JSS MAHAVIDYAPEETHA
JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU 60
Department of Electronics and Communication Engineering

Python Collaborative Work

Bharath Kumar SP · HA Yeshwanth · H Bharath Bhat · Hrushik Raj S

Guidance: HS Kavitha Maam

Contents

HTTP, World’s Simplest Browser

- HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the World Wide Web. It is an application layer protocol designed to transfer hypertext requests and information between clients (usually web browsers) and servers.

Key Concepts of HTTP

1.Client-Server Model:

1. Client: The entity making the request (e.g., a web browser).
2. Server: The entity responding to the request (e.g., a web server hosting a website).

2.Requests and Responses:

1. HTTP Request: A message sent by the client to initiate an action on the server. It consists of a request line, headers, and sometimes a body.
2. HTTP Response: A message sent by the server in response to an HTTP request. It consists of a status line, headers, and a body.

Retrieving an Image over HTTP

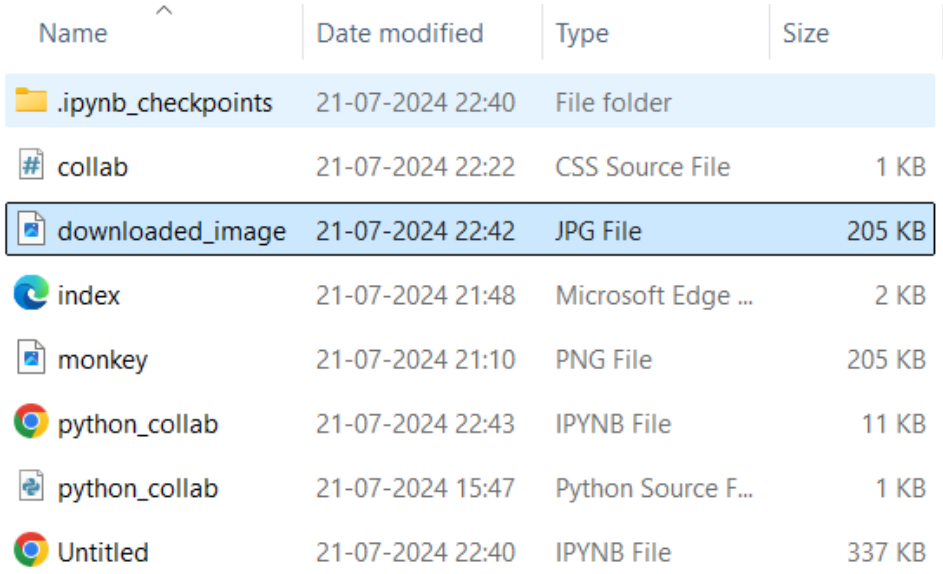
In [78]:
```python
import requests

# URL of the image
image_url = 'https://hbharathbhat.github.io/python_collab_act/monkey.png'

# Send an HTTP GET request to the image URL
response = requests.get(image_url, stream=True)

# Check if the request was successful
if response.status_code == 200:
    # Open a file in binary write mode
    with open('downloaded_image.jpg', 'wb') as out_file:
        # Write the content of the response (image) to the file
        for chunk in response.iter_content(chunk_size=8192):
            out_file.write(chunk)
    print('Image successfully downloaded and saved as downloaded_image.jpg')
else:
    print('Failed to retrieve image from', image_url)
```

```
Image successfully downloaded and saved as downloaded_image.jpg
```



1. Define the Image URL
2. Send an HTTP GET Request
3. Check if the Request was Successful
4. Save the Image to a File

Beautiful Soup

- BeautifulSoup is a Python library for pulling data out of HTML and XML files.
- It works with your any parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.
- It commonly saves programmers hours or days of work.

- https://hbharathbhat.github.io/python_collab_act
- https://quotes.toscrape.com/
- D:\projects\python_collab\index.html

Beautiful Soup

```
|: from bs4 import BeautifulSoup
   import requests
```

```
page=open('index.html','r')
soup=BeautifulSoup(page,'html')
print(soup)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
<title>Missing Students</title>
<link href="collab.css" rel="stylesheet" type="text/css"/>
</head>
<body>
<br/><br/><br/><br/><br/>
<header>Names of People Missing</header>
<p class="para01">
        These people were found missing from JSSATE
        </p><table>
<thead>
<th>Names</th>
<th>USN</th>
<th>Contact</th>
</thead>
<tbody>
<tr>
<td><a href="https://hbharathbhat.github.io/Portfolio/">H Bharath Bhat</a></td>
<td>1JS21EC052</td>
<td><a href="mailto:bharathbhat2805@gmail.com">Email</a></td>
</tr>
<tr>
<td>Hrushik Raj S</td>
<td>1JS21EC058</td>
<td><a href="mailto:hrushikrajs@gmail.com">Email</a></td>
```

1. Import BeautifulSoup Library
2. Open the Local HTML File
3. Parse the HTML Content Using BeautifulSoup

Beautiful Soup

Simple ways to navigate that data structure

In [57]:
```
print(soup.title)
```

```
<title>Missing Students</title>
```

In [58]:
```
print(soup.title.name)
```

```
title
```

In [59]:
```
print(soup.title.string)
```

```
Missing Students
```

In [60]:
```
for link in soup.find_all('a'):
    print(link.get('href'))
```

```
https://hbharathbhat.github.io/Portfolio/
mailto:bharathbhat2805@gmail.com
mailto:hrushikrajs@gmail.com
mailto:1js21ec051@jssateb.ac.in
mailto:bharathbhat2805@gmail.com
```

In [86]:
```
print(soup.a)
```

```
<a href="https://hbharathbhat.github.io/Portfolio/">H Bharath Bhat</a>
```

```
print(soup.get_text())
```

```
Missing Students



Names of People Missing

        These people were found missing from JSSATE


Names
USN
Contact



H Bharath Bhat
1JS21EC052
Email


Hrushik Raj S
1JS21EC058
Email
```

Retrieving Web Pages with urllib

In [27]:
```
import urllib.request
request_url = urllib.request.urlopen('https://hbharathbhat.github.io/python_collab_act')
print(request_url.read())
```

```
b'<!DOCTYPE html>\r\n<html lang="en">\r\n<head>\r\n    <meta charset="UTF-8">\r\n    <meta name="viewport" content="width=devic
e-width, initial-scale=1.0">\r\n    <title>Missing Students</title>\r\n    <link rel="stylesheet" type="text/css" href="collab.
css">\r\n</head>\r\n<body>\r\n    <br><br><br><br><br>\r\n    <header>Names of People Missing</header>\r\n    <p class="para0
1">\r\n        These people were found missing from JSSATE\r\n        <table>\r\n            <thead>\r\n                <th>Nam
es</th>\r\n                <th>USN</th>\r\n                <th>Contact</th>\r\n            </thead>\r\n            <tbody>\r\n
<tr>\r\n                <td><a href="https://hbharathbhat.github.io/Portfolio/">H Bharath Bhat</a></td>\r\n
<td>1JS21EC052</td>\r\n                <td><a href="mailto:bharathbhat2805@gmail.com">Email</a></td>\r\n                </t
r>\r\n                <tr>\r\n                <td>Hrushik Raj S</td>\r\n                <td>1JS21EC058</td>\r\n
<td><a href="mailto:hrushikrajs@gmail.com">Email</a></td>\r\n                </tr>\r\n                <tr>\r\n
<td>HA Yeshwanth</td>\r\n                <td>1JS21EC051</td>\r\n                <td><a href="mailto:1js21ec051@jssateb.
ac.in">Email</a></td>\r\n                </tr>\r\n                <tr>\r\n                <td>Bharath Kumar SP</td>\r\n
<td>1JS21EC0xx</td>\r\n                <td><a href="mailto:bharathbhat2805@gmail.com">Email</a></td>\r\n                </t
r>\r\n            </tbody>\r\n        </table>\r\n        <img src="monkey.png" height="200px" width="300px" caption="Monkey" a
lt="hrushik_raj_s">\r\n    </p>\r\n\r\n    <p id="para02">Bharath is a legend</p>\r\n</body>\r\n</html>'
```

import urlib.request from urllib library

Retrieving Web Pages with urllib

- The urllib.request module defines the following functions:
- urllib.request.urlopen(url, data=None, [timeout,]*, cafile=None, capath=None, cadefault=False, context=None)
- Open url, which can be either a string containing a valid, properly encoded URL, or a Request object.
- data must be an object specifying additional data to be sent to the server, or None if no such data is needed. See Request for details.
- urllib.request module uses HTTP/1.1 and includes Connection:close header in its HTTP requests.
- The optional timeout parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). This actually only works for HTTP, HTTPS and FTP connections.
- If context is specified, it must be a ssl.SSLContext instance describing the various SSL options. See HTTPSConnection for more details.
- The optional cafile and capath parameters specify a set of trusted CA certificates for HTTPS requests. cafile should point to a single file containing a bundle of CA certificates, whereas capath should point to a directory of hashed certificate files. More information can be found in ssl.SSLContext.load_verify_locations().
- The cadefault parameter is ignored.

Parsing html and scraping the Web

In [77]:
```
page_to_scrape=requests.get("https://quotes.toscrape.com")
soup=BeautifulSoup(page_to_scrape.text,"html.parser")
quotes=soup.findAll("span",attrs={"class":"text"})
authors=soup.findAll("small",attrs={"class":"author"})

for quote in quotes:
    print(quote.text)
for author in authors:
    print(author.text)
```

```
“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”
“It is our choices, Harry, that show what we truly are, far more than our abilities.”
“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a mirac
le.”
“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”
“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”
“Try not to become a man of success. Rather become a man of value.”
“It is better to be hated for what you are than to be loved for what you are not.”
“I have not failed. I've just found 10,000 ways that won't work.”
“A woman is like a tea bag; you never know how strong it is until it's in hot water.”
“A day without sunshine is like, you know, night.”
Albert Einstein
J.K. Rowling
Albert Einstein
Jane Austen
Marilyn Monroe
Albert Einstein
André Gide
Thomas A. Edison
Eleanor Roosevelt
Steve Martin
```

Parsing HTML using RE

Reading Binary Files using urllib

XML

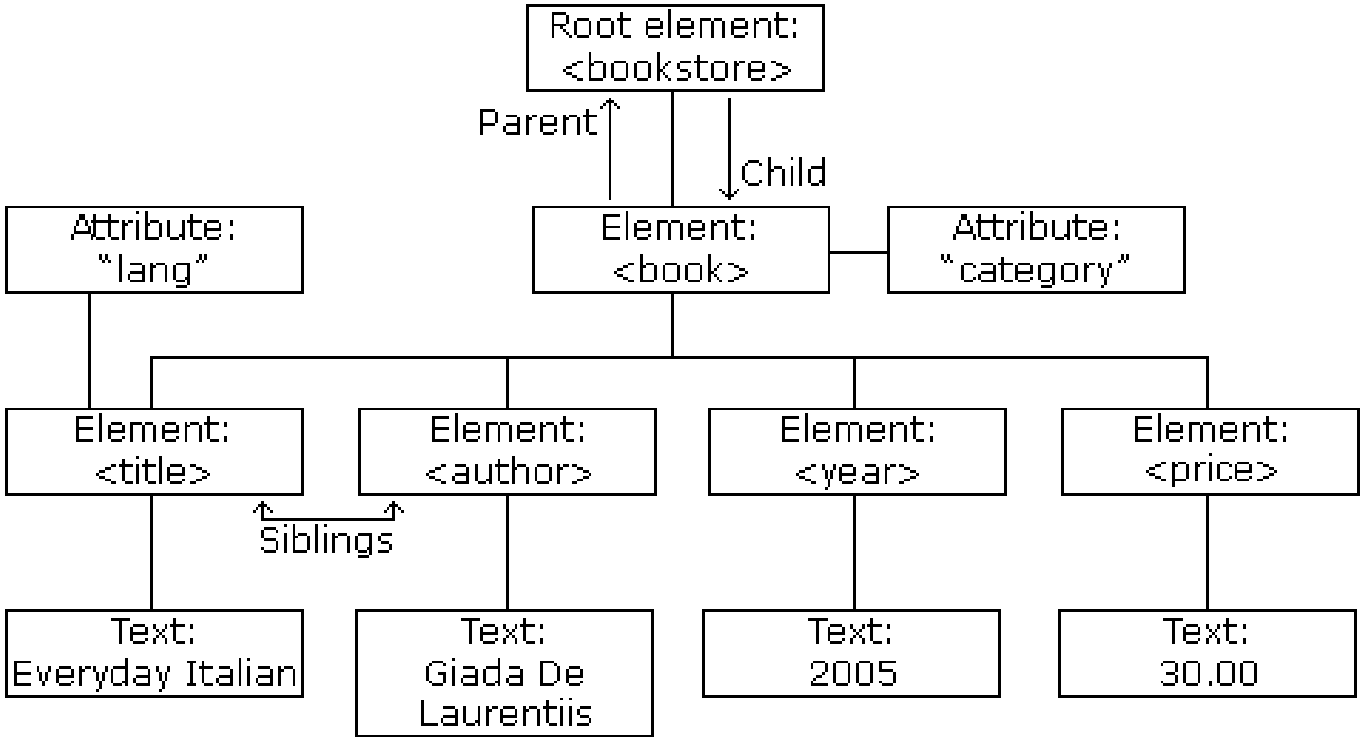
- XML stands for eXtensible Markup Language.
- XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable & machine-readable.
- XML plays an important role in many different IT systems.

Example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>robb</to>
    <from>Jane</from>
    <heading>Reminder</heading>
    <body>Don't forget to watch movie this weekend!</body>
</note>
```

XML

- XML documents form a tree structure that starts at "the root" and branches to "the leaves".



XML Tree Structure

Parsing XML

XML File:

xml-eg2.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<university>
    <student>
        <name>John Doe</name>
        <age>22</age>
        <major>Computer Science</major>
        <courses>
            <course>Introduction to Programming</course>
            <course>Data Structures</course>
            <course>Algorithms</course>
        </courses>
    </student>
    <student>
        <name>Jane Smith</name>
        <age>21</age>
        <major>Mathematics</major>
        <courses>
            <course>Calculus I</course>
            <course>Linear Algebra</course>
            <course>Statistics</course>
        </courses>
    </student>
</university>

```

Parsing Program Code:

xml-eg2.xml | parsexml.py

```python
import xml.etree.ElementTree as ET

#parse XML into element Tree
tree = ET.parse('xml-eg2.xml')
root = tree.getroot()

#root
print(root.tag)
print(len(root))
#access root child
print(root[0].tag)
print(len(root[0]))
 #loop over root children
for child in root:
    print(child[0].tag)
    print(child[0].text)
#root Grandchildren
print(root[0][0].tag) #name
print(root[0][0].text)
print(root[0][1].tag) #age
print(root[0][1].text)
print(root[0][2].tag) #major
print(root[0][2].text)
print(root[0][3].tag) #courses
print(root[0][3].text)
```

Looping through Nodes

- Looping through nodes in XML refers to the process of iterating over the elements in an XML document to access, read, or manipulate their values.

parsexml.py | xml-eg2.xml

```
1  import xml.etree.ElementTree as ET
2  #parse XML into element Tree
3  tree = ET.parse('xml-eg2.xml')
4  root = tree.getroot()
5   #iterate over each student node
6  for student in root.findall('student'):
7      name = student.find('name').text
8      age = student.find('age').text
9      major = student.find('major').text
10     courses = [course.text for course in student.find('courses').findall('course')]
11     #print student details
12     print(f"Name: {name}")
13     print(f"Age: {age}")
14     print(f"Major: {major}")
15     print("Courses: " + ", ".join(courses))
16     print()  #  newline '''
```

JSON, Parsing JSON

- JSON stands for JavaScript Object Notation.
- JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.
- It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition.
- JSON is used to send data between computers.
- JSON is language independent.

JSON Example:

```
'{"name":"John", "age":30, "car":null}'
```

JSON

The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.

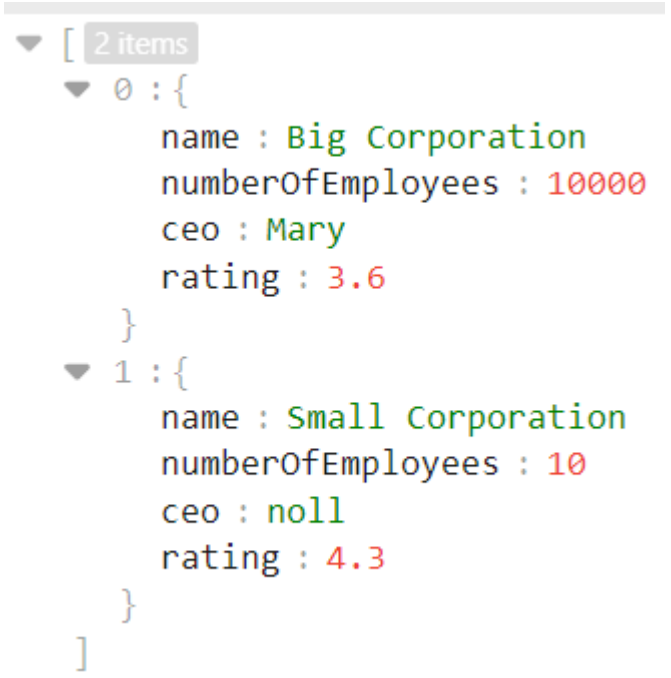
Applications of JSON:

- Web Development
- APIs
- Configuration Files
- Data Storage
- Serialization

and Deserialization

```
[
{
    "name": "Big Corporation",
    "numberOfEmployees": 10000,
    "ceo": "Mary",
    "rating": 3.6
},
{
    "name": "Small Corporation",
    "numberOfEmployees": 10,
    "ceo": "noll",
    "rating": 4.3
}
]
```

Example Code



Tree View

Parsing JSON

- To parse JSON string Python firstly we import the JSON module.

json-eg22.json | parsejson.py

```json
{
  "library": [
    {
      "title": "To Kill a Mockingbird",
      "author": "Harper Lee",
      "publishedYear": 1960,
      "genres": ["Fiction", "Classic"],
      "available": true
    },
    {
      "title": "1984",
      "author": "George Orwell",
      "publishedYear": 1949,
      "genres": ["Fiction", "Dystopian"],
      "available": false
    },

  ]
}
```

JSON File

json-eg22.json | parsejson.py

```python
import json
#Load the JSON file
with open('json-eg22.json', 'r') as file:
    data = json.load(file)
for book in data['library']:
    title = book['title']
    author = book['author']
    published_year = book['publishedYear']
    genres = ", ".join(book['genres'])
    available = book['available']
    #print book details
    print(f"Title: {title}")
    print(f"Author: {author}")
    print(f"Published Year: {published_year}")
    print(f"Genres: {genres}")
    print(f"Available: {'Yes' if available else 'No'}")
    # newline
    print()
```

Parsing Code

API

- API stands for Application Programming Interface.
- API is a collection of communication protocols and subroutines used by various programs to communicate between them.
- API facilitates programmers with an efficient way to develop their software programs.
- We can create an API for an operating system,

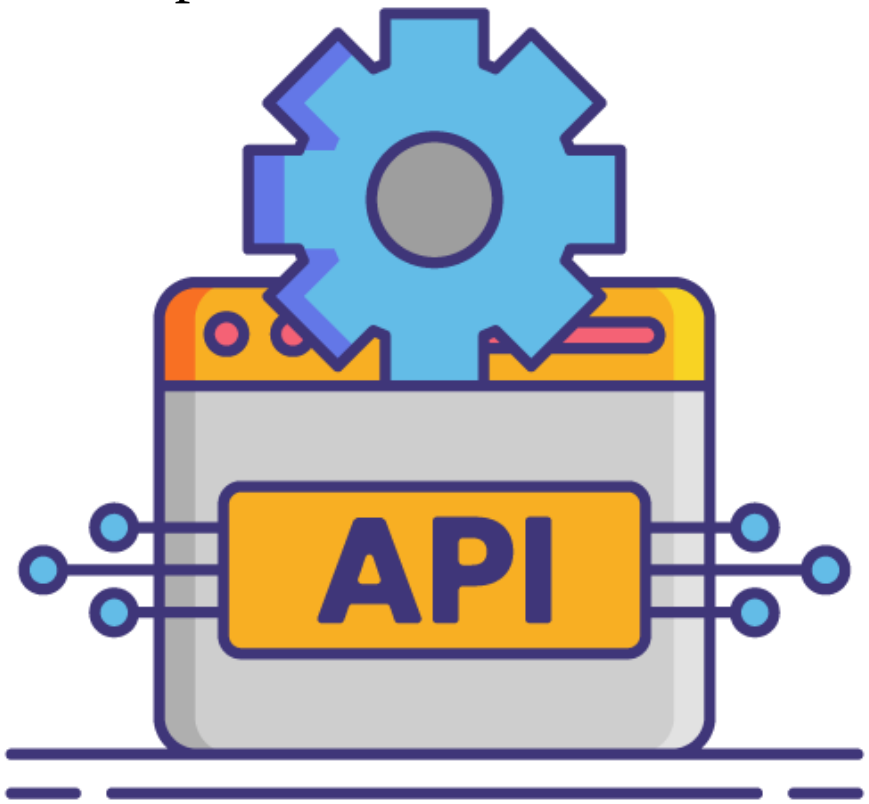
database system, hardware system, JavaScript file, or

similar object-oriented files.

- Advantages: Efficiency, Integration, Automation,

New Functionalities.

- Disadvantages: Cost, Security Issues.



API

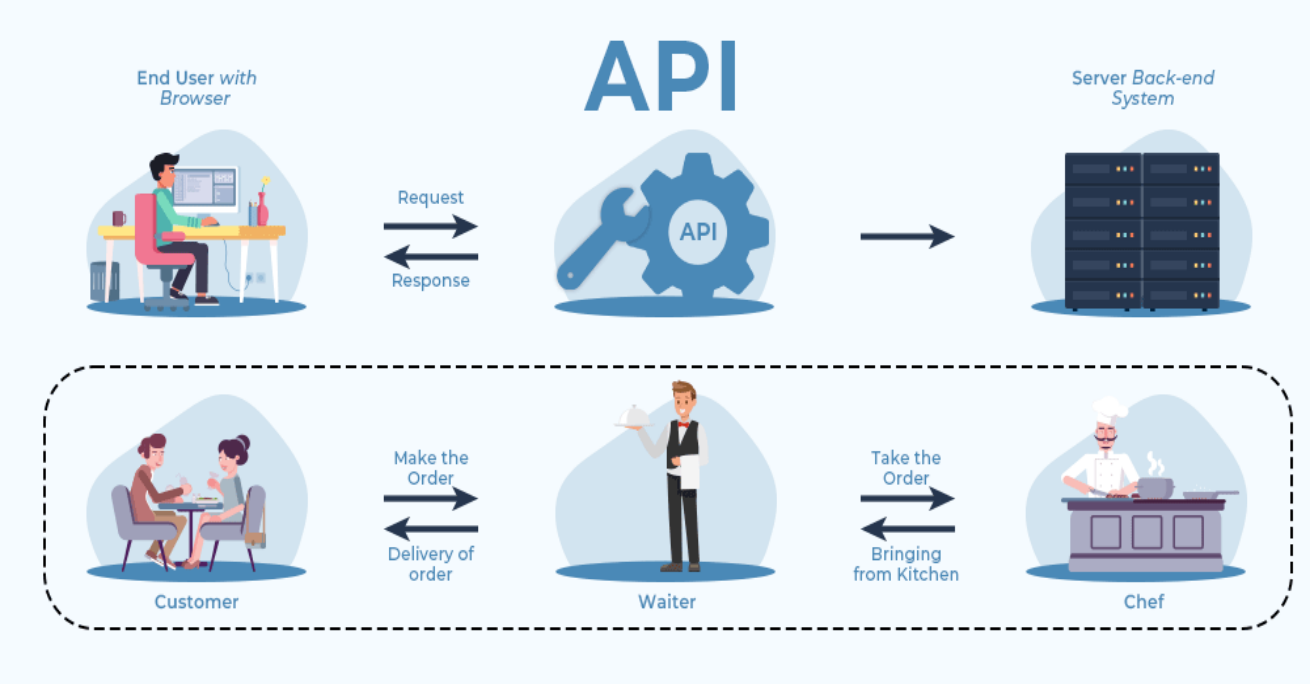
API's architectures are:

1. REST (Representational State Transfer): REST API is a way of accessing web services in a simple and flexible way without having any processing.
2. SOAP (Simple Object Access Protocol): Simple Object Access Protocol(SOAP) is a network protocol for exchanging structured data between nodes.

Types of APIs

There are three basic forms of API:

1. WEB APIs
2. LOCAL APIs
3. PROGRAM APIs



GEOCODING WEB SERVICES

Geocoding web services APIs are tools that convert addresses or place names into geographic coordinates (latitude and longitude) and vice versa. These services are essential for applications that need to place or find locations on a map, perform spatial analysis, or integrate location-based features.

Forward Geocoding: Converting an address or place name into geographic coordinates.

Example: Converting "1600 Amphitheatre Parkway, Mountain View, CA" into latitude and longitude.

Reverse Geocoding: Converting geographic coordinates into a human-readable address or place name.

Example: Converting latitude 37.423021 and longitude -122.083739 into "1600 Amphitheatre Parkway, Mountain View, CA".

GEOCODING WEB SERVICES

GEO-CODING.py

```python
import requests
#Replace 'YOUR_API_KEY' with your actual Google Maps API key
api_key = 'YOUR_API_KEY'
#Forward Geocoding: Address to Coordinates
address = '1600 Amphitheatre Parkway, Mountain View, CA'
forward_url = f'https://maps.googleapis.com/maps/api/geocode/json?address={address}&key={api_key}'
forward_response = requests.get(forward_url)
forward_data = forward_response.json()
if forward_data['status'] == 'OK':
    forward_results = forward_data['results'][0]
    location = forward_results['geometry']['location']
    print("Forward Geocoding:")
    print(f"Address: {forward_results['formatted_address']}")
    print(f"Latitude: {location['lat']}")
    print(f"Longitude: {location['lng']}")
else:
    print("Forward Geocoding Error:", forward_data['status'])
print("\n")  # Newline for separation
#reverse Geocoding: Coordinates to Address
latitude = 37.423021
longitude = -122.083739
reverse_url = f'https://maps.googleapis.com/maps/api/geocode/json?latlng={latitude},{longitude}&key={api_key}'
reverse_response = requests.get(reverse_url)
reverse_data = reverse_response.json()
if reverse_data['status'] == 'OK':
    reverse_results = reverse_data['results'][0]
    print("Reverse Geocoding:")
    print(f"Coordinates: ({latitude}, {longitude})")
    print(f"Address: {reverse_results['formatted_address']}")
else:
    print("Reverse Geocoding Error:", reverse_data['status'])

```

References

BeautifulSoup Documentation

Geeksforgeeks

W3schools

JSON documentation

JSONeditoronline