Python | Main course

Session 22 & 23

Regex

Python Regex

HTTP requests

Python requests module

JSON

by Mohammad Amin H.B. Tehrani

www.maktabsharif.ir

Regex

Regex

Example

Write a python function that validates emails string.

Hint:
Email addresses only contains: words, digits, dots, periods
+ contains ‘@’ character,
+ a valid domain name or IP

```
def email_validator(email) -> bool:
    # TODO: Code here
    ...

# example@email.co -> True
# exampleemail.co -> False
# akbar -> False
# asd @ gmail.com -> False
# akbar.babaii@yahoo.com -> True
```

Valid test cases:

- email@example.com
- firstname.lastname@example.com
- email@subdomain.example.com
- firstname+lastname@example.com
- email@123.123.123.123

Invalid test cases:

- plainaddress
- #@%^%#$@#$@#.com
- @example.com
- Joe Smith <email@example.com>
- email.example.com
- email@example@example.com

Intro

A **RegEx**, or **Regular Expression**, is a sequence of characters that forms a search pattern.
RegEx can be used to check if a string contains the specified search pattern.

Examples:

- `(\d{1,3})(\.)(\d{1,3})(\.)(\d{1,3})(\.)(\d{1,3}): IP address`
  ```
  -> 11.2.1.2
  -> 127.0.0.1
  -> ...
  ```
- `(\d{4})[\.\-\/](\d{2})[\.\-\/](\d{2}): Date`
  ```
  -> 1922-02-02
  -> 1340/02/01
  -> ...
  ```
- `(www.)?([\w\-]+\.)?([\w\-]+)\.([\w\-]{2,})(\/.*)?: ?`

Regex

Metacharacters

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"

Regex

Special Sequences

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"

Regex

Sets

Set	Description
[arn]	Returns a match where one of the specified characters (a, r, or n) are present
[a-n]	Returns a match for any lower case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a, r, and n
[0123]	Returns a match where any of the specified digits (0, 1, 2, or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z, lower case OR upper case
[+]	In sets, +, *, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string

Some useful references...

- https://regexr.com/
  A editor, document, reference, community for Regex.

- https://www.w3schools.com/python/python_regex.asp
  Python Regex reference.

Practice

Write a Regex that finds (validate) email addresses.

Hint:
Email addresses only contains: words, digits, dots, periods
+ contains ‘@’ character,
+ a valid domain name or IP

WRITE IT YOURSELF..

Valid test cases:

- email@example.com
- firstname.lastname@example.com
- email@subdomain.example.com
- firstname+lastname@example.com
- email@123.123.123.123

Invalid test cases:

- plainaddress
- #@%^%#$@#$@#.com
- @example.com
- Joe Smith <email@example.com>
- email.example.com
- email@example@example.com

Python Regex

Regex module

Python has a built-in package called **re**, which can be used to work with Regular Expressions.

Syntax:

```
import re
```

```
import re

# Check if the string starts with "The" and ends with "Spain":
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)

if x:
   print("YES! We have a match!")
else:
   print("No match")
```

Regex

findall() method

The **findall()** function returns a list containing all matches.

```python
import re

txt = """Python was conceived in the late 1980s[38] by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in
the Netherlands as a successor to ABC programming language, which was inspired by SETL,[39] capable of exception
handling and interfacing with the Amoeba operating system.[9] Its implementation began in December 1989.[40] """

references = re.findall("(\[[\d+\])", txt)
print(references)
```

```
['[38]', '[39]', '[9]', '[40]']
```

```python
import re

txt = """Akbar's reign was chronicled extensively by his court historian Abul Fazl in the books Akbarnama and
Ain-i-akbari. Other contemporary sources of Akbar's reign include the works of Badayuni, Shaikhzada Rashidi and
Shaikh Ahmed Sirhindi."""

upper_cases = re.findall("([A-Z]\w*)", txt)
print(upper_cases)
```

```
['Akbar', 'Abul', 'Fazl', 'Akbarnama', 'Ain', 'Other', 'Akbar',
'Badayuni', 'Shaikhzada', 'Rashidi', 'Shaikh', 'Ahmed', 'Sirhindi']
```

Regex

search() method

The **search()** function searches the string for a match, and returns a **Match** object if there is a match.
If there is more than one match, only the first occurrence of the match will be returned:

```
import re

txt = """Non tempora amet 1994-02-24 18:26:25.680292 est. Sed dolor labore ut labore velit porro tempora.
Quisquam
dolor non voluptatem. Numquam quiquia adipisci dolore eius numquam amet voluptatem.
14:39:40.982917 est. Ut tempora quisquam amet  1998-03-16 16:14:16.647591..."""

pattern = r"(\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2}(.\d+)?)"
timestamp = re.search(pattern, txt)
print(timestamp)
```

```
<re.Match object; span=(17, 43), match='1994-02-24 18:26:25.680292'>
```

Regex

finditer() method

Return an iterator yielding Match Object instances over all non-overlapping matches for the RE pattern in string.

```python
import re

txt = """Non tempora amet 1994-02-24 18:26:25.680292 est. Sed dolor labore ut labore velit porro tempora.
Quisquam
dolor non voluptatem. Numquam quiquia adipisci dolore eius numquam amet voluptatem.
14:39:40.982917 est. Ut tempora quisquam amet  1998-03-16 16:14:16.647591..."""

pattern = r"(\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2}(.\d+)?)"
for ts in re.finditer(pattern, txt):
    print(ts)
```

```
<re.Match object; span=(17, 43), match='1994-02-24 18:26:25.680292'>
<re.Match object; span=(295, 321), match='1998-03-16 16:14:16.647591'>
<re.Match object; span=(347, 373), match='2006-02-04 09:14:57.833855'>
...
```

Match object

A **Match Object** is an object containing information about the search and the result.

The Match object has properties and methods used to retrieve information about the search, and the result:

- .span() returns a tuple containing the start-, and end positions of the match.
- .string returns the string passed into the function
- .group() returns the part of the string where there was a match
- .groups() returns all groups tuple
- .groupdict() returns all groups dict

Regex

Example

Complete source:
Github Repo

Extract times from string:

```python
import re

txt = """Non tempora amet 1994-02-24 18:26:25.680292 est. Sed dolor labore ut labore velit porro tempora.
Quisquam
dolor non voluptatem. Numquam quiquia adipisci dolore eius numquam amet voluptatem. Adipisci 2010-09-15
14:39:40.982917 non sed est quiquia dolore quisquam est. Ut tempora quisquam amet  1998-03-16 16:14:16.647591
..."""

time_pattern = r"((\d{2}):(\d{2}):(\d{2})(.\d+)?)"
for time_match in re.finditer(time_pattern, txt):
   print()
   print('Time:', time_match)
   print('Groups:', time_match.groups())
   print('Hour:', time_match.group(2))
   print('Minute:', time_match.group(3))
   print('Second:', time_match.group(4))
   print('Nano_secs:', time_match.group(5))
```

Practice: Sort lines by datetime

Maktab Sharif

Sort by datetime

Write a program that reads **test.txt** file, then sorts the lines by the timestamp that's mentioned in each line.

- Each line contains a timestamp in ISO format.

File link: https://github.com/mohammadT77/Makab52/blob/master/session22-23/test.txt

HTTP requests

HTTP requests

Intro

What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a **request-response** protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

HTTP Methods

- GET
- POST
- PUT
- HEAD
- DELETE
- PATCH
- OPTIONS

HTTP requests

Intro

What is HTTP?

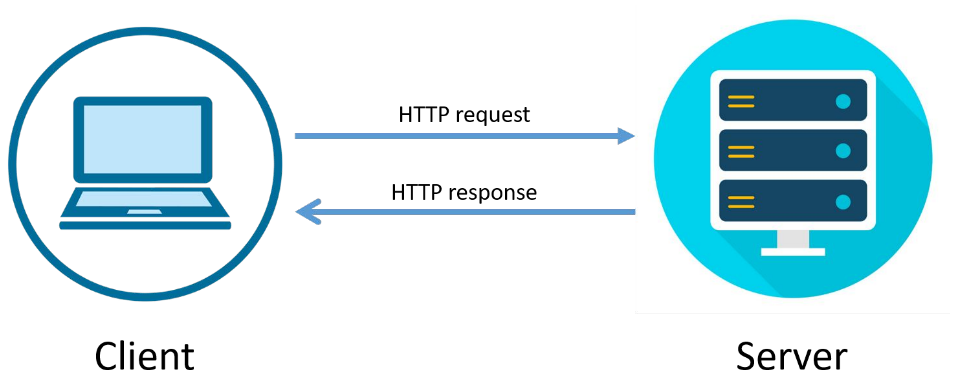
The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a **request-response protocol** between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

HTTP Methods

- GET
- POST
- PUT
- HEAD
- DELETE
- PATCH
- OPTIONS



HTTP requests

GET request

GET is used to request data from a specified resource.

GET is one of the most common HTTP methods.

Note that the query string (name/value pairs) is sent in the URL of a GET request:

```
http://google.com
```

```
http://ma-web.ir/maktab52
```

```
http://ma-web.ir/maktab52/?name=akbar
```

POST request

POST is used to send data to a server to create/update a resource.

The data sent to the server with POST is stored in the request **body** of the HTTP request:

HOW TO SEND POST REQUESTS?

HTTP requests

Curl

Use cURL to request a server

cURL is a computer software project providing a library and command-line tool for transferring data using various network protocols.
cURL[edit]. cURL is a command-line tool for getting or sending data including files using URL syntax.

```
m-tehrani@MohammadAmin:~$ curl http://ma-web.ir/maktab52/?name=akbar
<H1>GET</H1><p style='color:blue'>Hello akbar!</p>
```

```
m-tehrani@MohammadAmin:~$ curl http://ma-web.ir/maktab52/
-X POST --data name=akbar
<H1>POST</H1><p style='color:red'>Hello akbar!</p>
```

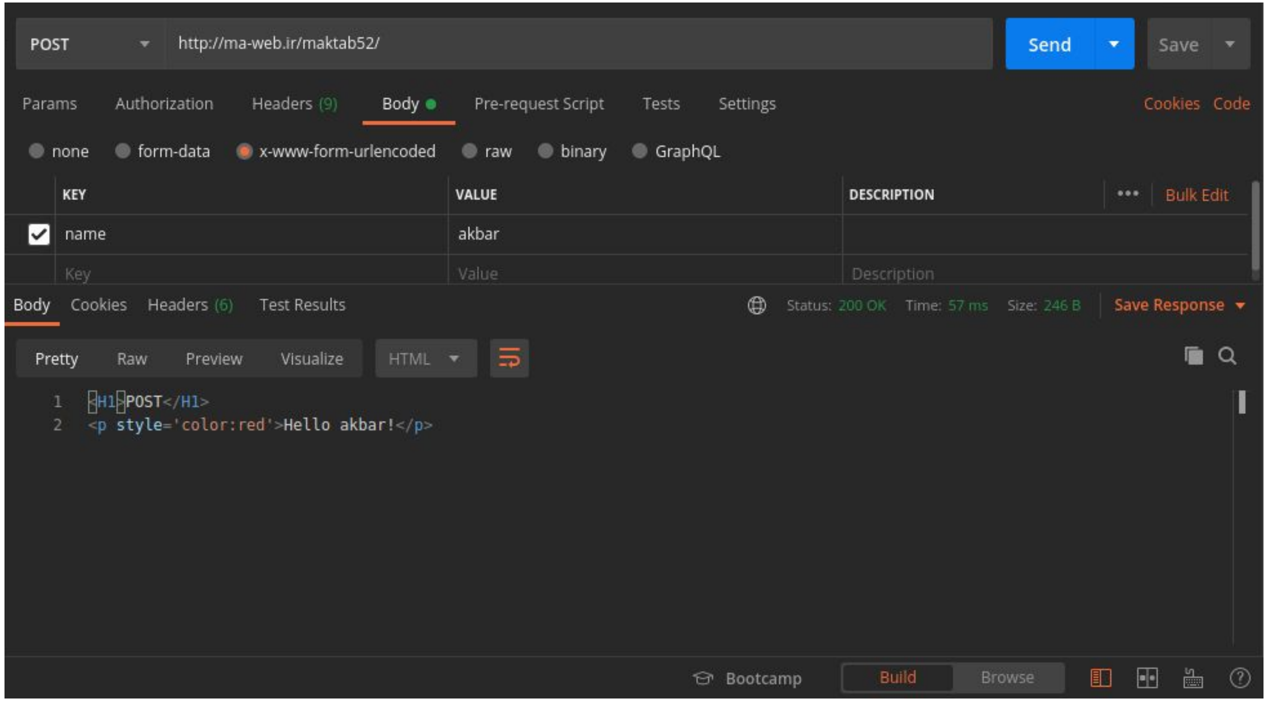
HTTP requests

Postman

POSTMAN

Use Postman to request a server

Download Postman app
Or Add extension to your browser


POST http://ma-web.ir/maktab52/

Body: x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
✓ name	akbar	

Status: 200 OK Time: 57 ms Size: 246 B

```
<H1>POST</H1>
<p style='color:red'>Hello akbar!</p>
```

Python requests module

Python requests module

Intro

The **requests** module allows you to send HTTP requests using Python.

The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

Install:

```
pip install requests
```

Import:

```
import requests
```

```python
import requests

url = 'http://ma-web.ir/maktab52'
method = 'GET'
response = requests.request(method, url)

print(response.text)
```

```
Enter your name please!
```

Python requests module

Methods

● **delete(url, args)**	Sends a DELETE request to the specified url
● **get(url, params, args)**	Sends a GET request to the specified url
● **head(url, args)**	Sends a HEAD request to the specified url
● **patch(url, data, args)**	Sends a PATCH request to the specified url
● **post(url, data, json, args)**	Sends a POST request to the specified url
● **put(url, data, args)**	Sends a PUT request to the specified url
● **request(method, url, args)**	Sends a request of the specified method to the specified url

Python requests module

Example

GET /maktab52/?name=akbar

```
import requests

url = 'http://ma-web.ir/maktab52/'
method = 'GET'
get_response = requests.request(method , url, params={'name': 'akbar'})  # = request.get(url,
...)
print(get_response.content)
```

```
<H1>GET</H1><p style='color:blue'>Hello akbar!</p>
```

POST /maktab52

```
import requests

url = 'http://ma-web.ir/maktab52/'
get response = requests.post(url , data={'name': 'akbar'})  # = request.get(url, ...)
print(get_response.text)
```

```
<H1>POST</H1><p style='color:red'>Hello akbar!</p>
```

Python requests module

Practice: Wikipedia headings & paragraphs

Maktab Sharif

Wikipedia headings & paragraphs

Write a script that gets an wikipedia query from user, then downloads wikipedia web page, saves it into a .html file, and extracts paragraphs (<p>) & headings (<h1> – <h6>) elements from html file.

- Arguments: positional argument *query* (eg. : python, RSA, JavaScript, …)
- Target url: **https://en.wikipedia.org/wiki/*{query}***
- Save content into a .html file.
- Use regex to extract content of <p> tags
- Use regex to extract content of <h1>, <h2>, …, <h6> tags
- Log informations like,getting response, saving the file, …

```
m-tehrani@MohammadAmin:~/PycharmProjects/Maktab52/session22-23$ python3 wikipedia.py python
<p><b>Python</b> may refer to:
</p>
<h1 class="firstHeading" id="firstHeading">Python</h1>
<h2 id="mw-toc-heading">Contents</h2>
```

BeautifulSoup

Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

Installation:

```
pip install beautifulsoup4
```

Import:

```
from bs4 import BeautifulSoup
```

```
from bs4 import BeautifulSoup

bs = BeautifulSoup('<h1>Hello world!</h1>')
print(bs.prettify())
print(bs.find(text='Hello world!'))
print(bs.find_all(name='h1'))
```

```
<html>
 <body>
  <h1>
   Hello world!
  </h1>
 </body>
</html>
Hello world!
[<h1>Hello world!</h1>]
```

```python
# wikipedia bs.py
import argparse
import logging
import requests

from bs4 import BeautifulSoup

if __name__ == '__main__':
   parser = argparse.ArgumentParser()
   parser.add_argument('query')

   args = parser.parse_args()

   url = f"https://en.wikipedia.org/wiki/{args.query}"
   resp = requests.get(url)
   html_content = resp.text
   file_name = args.query+'.html'
   with open(file_name, 'w') as f:
       f.write(html_content)
       logging.info('File saved at:', file_name)

   bs = BeautifulSoup(html_content, features="lxml")
   ps = bs.find_all('p')
   h1s = bs.find_all('h1')
   h2s = bs.find_all('h2')
   h3s = bs.find_all('h3')
   h4s = bs.find_all('h4')
   h5s = bs.find_all('h5')
   h6s = bs.find_all('h6')

   print(*ps, *h1s, *h2s, *h3s, *h4s, *h5s, *h6s, sep='\n')
```

Complete source:
Github Repo

BeautifulSoup Example

JSON

JSON

Intro

JavaScript Object Notation

- **JSON** stands for JavaScript Object Notation
- **JSON** is a lightweight format for storing and transporting data
- **JSON** is often used when data is sent from a server to a web page
- **JSON** is "self-describing" and easy to understand

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

Syntax:

- **“Key”: “value”**
- {...} object notation (contains key-value pairs)
- [...] list notation (contains objects)

```
{
 "users": [
   {
     "firstName": "John",
     "lastName": "Doe",
     "marks": [12,20,14,18,3,19],
     "address": {
       "country": "iran",
       "city": "karaj",
       "street": "..."
     }
   },
   {
     "firstName": "Anna",
     "lastName": "Smith",
     "marks": [12,20,14,18,3,19],
     "address": {
       "country": "USA",
       "city": "NY",
       "street": "..."
     }
   }
 ]
}
```

JSON

Python json module

Python has a built-in package called **json**, which can be used to work with JSON data.
Use can simply **load** json file (Deserialize python objects from strings)
Or **dump** json file (Serialize python objects into json string)

Methods:

- **loads():** Deserialize to a Python object.
- **dumps():** Serialize obj to a JSON formatted str.
- **load():** Deserialize a file to a Python object.
- **dump():** Serialize obj as a JSON formatted stream to file.

```
content = {
    'users': [
        {
            'first_name': 'akbar',
            'last_name': 'babaii',
            'phone': '09379880665',
            'is_admin': True
        }
    ]
}
```

```
import json

# Serialize content in to json file
with open('test.json', 'w') as f:
    json.dump(content, f)
```

```
import json

# Deserialize content from .json file
with open('test.json', 'r') as f:
    content = json.load(f)
```

JSON

Python json module

Python has a built-in package called **json**, which can be used to work with JSON data.
Use can simply **load** json file (Deserialize python objects from strings)
Or **dump** json file (Serialize python objects into json string)

Methods:

● **loads():**	Deserialize to a Python object.
● **dumps():**	Serialize obj to a JSON formatted str.
● **load():**	Deserialize a file to a Python object.
● **dump():**	Serialize obj as a JSON formatted stream to file.

```
content = {
    'users': [
        {
            'first_name': 'akbar',
            'last_name': 'babaii',
            'phone': '09379880665',
            'is_admin': True
        }
    ]
}
```

```
import json

# Serialize content in to json file
with open('test.json', 'w') as f:
    json.dump(content, f)
```

```
import json

# Deserialize content from .json file
with open('test.json', 'r') as f:
    content = json.load(f)
```

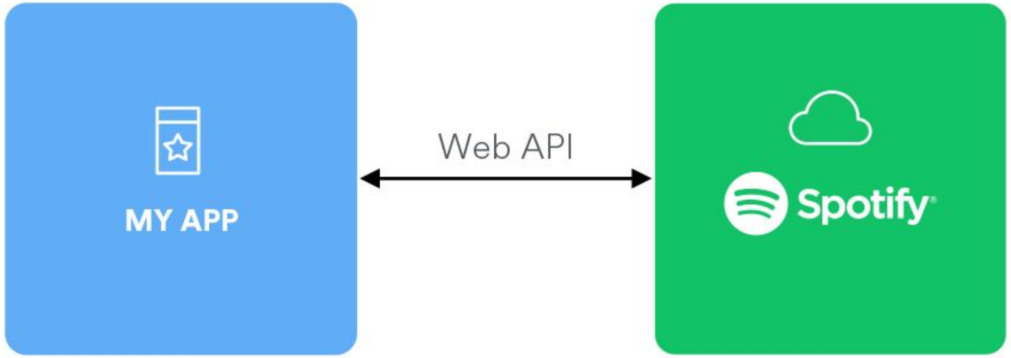
API

What is Web API?

- API stands for Application Programming Interface.
- A Web API is an application programming interface for the Web.
- A Browser API can extend the functionality of a web browser.
- A Server API can extend the functionality of a web server other.

How can application talk to each other??

API is the acronym for Application Programming Interface, which is a software intermediary **that allows two applications to talk to each other.** Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.



JSON

Example: Neshan ‘Search’ map api

Complete source:
Github Repo

Document link: https://platform.neshan.org/api/search

```python
from pprint import pprint
import requests

search_location_api_url = "https://api.neshan.org/v1/search"

# Authentication API token
api_key = "service.GDwNbS3U7svILS8SwBIYqpH5PpL3rrq2Fz9v3kK8"

params = {
    'term': 'اکبر',
    'lat': 36.2880443,
    'lng': 59.615743,
}

resp = requests.get(search_location_api_url, params=params,
                    headers={'Api-key': api_key})
pprint(resp.json())
```

```
{'count': 30,
 'items': [{'address': 'بلوار شهید کاوه',
            'category': 'place',
            'location': {'x': 59.539951872066005,
                         'y': 36.278954087954936,
                         'z': 'NaN'},
            'neighbourhood': 'محله زکریا',
            'region': 'مشهد، خراسان رضوی',
            'title': 'بیمارستان کودکان اکبر',
            'type': 'hospital'},
           {'address': 'صدوقی 23',
            'category': 'municipal',
            'location': {'x': 59.6299085648893,
                         'y': 36.270723962700615,
                         'z': 'NaN'},
            'neighbourhood': 'محله شیرودی',
...
```

JSON

Practice: Neshan ‘Search’ map api

Complete source:
Github Repo

Document link: https://platform.neshan.org/api/search

```python
from pprint import pprint
import requests

search_location_api_url = "https://api.neshan.org/v1/search"

# Authentication API token
api_key = "service.GDwNbS3U7svILS8SwBIYqpH5PpL3rrq2Fz9v3kK8"

params = {
    'term': 'اکبر',
    'lat': 36.2880443,
    'lng': 59.615743,
}

resp = requests.get(search_location_api_url, params=params,
                    headers={'Api-key': api_key})
pprint(resp.json())
```

```
{'count': 30,
 'items': [{'address': 'بلوار شهید کاوه',
            'category': 'place',
            'location': {'x': 59.539951872066005,
                         'y': 36.278954087954936,
                         'z': 'NaN'},
            'neighbourhood': 'محله زکریا',
            'region': 'مشهد، خراسان رضوی',
            'title': 'بیمارستان کودکان اکبر',
            'type': 'hospital'},
           {'address': '23 صدوقی',
            'category': 'municipal',
            'location': {'x': 59.6299085648893,
                         'y': 36.270723962700615,
                         'z': 'NaN'},
            'neighbourhood': 'محله شیرودی',
...
```

Practice: Neshan reverse Geocoding

Maktab Sharif

Neshan reverse Geocoding

Write a script that gets two positional arguments **lat**, **lng** then requests to **neshan reverse geocoding API**, fetches results, and saves response json content into a json file and prints it.

API url: https://api.neshan.org/v2/reverse
Document: https://platform.neshan.org/api/reverse-geocoding
API Key: **service.GDwNbS3U7svILS8SwBIYqpH5PpL3rrq2Fz9v3kK8**

Hint: Try the api using postman, before start coding.

Advanced topics

- Timezone
- Jdatetime
- Venv (Virtual Environment)