

# API 101 (Python3)

DAT-A SMASH! #01:  
"API with Python Fundamental"

SIVAPHONG NIYOMPHANICH

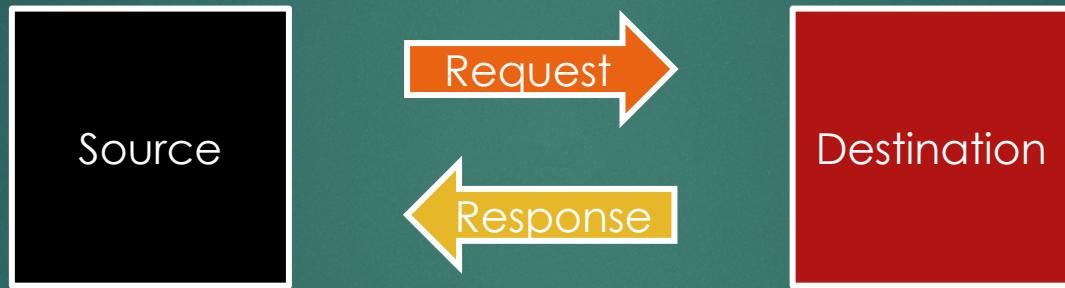
# Agenda

Pongier  
8 May 2021

- ▶ API Overview
- ▶ Basic Network – IP Address (Public/Private), Protocol/Port, Payload
- ▶ Server & Infrastructure – Hardware, OS, (Hypervisor/Container) Middleware, Application
- ▶ Cloud – IaaS, PaaS, SaaS
- ▶ Web API
- ▶ HTTP – GET, POST, PUT, DELETE, PATCH
- ▶ Restful, JSON
- ▶ API with Python
- ▶ Flask

# API Overview

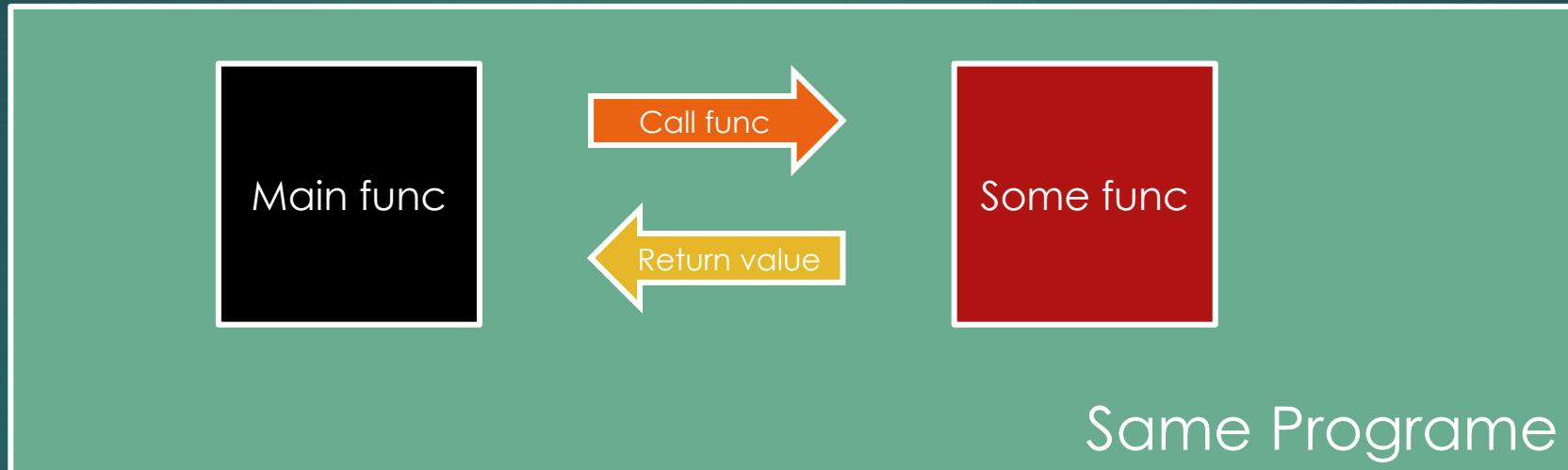
- ▶ Connect between two or more object (and exchange data)



# API Overview

Pongier  
8 May 2021

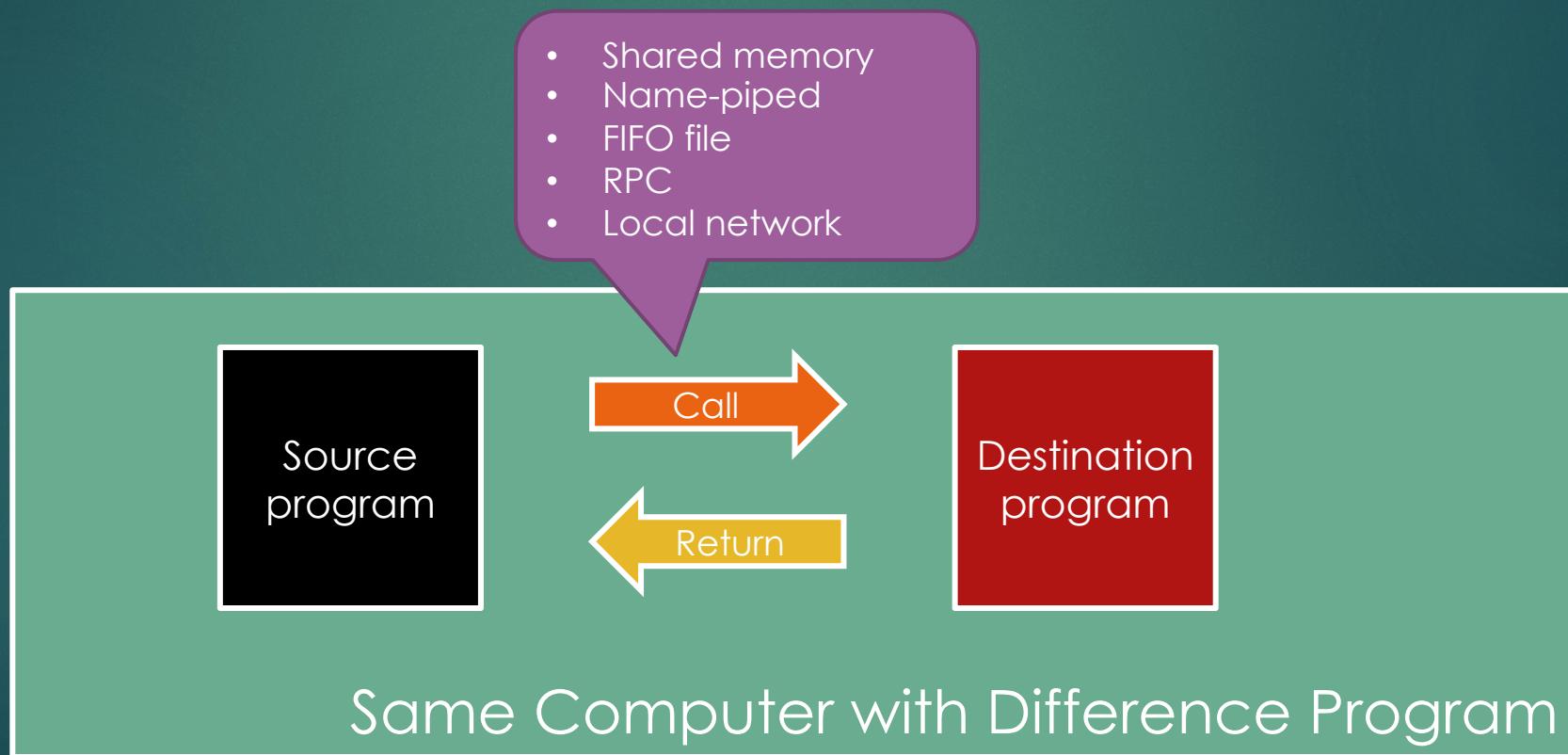
- ▶ Connect between two or more object (and exchange data)



# API Overview

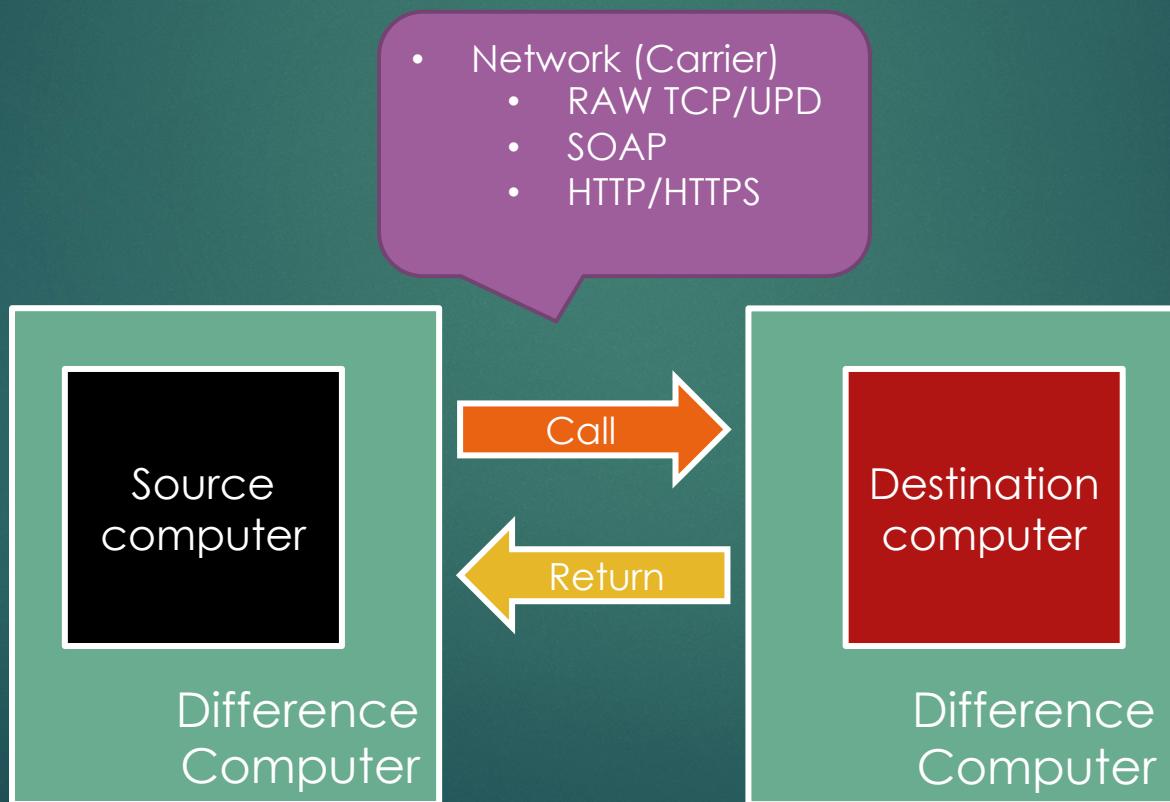
Pongier  
8 May 2021

- ▶ Connect between two or more object (and exchange data)



# API Overview

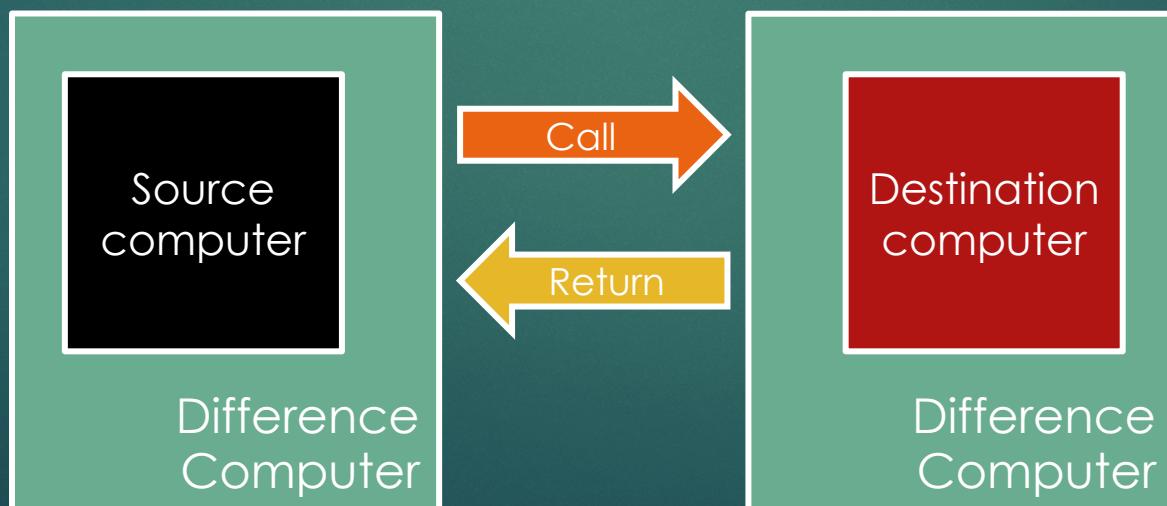
- ▶ Connect between two or more object (and exchange data)



# Computer Network Concept

Pongier  
8 May 2021

- ▶ Connect between two or more object (and exchange data)
- ▶ Concept
  - ▶ IP Address
  - ▶ Protocol/Port
  - ▶ Protocol/ Data Payload



# Computer Network Concept

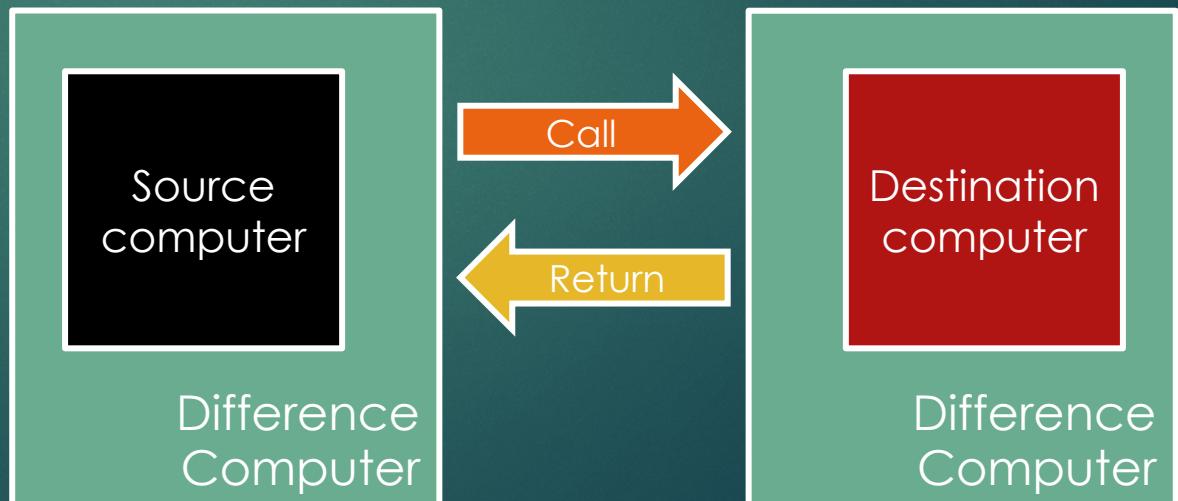
- ▶ Connect between two or more object (and exchange data)

- ▶ Concept

- ▶ IP Address
- ▶ Protocol/Port
- ▶ Protocol/ Data Payload

- Requester
  - Source IP Addr
  - Source Port
  - Request Payload

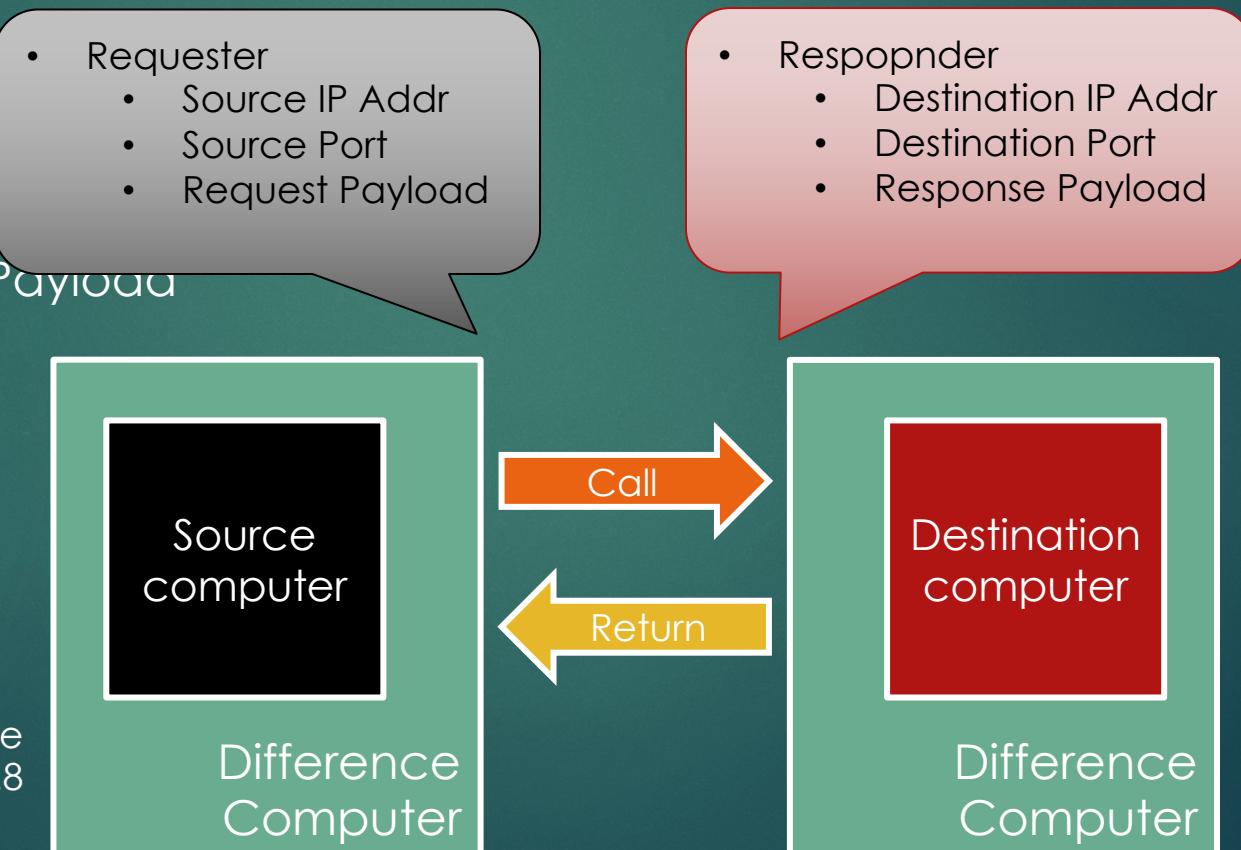
- Resopnder
  - Destination IP Addr
  - Destination Port
  - Response Payload



# Computer Network Concept

Pongier  
8 May 2021

- ▶ Connect between two or more object (and exchange data)
- ▶ Concept
  - ▶ IP Address
  - ▶ Protocol/Port
  - ▶ Protocol/Data Payload

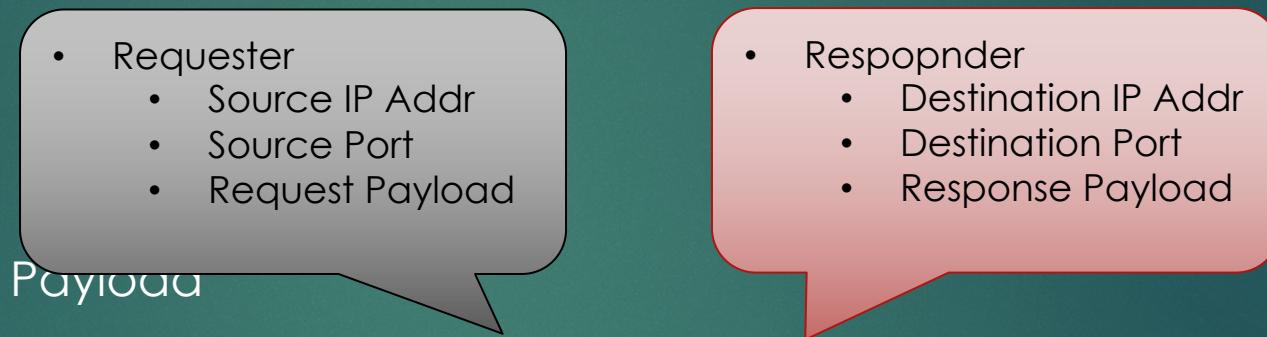


Open Chrome  
URL: 8.8.8.8

Dst IP: 8.8.8.8  
Dst Port: HTTPS/443  
Res Payload: HTTP Code 202

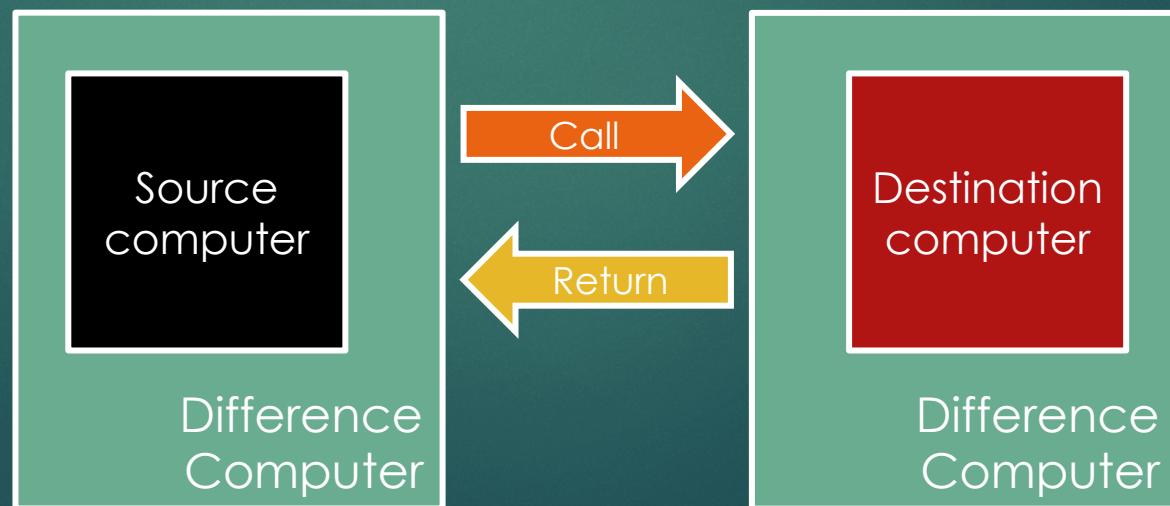
# Computer Network Concept

- ▶ Connect between two or more object (and exchange data)
- ▶ Concept
  - ▶ IP Address
  - ▶ Protocol/Port
  - ▶ Protocol/Data Payload



DNS  
(Domain Name Service)  
Convert "Name" to "IP Addr"  
e.g. www.google.com = 8.8.8.8

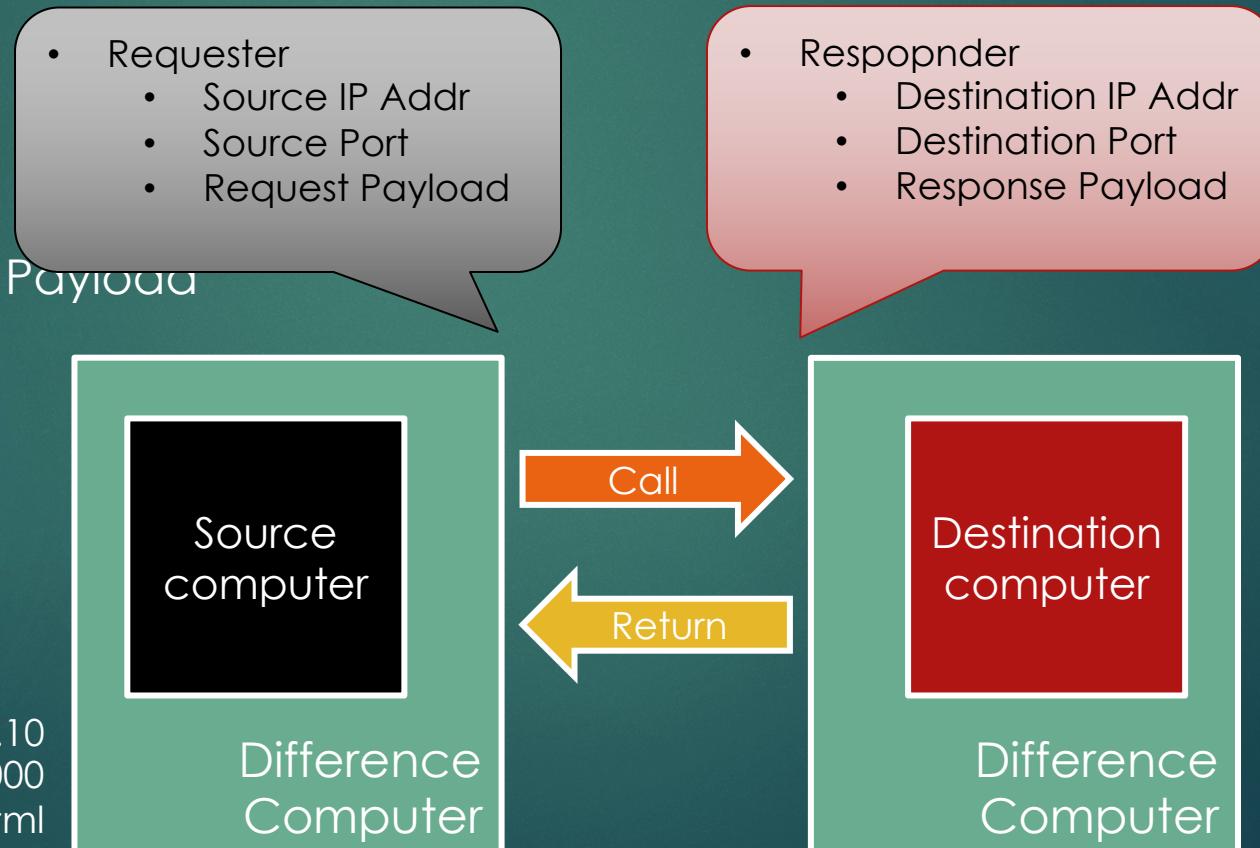
Open Chrome  
URL: [www.google.com](http://www.google.com)



Dst IP: 8.8.8.8  
Dst Port: HTTPS/443  
Res Payload: HTTP Code 202

# Computer Network Concept

- ▶ Connect between two or more object (and exchange data)
- ▶ Concept
  - ▶ IP Address
  - ▶ Protocol/Port
  - ▶ Protocol/Data Payload



# Computer Network Concept

12

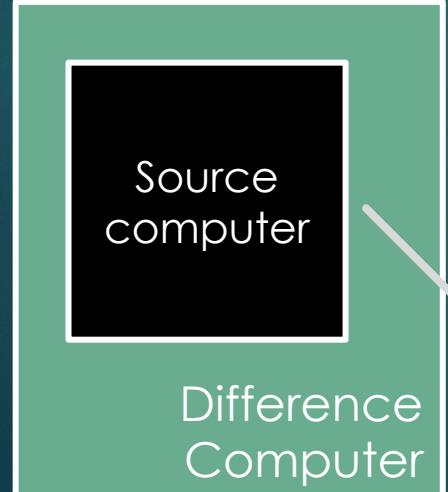
Pongier  
8 May 2021

## ► In Real world

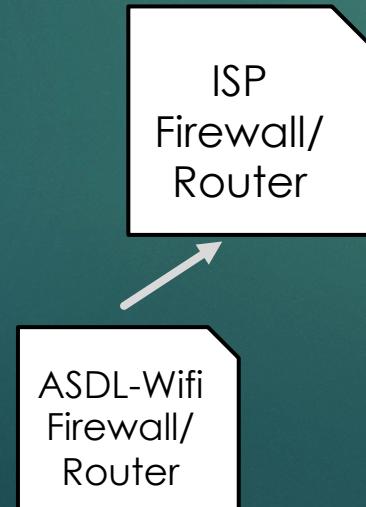
- Private IP – See only in our home, office, organization
- Public IP – See all in the world

• <https://www.whatismyip.com/>

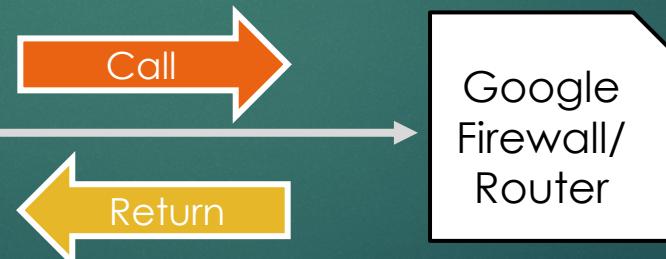
Pri Src IP: 192.168.1.10  
Pri Src Port: HTTP/60000  
**Req Payload: GET /index.html**



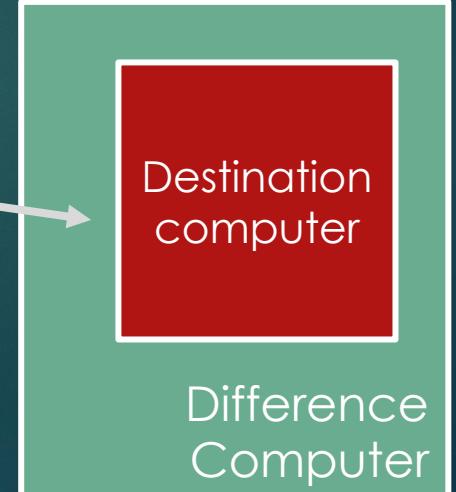
Pub Src IP: 203.204.205.99  
Pub Src Port: HTTP/50000



Pub Dst IP: 8.8.8.8  
Pub Dst Port: HTTPS/443



Pri Dst IP: 10.1.2.9  
Pri Dst Port: HTTPS/443  
**Res Payload: HTTP Code 202**



# Computer Network Concept

13

LINE Server  
P. Böger  
8 May 2021

Difference Computer

Pub Dst IP: 1.1.1.1  
Pub Dst Port: HTTPS/443

My computer  
Difference Computer

ISP Firewall/  
Router  
ASDL-Wifi Firewall/  
Router



Cloud/  
Heroku  
Difference Computer

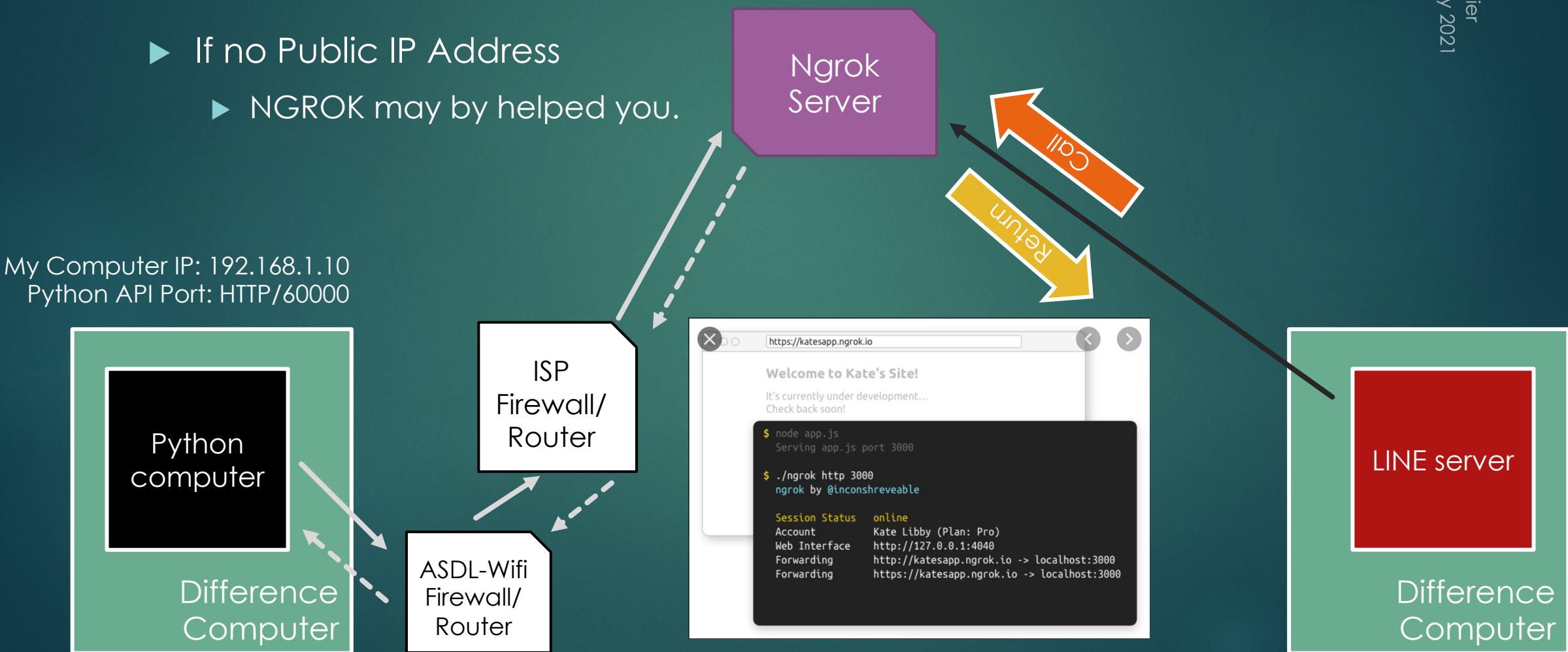
# Computer Network Concept

14

Bongier  
8 May 2021

- ▶ If no Public IP Address
    - ▶ NGROK may by helped you.

Ngrok Pub Src IP: 203.204.205.99 (Assign with name)  
Ngrok Pub Src Port: HTTP/10000 (Assign per source computer)



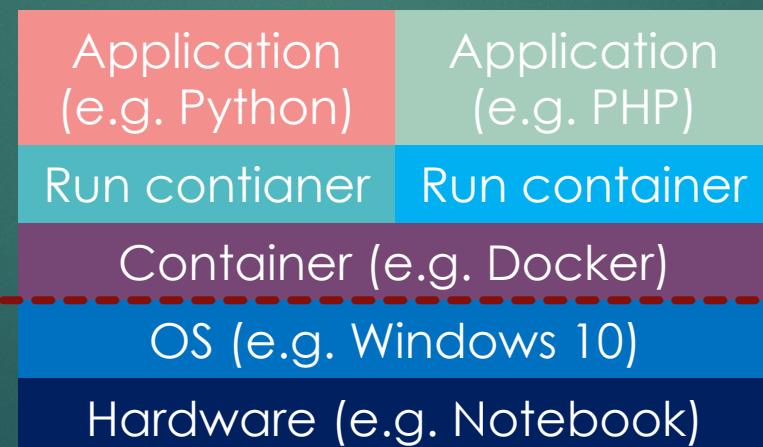
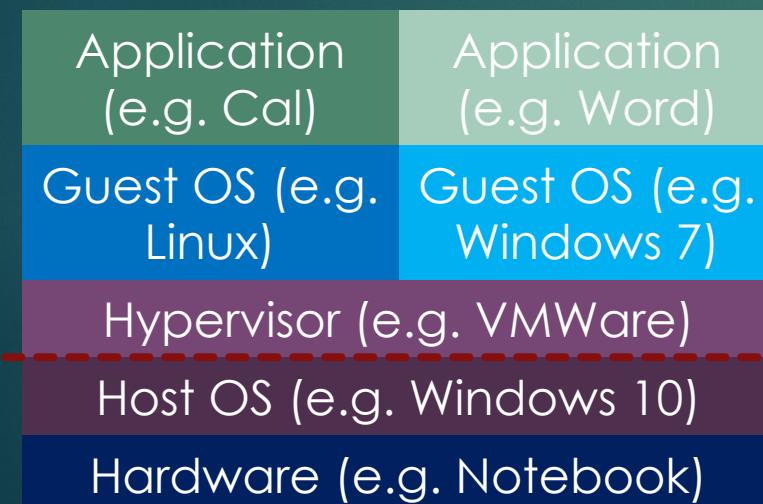
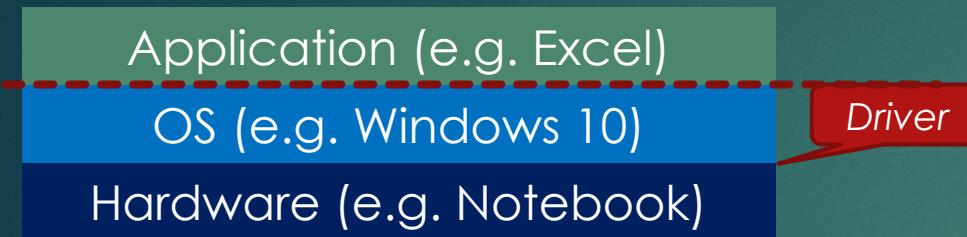
# Computer Network Concept

15

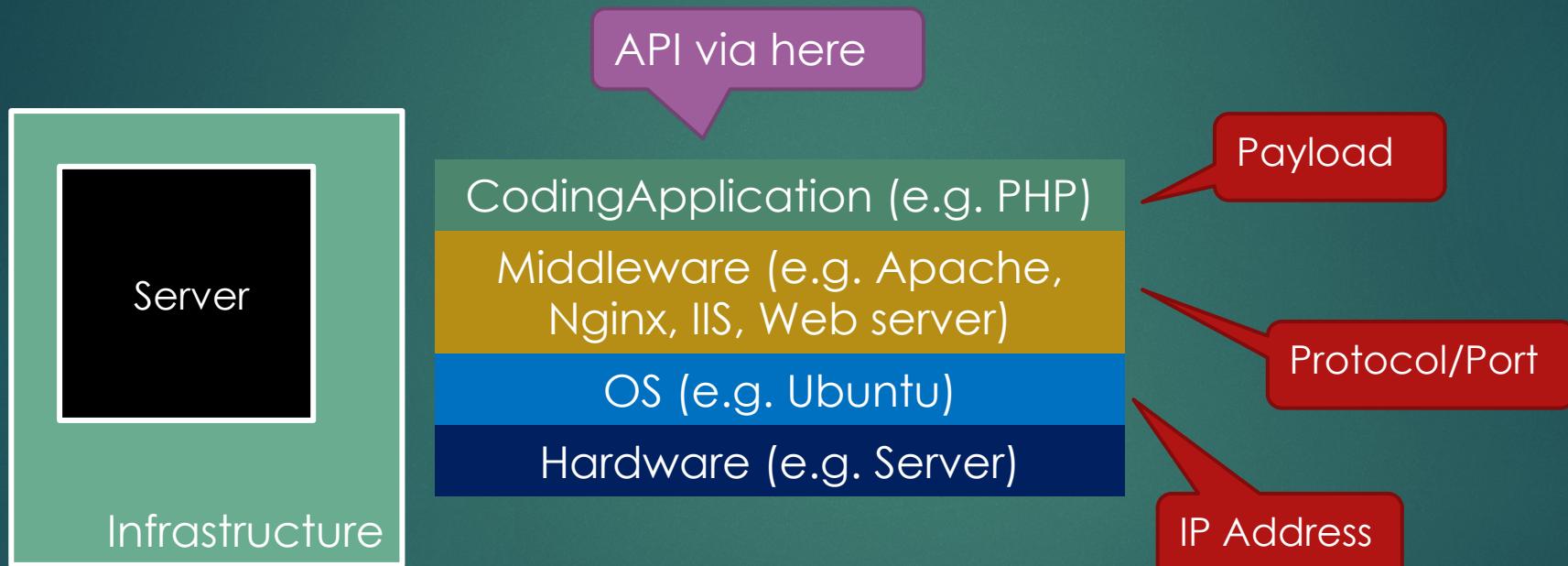
Pongier  
8 May 2021

- ▶ Connect between two or more object (and exchange data)
- ▶ Concept
  - ▶ IP Address
  - ▶ Protocol/Port (carrier)
    - ▶ HTTP (TCP/80), HTTPS (TCP/443), SSH (TCP/22), RDP (TCP/3389), DNS (UDP/53)
  - ▶ Protocol/ Data Payload
    - ▶ HTTP Method
    - ▶ JSON
    - ▶ RAW
  - ▶ Protocol payload vs Data payload

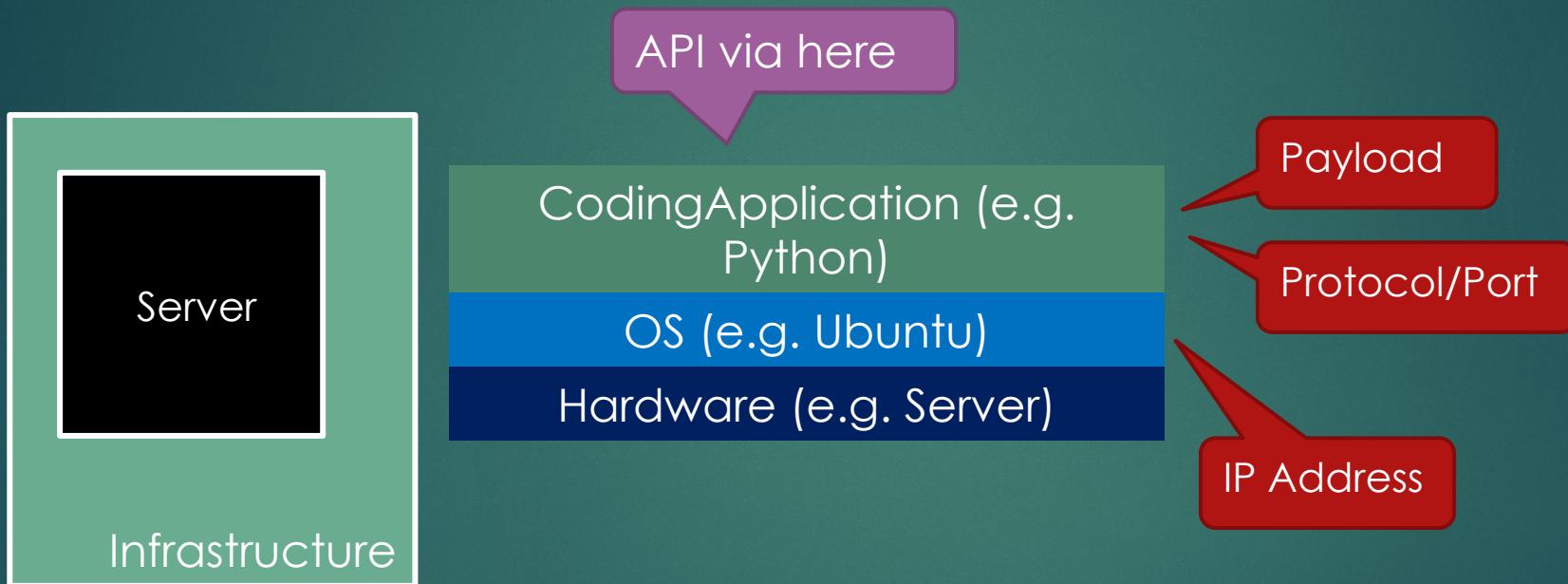
# Server & Infrastructure



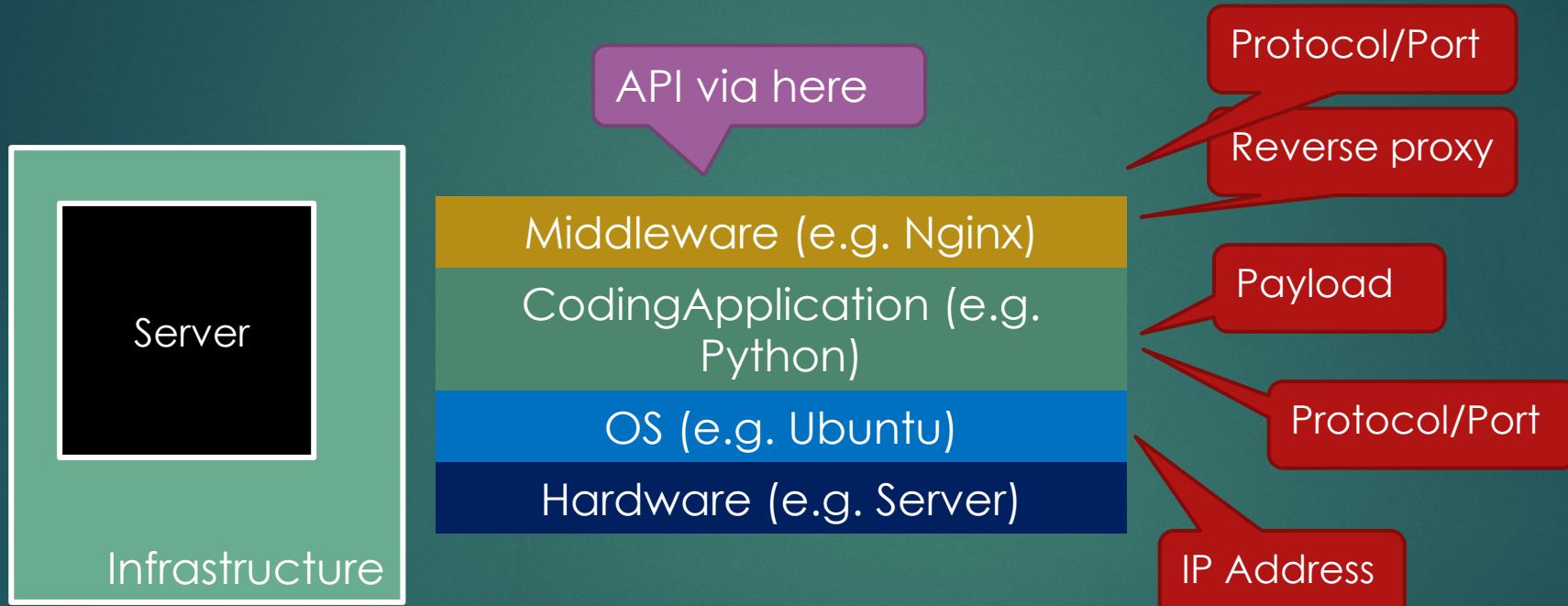
# Hierarchy Example I



# Hierarchy Example II



# Hierarchy Example III



# Cloud

- ▶ IaaS (Infrastructure As A Service)
  - ▶ Rent a server
- ▶ PaaS (Platform As A Service)
  - ▶ Deploy a python web API
- ▶ SaaS (Software As A Service)
  - ▶ Using a mail account
  - ▶ Using a Google drive

User managed

Provider managed

On premises

Application

Data

Runtime

Middleware

Operating system

Virtualization

Networking

Storage

Servers

IaaS

Application

Data

Runtime

Middleware

Operating system

Virtualization

Networking

Storage

Servers

PaaS

Application

Data

Runtime

Middleware

Operating system

Virtualization

Networking

Storage

Servers

SaaS

Application

Data

Runtime

Middleware

Operating system

Virtualization

Networking

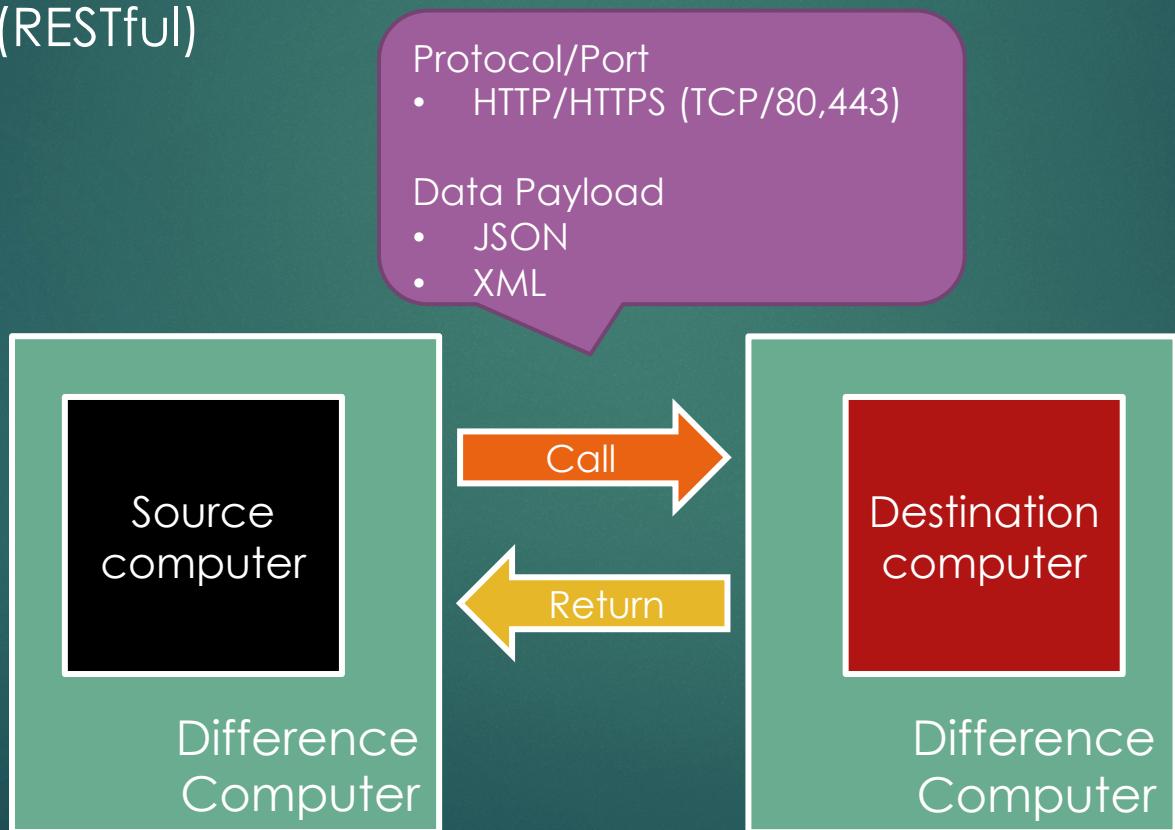
Storage

Servers

Paragiter  
8 May 2021

Hardware

- ▶ Exchange data between two or more objects by using HTTP protocol (RESTful)



# Understanding HTTP Protocol

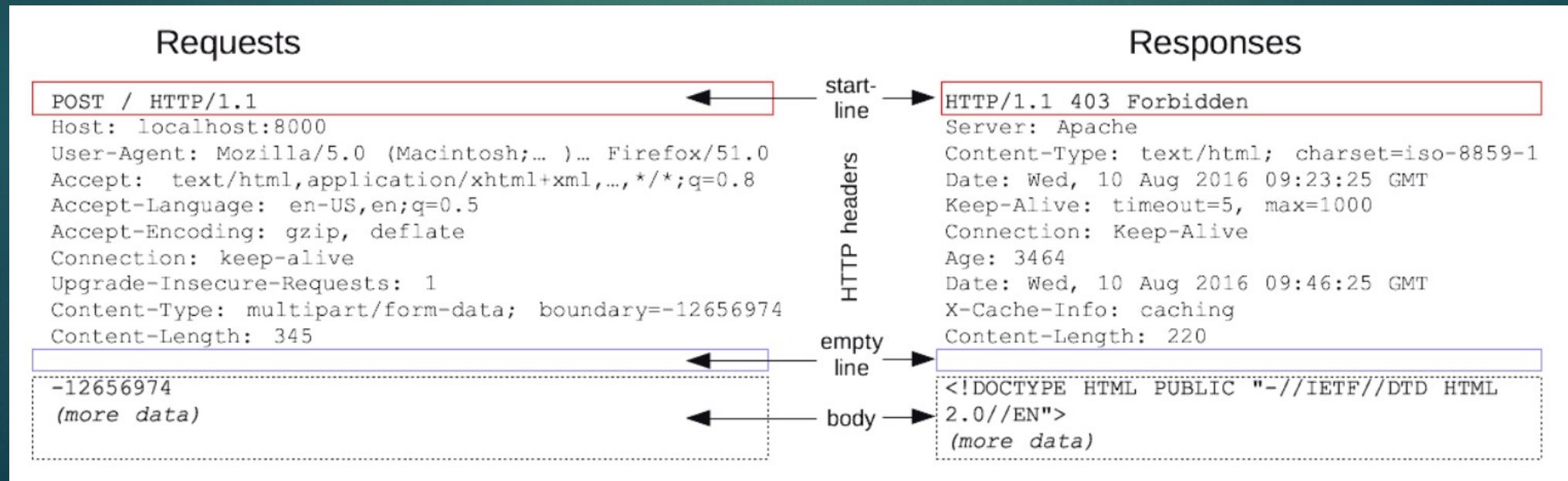
- ▶ HTTP = Hyper Text Transfer Protocol
- ▶ HTTPS = Secure using SSL/TLS
- ▶ HTTP Request Method
  - ▶ GET
  - ▶ POST
  - ▶ PUT
  - ▶ DELETE
  - ▶ PATCH
- ▶ HTTP Response Code
  - ▶ 1XX
  - ▶ 2XX
  - ▶ 3XX
  - ▶ 4XX
  - ▶ 5XX

# Understanding HTTP Protocol

24

Pongier  
8 May 2021

- ▶ Payload (HTTP Message)
  - ▶ HTTP Header
  - ▶ HTTP Body



# Understanding HTTP Protocol

Pongier  
8 May 2021

## HTTP URL Anatomy

1

2

3

4

5

6

7

8

<https://www.example.com:3000/path/resource?id=123#section-id>

### Key

- ① Scheme - defines how the resource will be obtained.
- ② Subdomain - www is most common but not required.
- ③ Domain - unique value within its top-level domain.
- ④ Top-level Domain - hundreds of options now exist.
- ⑤ Port - if omitted HTTP will connect on port 80, HTTPS on 443.
- ⑥ Path - specify and perhaps find requested resource.
- ⑦ Query String - data passed to server-side software, if present.
- ⑧ Fragment Identifier - a specific place within an HTML document.

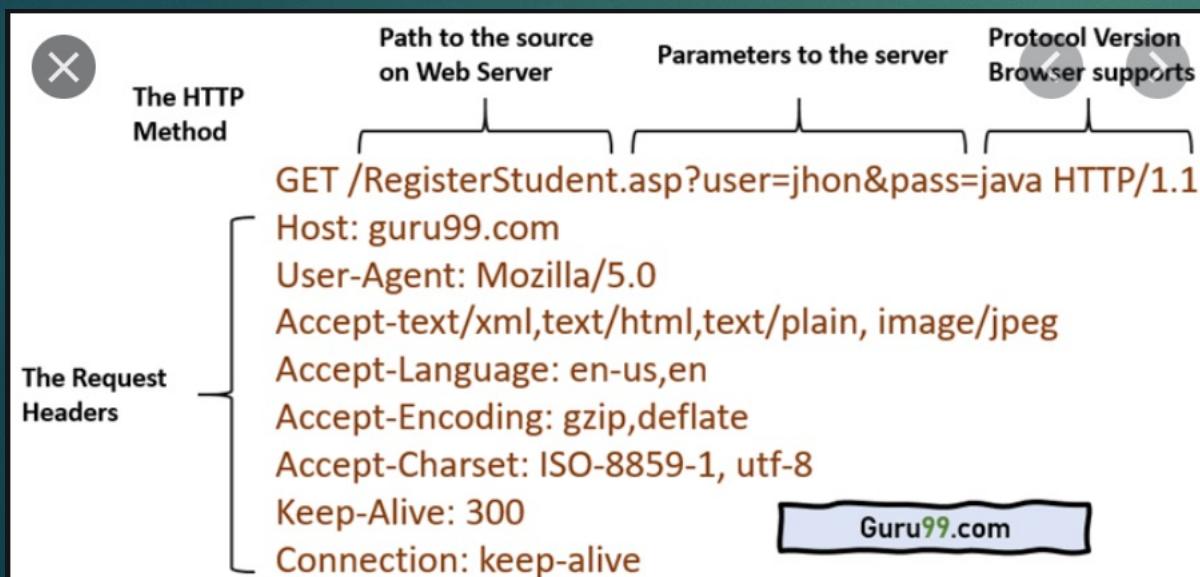
# Understanding HTTP Protocol

26

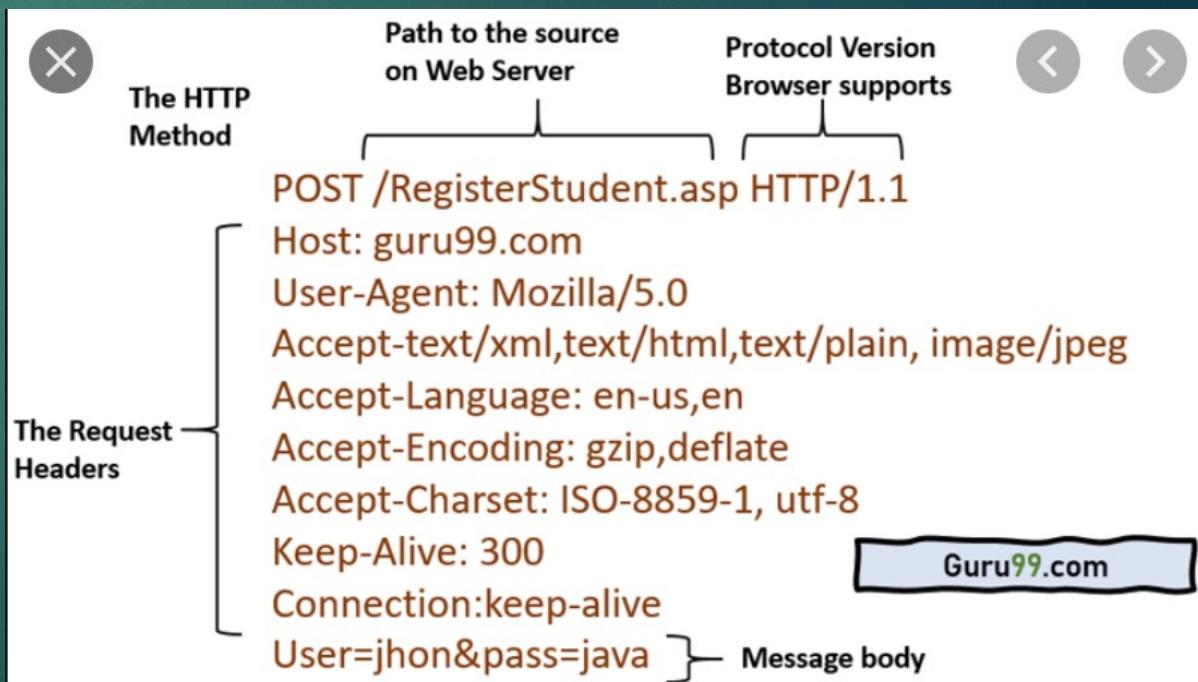
Pongier  
8 May 2021

## ► GET vs POST method

http://guru99.com/RegisterStudent.asp?user=jhon&pass=java



http://guru99.com/RegisterStudent.asp



# First calling Web API

Pongier  
8 May 2021

- ▶ Most helpful tool
- ▶ Postman
  - ▶ <https://www.postman.com>
- ▶ API
  - ▶ <https://api.agify.io/?name=bella>
- ▶ Keyword
  - ▶ URL
  - ▶ HTTP Method
  - ▶ Parameter/Payload (Both Request and Response)
  - ▶ Is need subscriber (e.g. token, key)

# Postman

- ▶ Create a "Collection" and rename to "API101"
- ▶ Create a "Request" with following setting
  - ▶ Method: GET
  - ▶ URL: <https://api.agify.io/?name=bella>
- ▶ Send the request
- ▶ Observed the response
- ▶ Try the other one API
  - ▶ <https://api.exchangerate-api.com/v4/latest/THB>

The screenshot shows the Postman application interface. On the left, there's a sidebar with icons for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main area is titled 'Scratch Pad' and shows a collection named 'API101'. A request card is visible with the method 'GET' and the URL 'https://api.agify.io/?name=bella'. Below the request card, under the 'Query Params' section, there's a table with a single row: 'name' with value 'bella'. In the bottom right corner of the main area, there's a preview of the response body, which is a JSON object:

```
1 "name": "bella",
2 "age": 34,
3 "count": 40138
```

# More Advanced API

▶ <https://openweathermap.org/api/hourly-forecast>

▶ <https://jsonplaceholder.typicode.com/guide/>

<https://jsonplaceholder.typicode.com/posts>

## Creating a resource

```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  body: JSON.stringify({
    title: 'foo',
    body: 'bar',
    userId: 1,
  }),
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  },
})
  .then((response) => response.json())
  .then((json) => console.log(json));
```

```
{
  "title": "foo",
  "body": "bar",
  "userId": 1
}
```

# Advanced API

The screenshot shows the Postman interface. At the top, there are tabs for Params, Authorization, Headers (10), Body (green dot), Pre-request Script, Tests, and Settings. The Headers tab is active, showing a table with one row: Content-Type with value application/json; charset=UTF-8. Below the Headers tab, the Body tab is also active, showing a JSON payload:

```
1 {  
2   "title": "foo",  
3   "body": "bar",  
4   "userId": 1  
5 }  
6
```

At the bottom right of the interface, there is a watermark: "gier 2021".

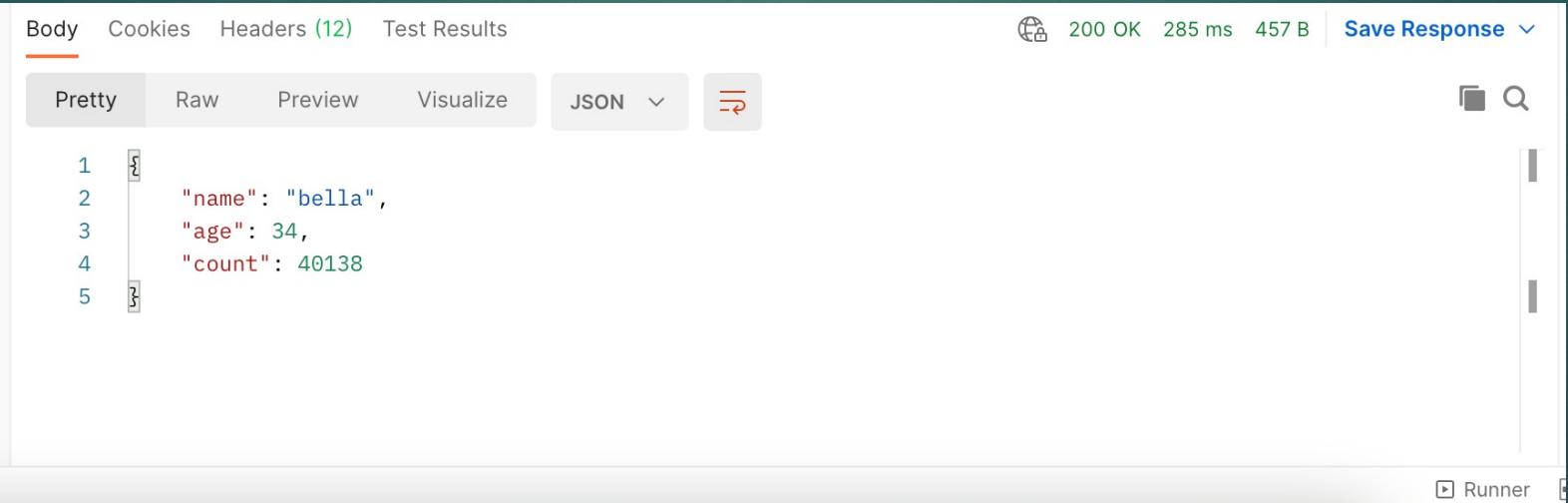
The screenshot shows the Postman interface. At the top, there are tabs for POST, https://jsonplaceholder.typicode.com/posts, Send (blue button), and a dropdown. Below these are tabs for Params, Authorization, Headers (10) (which is active and highlighted in orange), Body (green dot), Pre-request Script, Tests, and Settings. The Body tab is active, showing a JSON payload:

```
1 {  
2   "title": "foo",  
3   "body": "bar",  
4   "userId": 1  
5 }  
6
```

Below the body editor, there are tabs for Body (active), Cookies (1), Headers (26), and Test Results. To the right, there is a status bar showing 201 Created, 579 ms, 1.23 KB, and a Save Response button. At the bottom, there are tabs for Pretty (active), Raw, Preview, Visualize, and JSON (with a dropdown arrow). The JSON preview area shows the same JSON payload as the editor.

# Observed the Responding

- ▶ The responding is JSON format



The screenshot shows a JSON response in a browser's developer tools or a tool like Postman. The response is a 200 OK status with 285 ms latency and 457 B size. The body of the response is displayed in Pretty JSON format:

```
1 {  
2   "name": "bella",  
3   "age": 34,  
4   "count": 40138  
5 }
```

# Restful & JSON

- ▶ RESTful
  - ▶ <https://iamgique.medium.com/restful-api-ກັນ-rest-api-ຕ່າງກັນນະຫຼຸດ-2c70c42990e3>
- ▶ RESTful web service
  - ▶ Carrier = HTTP Protocol
  - ▶ Data Payload = JSON (or XML or HTML or other format)
- ▶ Other word
  - ▶ SOAP, REST, ...

# Restful & JSON

- ▶ HTTP Request Method for RESTful service
  - ▶ <https://www.restapitutorial.com/lessons/httpmethods.html>

HTTP Verb	CRUD	Entire Collection (e.g. /customers)	Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

# Restful & JSON

34

Pongier  
8 May 202

- ▶ JSON (Java Script Object Notation)
- ▶ Key : Value
- ▶ "key": "value"
- ▶ [] is array
- ▶ {}



The screenshot shows a JSON editor window titled "personal.json". The JSON code defines a schema and a personnel list. The schema is defined in "personal-schema.json". The personnel list contains a single person named "Big.Boss" with the following details:

```
1 {  
2     "$schema": "personal-schema.json",  
3     "personnel": [  
4         "person": [  
5             {  
6                 "id": "Big.Boss",  
7                 "name": {  
8                     "family": "Boss",  
9                     "given": "Big"  
10                },  
11                "email": "chief@oxygenxml.com",  
12                "link": {  
13                    "subordinates": [  
14                        "one.worker",  
15                        "two.worker",  
16                        "three.worker",  
17                        "four.worker",  
18                        "five.worker"  
19                    ]  
20                }  
21            }  
22        ]  
23    ]  
24}
```

# API with Python

- ▶ Create your RESTful API with Python
- ▶ Useful framework
  - ▶ Django
  - ▶ Flask
  - ▶ FastAPI
- ▶ Did you remember
  - ▶ Src/Dst IP Addr
  - ▶ Src/Dst Protocol,Port
  - ▶ Data Payload

# Flask Overview

- ▶ <https://palletsprojects.com/p/flask/>
- ▶ <https://flask.palletsprojects.com/en/1.1.x/quickstart/>

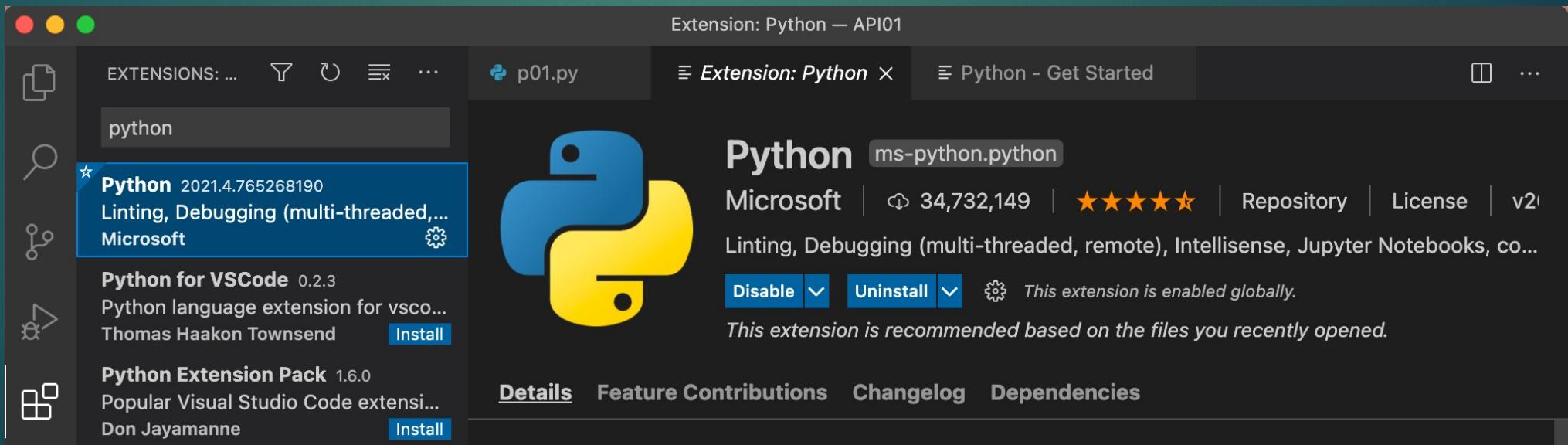
```
# save this as app.py
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

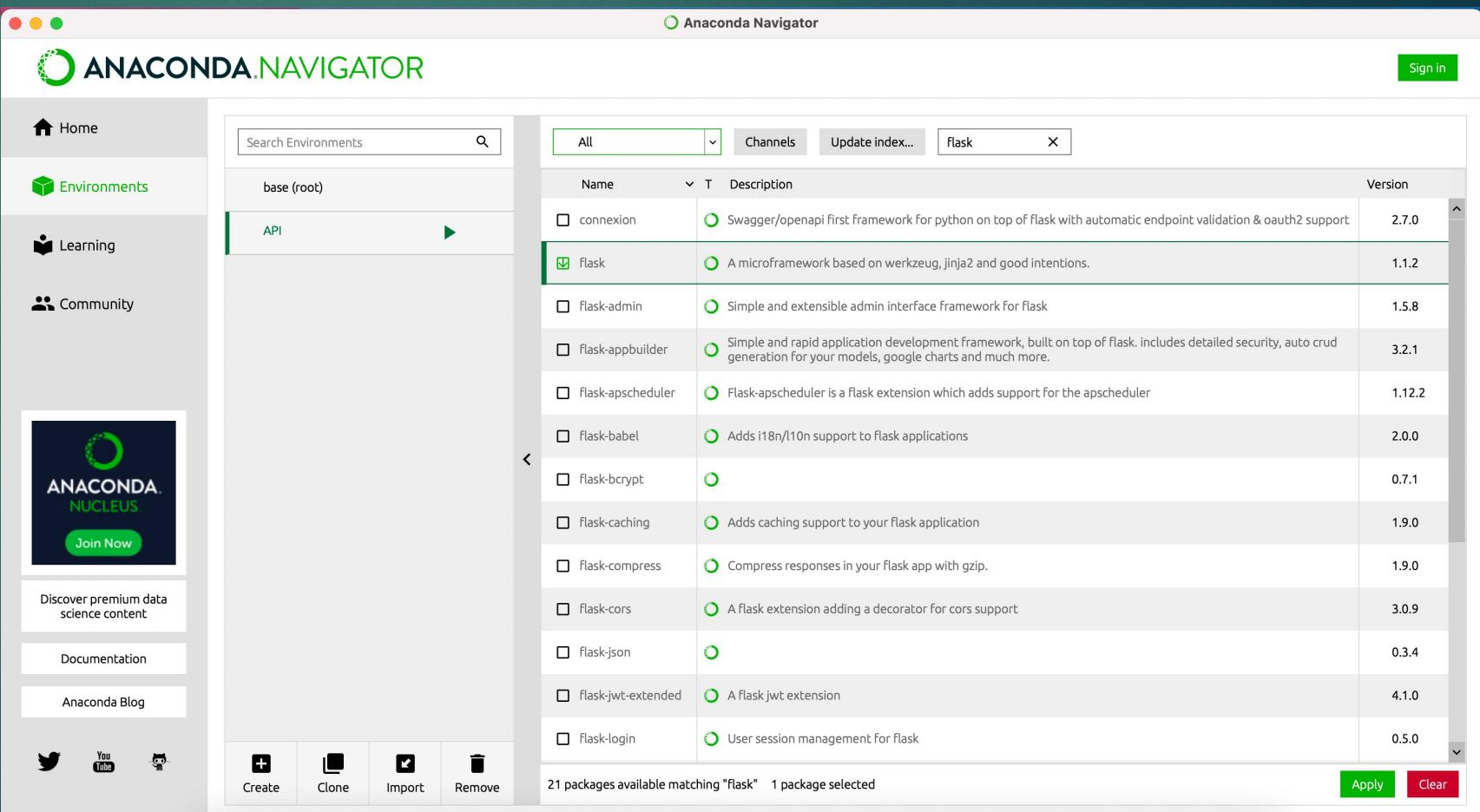
# Prepared tools

- ▶ Python & Env (example using Anaconda)
  - ▶ <https://www.anaconda.com>
- ▶ VS Code & Python Extension
  - ▶ <https://code.visualstudio.com>



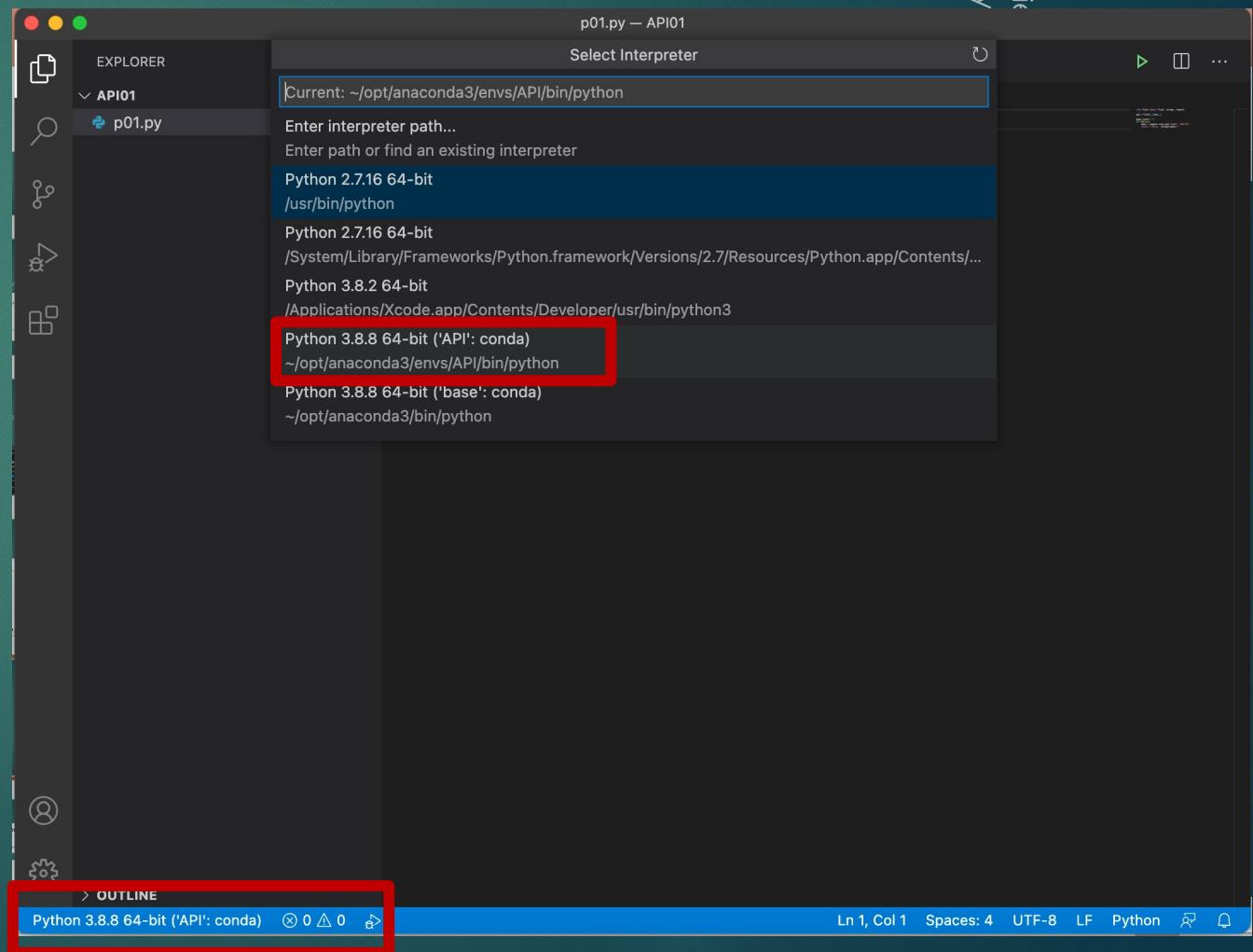
# Prepared tools

► pip install flask



# VS Code

- ▶ Create new file named “app.py”
- ▶ Select Python (Anaconda API)



# First API with Python Flask

```
from flask import Flask, escape, request

app = Flask(__name__)

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

Pongier  
8 May 2021

- ▶ Run -> Start Debugging -> Flask



The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the output of a Flask application run via the debugger. It includes a warning about using a development server in production, the current environment (development), debug mode status, the running URL (http://127.0.0.1:5000), and a log message indicating it's serving from app.py. The line 'Running on http://127.0.0.1:5000/' is highlighted with a red box.

```
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
^C%
(API) pongier@pongiermbp API01 % cd /Users/pongier/Desktop/Python/API01 ; /usr/bin/env /Users/pongier/opt/anaconda3/envs/API/bin/python /Users/pongier/.vscode/extensions/ms-python.python-2021.4.765268190/pythonFiles/lib/python/debugpy/launcher 52987 -- -m flask run --no-debugger
* Serving Flask app "app.py" (lazy loading)
* Environment: development
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
```

Test with Postman

The screenshot shows the Postman application interface. At the top, there is a header bar with a dropdown menu set to "GET", a URL input field containing "http://localhost:5000", and a blue "Send" button. Below the header, a navigation bar includes tabs for "Params" (which is selected and highlighted in orange), "Authorization", "Headers (6)", "Body", "Pre-request Script", "Tests", "Settings", and "Cookies". A "Query Params" section is present, containing a table with one row. The table has columns for "KEY", "VALUE", "DESCRIPTION", and "Bulk Edit". The first row contains "Key" in the KEY column and "Value" in the VALUE column. In the DESCRIPTION column, the word "Description" is partially visible. At the bottom of the interface, there is a footer bar with tabs for "Body", "Cookies", "Headers (4)", and "Test Results" (the last one is also highlighted in orange). To the right of these tabs, there is a status indicator showing "200 OK", "15 ms", and "163 B", followed by a "Save Response" button. Below the footer, there are buttons for "Pretty", "Raw", "Preview", "Visualize", and "HTML" (with a dropdown arrow), and a search icon. The main content area displays the response body, which is a single item labeled "1 Index Page".

# Flask Route

Pongier  
8 May 2021

## HTTP URL Anatomy

1    2    3    4    5    6    7    8  
`https://www.example.com:3000/path/resource?id=123#section-id`

```
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
    return 'How are you'
```

Test with Postman

The screenshot shows the Postman application interface. At the top, there is a header bar with a dropdown menu, a search bar containing "http://localhost:5000/hello", and a "Send" button. Below the header, the main interface has tabs for "Params", "Authorization", "Headers (6)", "Body", "Pre-request Script", "Tests", "Settings", and "Cookies". The "Params" tab is currently selected. A table titled "Query Params" is displayed, with columns for KEY, VALUE, DESCRIPTION, and Bulk Edit. One row is present with the key "Key" and value "Value", and the description "Description". At the bottom of the interface, there are tabs for "Body", "Cookies", "Headers (4)", and "Test Results", with "Body" being the active tab. To the right of these tabs, there is a status bar showing "200 OK", "7 ms", "164 B", and a "Save Response" button. Below the status bar, there are buttons for "Pretty", "Raw", "Preview", "Visualize", and "HTML" (with a dropdown arrow), along with a search icon. The main body area displays the response content: "1 How are you".

# Flask with parameters

Security purpose

```
from markupsafe import escape

@app.route('/user/<username>')
def show_user_profile(username):
    # http://YOUR_SERVER/user/abcd
    return 'User %s' % escape(username)

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # http://YOUR_SERVER/post/9
    return 'Post %d' % post_id

@app.route('/user2')
def login():
    # http://YOUR_SERVER/user2?uname=dcba
    var1 = request.args.get('uname')
    return 'User2 %s' % escape(var1)
```

Test with Postman

GET http://localhost:5000/user2?uname=dcba Send

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	ooo	Bulk Edit
<input checked="" type="checkbox"/>	uname	dcba			
	Key	Value	Description		

Body Cookies Headers (4) Test Results

200 OK 9 ms 163 B Save Response ▾

Pretty Raw Preview Visualize HTML ▾

1 User2 dcba

# Flask HTTP Method

```
@app.route('/user3', methods=['POST'])
def login3():
    var1 = request.form['uname']
    return 'User3 %s' % escape(var1)
```

Test with Postman

POST http://localhost:5000/user3 Send

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	uname	MyName			
	Key	Value	Description		

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize HTML ▾

1 User3 MyName

200 OK 7 ms 165 B Save Response ▾

Filter Search

# Flask with JSON output

```
from flask import Flask, escape, request, jsonify

@app.route('/yourjson')
def ojson():
    person = {"name": "Alice", "birth-year": 1986}
    return jsonify(person)
```

# Test with Postman

The screenshot shows the Postman application interface. At the top, there is a header with 'GET' and the URL 'http://localhost:5000/yourjson'. To the right of the URL is a 'Send' button. Below the header, there are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is currently selected, indicated by an orange underline. Under the 'Body' tab, there is a dropdown menu with options: 'none', 'form-data', 'x-www-form-urlencoded' (which is selected, indicated by an orange dot), 'raw', 'binary', and 'GraphQL'. Below this is a table with columns 'KEY', 'VALUE', 'DESCRIPTION', and 'Bulk Edit'. There is one row in the table with 'Key' in the KEY column and 'Value' in the VALUE column. At the bottom of the table, there is a 'Description' field. In the footer area, there are tabs for 'Body' (selected), 'Cookies', 'Headers (4)', and 'Test Results'. To the right of these tabs, there is a status bar showing '200 OK', '13 ms', '190 B', and a 'Save Response' button. Below the status bar, there is a preview section with tabs for 'Pretty' (selected), 'Raw', 'Preview', 'Visualize', and 'JSON' (selected, indicated by an orange underline). The JSON preview shows the following code:

```
1 {  
2   "birth-year": 1986,  
3   "name": "Alice"  
4 }
```

# Flask call other API

50

Pongier  
8 May 2021

- ▶ pip install requests

The screenshot shows a software interface for managing Python packages. On the left, there's a sidebar with a 'base (root)' environment and an 'API' section. The main area is a search interface with a 'Search Environments' input field and a magnifying glass icon. Above the search bar are buttons for 'All' (highlighted), 'Channels', 'Update index...', and a search term 'requests' with a clear button. The results table has columns for 'Name', 'Description', and 'Version'. There are four rows:

Name	Description	Version
aws-requests-auth	Aws signature version 4 signing process for the python requests module	0.4.3
cachecontrol	Httplib2 caching algorithms for use with requests	0.12.6
pyramid_tm	Allows pyramid requests to join the active transaction	2.2.1
<input checked="" type="checkbox"/> requests	Requests is an elegant and simple http library for python, built with ❤.	2.25.1

# Flask call other API

Pongier  
8 May 2021

```
import requests

@app.route('/callext/<username>')
def callext(username):
# call to https://api.agify.io/?name=bella
r = requests.get('https://api.agify.io/?name=' + escape(username))
return r.text
```

- ▶ More for call other API with POST method
  - ▶ [https://www.w3schools.com/python/ref\\_requests\\_post.asp](https://www.w3schools.com/python/ref_requests_post.asp)

The screenshot shows the Postman application interface. At the top, there is a header bar with the method "GET", the URL "http://localhost:5000/callext/kitkat", a "Send" button, and a dropdown arrow. Below the header, there are tabs for "Params", "Authorization", "Headers (6)", "Body", "Pre-request Script", "Tests", "Settings", and "Cookies". The "Body" tab is selected and has a red underline. Underneath it, there are radio buttons for "none", "form-data", "x-www-form-urlencoded" (which is selected), "raw", "binary", and "GraphQL". A table below these buttons has columns for "KEY", "VALUE", "DESCRIPTION", and "Bulk Edit". The first row of the table has "Key" in the KEY column and "Value" in the VALUE column. In the DESCRIPTION column, there is a link labeled "Description". At the bottom of the interface, there are tabs for "Body", "Cookies", "Headers (4)", and "Test Results", with "Body" being the active tab. There are also buttons for "Pretty", "Raw", "Preview", "Visualize", and "HTML" (with a dropdown arrow). On the right side of the bottom panel, there are icons for "Save Response" (with a dropdown arrow), a clipboard, and a magnifying glass. The response body is displayed as a JSON object: 

```
1  [{"name": "kitkat", "age": 48, "count": 368}]
```

# Flask API with Prediction model

53

```
def predictmodel():
    TRAIN_SET_LIMIT = 1000
    TRAIN_SET_COUNT = 100

    TRAIN_INPUT = list()
    TRAIN_OUTPUT = list()
    for i in range(TRAIN_SET_COUNT):
        a = randint(0, TRAIN_SET_LIMIT)
        b = randint(0, TRAIN_SET_LIMIT)
        c = randint(0, TRAIN_SET_LIMIT)
        op = a + (2*b) + (3*c)
        TRAIN_INPUT.append([a, b, c])
        TRAIN_OUTPUT.append(op)

    predictor = LinearRegression(n_jobs=-1)
    predictor.fit(X=TRAIN_INPUT, y=TRAIN_OUTPUT)

    X_TEST = [[10, 20, 30]]
    outcome = predictor.predict(X=X_TEST)
    coefficients = predictor.coef_

    print('Outcome : {}\nCoefficients : {}'.format(outcome, coefficients))

    return 'Outcome : {}\nCoefficients : {}'.format(outcome, coefficients)
```

```
@app.route('/predict')
def callpred():
    |    return predictmodel()
```

Pongier  
8 May 2021

# Q&A

► A&Q

# Python API for Machine Learning

55

Pongier  
8 May 2021

- ▶ API 101 (08 May 2021)
  - ▶ Basic API with Python3 and Flask
- ▶ API 201 (Tentative 15 May 2021)
  - ▶ Advance API input
    - ▶ Image
  - ▶ Train model and save model
  - ▶ Using saved model and conduct API
  - ▶ Tip & Trick for API (Speed and Problem)
  - ▶ Cloud Platform example