")

def about_me(request):
    # Context dictionary: Variables
    # that go into the template
    ctx = {
        "name": "Ash Ketchum",
    }
    # render: Do templating with
    # given context variables
    return render(
        request, "about.html", ctx)

# urlpatterns connects URLs to
# view functions.
urlpatterns = [
    path("hello-world/", hi_world),
    path("about-me/", about_me),
]
```

Heroku

```
heroku create # Create a new app
git remote -v # Check git remotes

# Launch site via git push, do
# after every change:
git push heroku master

heroku open # View your web app
heroku logs # Debug (view output)

# Lesser used:
heroku local # Test Procfile
heroku config # Inspect environment
heroku ps:exec # Start Heroku bash
```

Mini-Django Boilerplate

minidjango Our boilerplate to get you started on small web apps .

Procfile Tells Heroku how to start.

Pipfile Contains PyPI dependencies.

static/ CSS, images, and JS go here.

templates/ HTML templates go here.

urls.py Contains "routing": Matches paths (eg urls) to *view functions.*

views.py Contains “view functions”, which do templating and send back responses based on requests.