Reading from Web using APIs

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Input for Programs

- Programs solve problems - for specific instance of the problem inputs for that instance are to be given as input to the program
- To provide input data to programs, we have seen
    - Input through terminals - limiting
    - Input through files - versatile; you need to get data in a local file
- For both, python interacts with OS to get the data

Data from the Web

- Websites accessed using HTTP, i.e. an http request is sent to the serving machine, which responds by sending a page
- Often returns data in html format - which can be displayed by a browser; meant for human consumption
- But programs can process this information also - programs "scrape" the webpages for information

Websites Providing Data

- Many websites have a lot of data, and they sometimes want to make it available for processing also not just viewing (free or paid)
  - Simple model - provide data for download - user downloads data from the webpage in a file (e.g. csv); cannot access programmitically
- API based - provides Application Programming Interface for programs to "read" data from website - programs can "read"
- With APIs, the whole web is a source of input data for programs - hugely empowering what programs you can write
- If a website provides APIs, then you can access it through programs (generally servers will have programs to process these requests)

Accessing APIs (using HTTP)

- The request and responses for APIs use HTTP for the client-server interaction - client sends a request, and server sends a response
- Commonly the client requests are of these types:
    - GET - read / retrieve resource representation/information
    - (POST - create new resources, e.g., a file)
    - (PUT - update an existing resource; DELETE - delete a resource)
- For reading an API, we will be using GET requests only
    - If resource found on server, return HTTP response code 200 (OK), along with response body (generally XML or JSON content
    - If resource is NOT found, API returns HTTP code 404 (NOT FOUND).
    - If GET request not correctly formed, return code 400 (BAD REQUEST)

APIs and Endpoints

- An API is a regular URL - which provides data in some format (e.g. text, JSON, binary, …)
    - As a URL, you can see it on browser also
- A website typically provides many APIs - each with its own URL - called end points
- Generally, they will have a base URL, like:
    - https: //api.twitter.com
    - https: //api.github.com
    - https://randomuser.me/api (.../documentation gives the doc)
- Endpoint - addition to base URL specifying the resource being requested

API reference / documentation

- For programs to process it, they need to understand the structure of data being returned by an API
- Most website will provide some API reference / documentation - to provide this info about the structure of data in response
  - With this knowledge, can write code to make a request, get the response, extract different components of the data from response, and use it…

How to make an API request in Python

- Import the requests package
- Make a request to an API, save the response
- Requests package handles communication with server - makes it look like a local call

```
#python code
import requests
resp = requests.get(<api-url>)
```

Response attributes

Response returned is an object with many attributes; key ones:

- resp.status_code # the status code of the request
- resp.headers # the headers of the response - dictionary in json
- resp.text # the text returned - a string; often json in a string
- resp.request # information about the request - like header, URL, ..
- Many others

status_code

Many standard status_code : these are really HTTP codes

- 200: OK, request successful
- 400: Bad request - request was incomplete
- 401: unauthorised - need credentials to access
- 404: Not found
- …

Response headers

Gives header information about the response, eg.

```
{'Access-Control-Allow-Credentials': 'true',
 'Access-Control-Allow-Methods': 'GET, POST',
 'Access-Control-Allow-Origin': '*',
 'Connection': 'keep-alive',
 'Content-Length': '451',
 'Content-Type': 'application/json; charset=utf-8',
 'Date': 'Thu, 17 Feb 2022 16:21:40 GMT',
 'Server': 'openresty',
 'X-Cache-Key':
'/data/2.5/weather?APPID=c628a3bc183d28f9acc2376d6ee0aaa4&q=delhi'}
```

Response text

- This is the main data/information in the response
- Is a text string - typically a dictionary in a string (particularly if the response is JSON)
- Can extract the dictionary from this string by resp.json() - with this we have dictionary containing all the response data; from the structure of response data, can extract relevant info, process it, …

Simple Example

- The site https://ip-api.com/ provides free API to get information about an IP address
- It is totally free and users can use it without any authentication or authorization

Simple Example

- The site https://dictionaryapi.dev/ provides free dictionary API
- It is totally free and users can use it without any authentication or authorization
- It has one main end point:
- `https://api.dictionaryapi.dev/api/v2/entries/en/<word>`
- Given a word, this API will returns info about <word>
- Lets try it - and see different parts of the response

JSON

- JSON (JavaScript object notation) is a text representation of data
- It can be used to represent all standard types of data - scalar as well as structured like lists, dictionaries, etc (it is like XML)
- It is commonly used to share structured data between programs
  - How do you share the dictionary you create in your program with your friend (e.g. so she can write some functions on that data)
- It is really like a python dictionary

- Now most websites provide data using JSON
- And response package provides converting it to a dictionary - by just invoking operation `json()` on the response (resp.json())

JSON Example

```
{
  "first_name": "Sherlock",
  "last_name": "Holmes",
  "gender":"Male",
  "age": 36,
  "occupation": "Detective",
  "is_active": True,
  "address": {
    "home": "221B Baker Street",
    "city": "London",
    "country": "United Kingdom"
  },
  "contact_info": [
    {
      "type": "home",
      "number": "+44 123 456 7890"
    },
    {
      "type": "office",
      "number": "+44 987 654 3210"
    }
  ],
  "friends": ["John Watson", "Mrs Hudson", "Lestrade"]
}
```

Authentication

- There are some open (and free) APIs
- But most websites typically want to allow access to only authorized users - done through authentication
- Two main methods of authentication
  - API keys - simpler
  - Oauth - more sophisticated
- We will only discuss briefly API keys

API Keys

- Most common level of authentication used by API providers
- Keys are provided to you on request and are to identify the user of the API
- Keys are sent as a query parameter - the name of this parameter is as specified by the API (APPID, api_key, …)
- Most sites will provide method to get your API key - that key you can use for all the APIs

API Query

- The endpoint often provide a lot of information (as few endpoints, provider may "dump" a lot of data)
- If you need only a small portion of it, or specific data, then your program has to first get all the data in memory (so prog requires all this memory) and then extract the portion it wants
- APIs also sometimes allow some query to be specified
- As in http, query parameters are after the URL, separating them using '?' - all after '?' are parms
- Parms are specified using keyword-value pairs (like keyword based argument passing). E.g.

Example – openweathermap.org

- Gives weather information for any place around the world
- Current weather, short term and long term forecasts, …
- Other information also available
- Many APIs - https://openweathermap.org/api - has doc for each, giving brief description, the format of the API call including the parameters needed, as well as the JSON response
- Need an API key even for free use
- Getting and API key - just enroll, confirm email, log in and API key will be there
- You can use the key for making any API call and multiple times…

Example...

Let us show use of the current weather api (doc: .../current)

Goal: Get the current temperature of a city (Delhi)

Use Geocoding api - where we can give the city name

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```

There are some other optional parms

```
import requests
api_url = "http://api.openweathermap.org/data/2.5/weather?"
loc = "Delhi"
api_key = "<my key>"
url = api_url+f'q={loc}&APPID={api_key}'
resp = requests.get(url)
if resp.status_code == 200:  # call was successful
    data = resp.json()
    print(f'Temp in {data["name"]}: {data["main"]["temp"]-272} C')
    print(f'Wind Speed is {data["wind"]["speed"]}')
else:
    print("Request failed, status code: ", resp.status_code)
```

Processing API Data

- To process API data, you need to understand structure of its text
    - Type the API endpoint in a browser - see the structure of data
    - Python terminal - make a call, get data, print it - see structure
    - Get data, see its keys - will give you top level structure; then dig deeper into the structure of values for the keys
- Generally get data in dictionary

```
data = requests.get(url).json()
```

Example 2 – Last.fm

- Last.fm is a service that allows users to “scrobble” or store and track what songs they have been listening to.
- These statistics can be found on each users profile page, on their website.

- Last.fm provides a rich API for accessing data about songs and music
    - Albums
    - Artists
    - Tracks

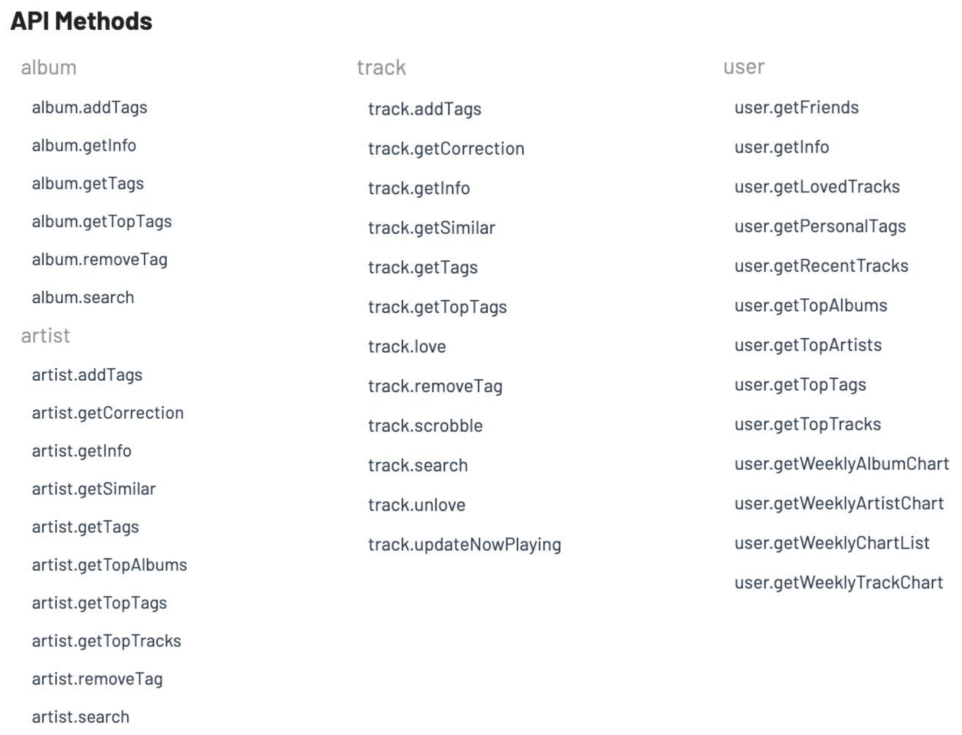
Example 2 – Last.fm

- Documentation for the API: https://www.last.fm/api
- Authentication using simple API Key query param
    - API Key can be registered (for free)
    - An API Key is tied to an authenticated user
    - The API Key can be used to retrieve AND update/create data

- Many types of queries can be asked:
    - Top albums by Kishore Kumar?
    - Top songs in India in 2015?
    - Currently trending Pop songs?

Example 2 – Last.fm

- Examples of “methods” or “endpoints”



API Methods

album

- album.addTags
- album.getInfo
- album.getTags
- album.getTopTags
- album.removeTag
- album.search

artist

- artist.addTags
- artist.getCorrection
- artist.getInfo
- artist.getSimilar
- artist.getTags
- artist.getTopAlbums
- artist.getTopTags
- artist.getTopTracks
- artist.removeTag
- artist.search

track

- track.addTags
- track.getCorrection
- track.getInfo
- track.getSimilar
- track.getTags
- track.getTopTags
- track.love
- track.removeTag
- track.scrobble
- track.search
- track.unlove
- track.updateNowPlaying

user

- user.getFriends
- user.getInfo
- user.getLovedTracks
- user.getPersonalTags
- user.getRecentTracks
- user.getTopAlbums
- user.getTopArtists
- user.getTopTags
- user.getTopTracks
- user.getWeeklyAlbumChart
- user.getWeeklyArtistChart
- user.getWeeklyChartList
- user.getWeeklyTrackChart

Last.fm API Demo!

- **Question:** What is the best album by AP Dhillon? 🤔
- **Steps:**
  - Register an account on Last.fm
  - Create an API Key (https://www.last.fm/api/account/create)
  - Search the documentation for the appropriate method
  - Query the endpoint to get our answer!

- Method to use: `artist.getTopAlbums`
  - Query params:
    - Method
    - API Key
    - Artist name
    - Response format - JSON or XML

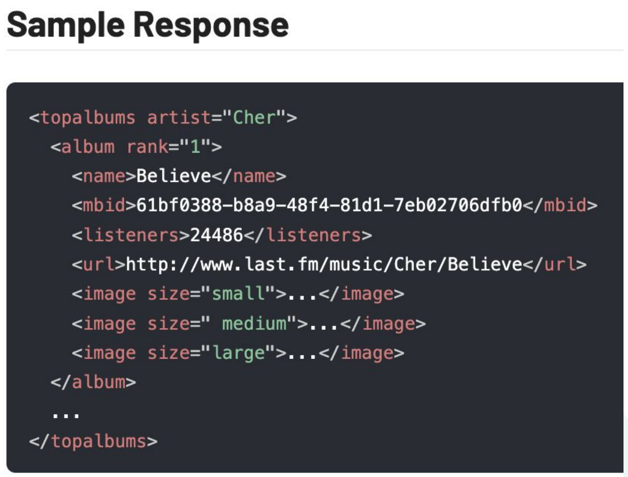
- Final URL/Endpoint:

```
http://ws.audioscrobbler.com/2.0?method=artist.gettopalbums
&artist=AP+Dhillon&api_key={API_KEY}&format=json
```

Last.fm API Demo!

- **Question:** What is the best album by AP Dhillon? 🤔
- **Code:**

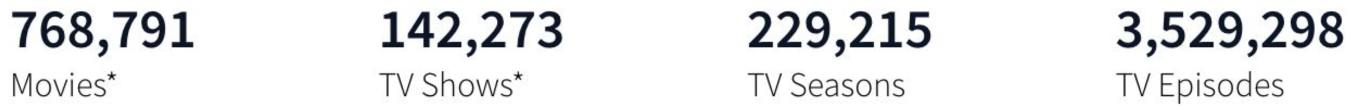
```
import requests

API_KEY = "<API-Key>"
BASE_URL = "http://ws.audioscrobbler.com/2.0"

def lastfm():
    url =
f"{BASE_URL}?method=artist.gettopalbums&artist=AP+Dhi
llon&api_key={API_KEY}&format=json"
    data = requests.get(url).json()
    album = data["topalbums"]["album"][0]
    print(album["name"])


lastfm()
```



Sample Response

```
<topalbums artist="Cher">
  <album rank="1">
    <name>Believe</name>
    <mbid>61bf0388-b8a9-48f4-81d1-7eb02706dfb0</mbid>
    <listeners>24486</listeners>
    <url>http://www.last.fm/music/Cher/Believe</url>
    <image size="small">...</image>
    <image size=" medium">...</image>
    <image size="large">...</image>
  </album>
  ...
</topalbums>
```

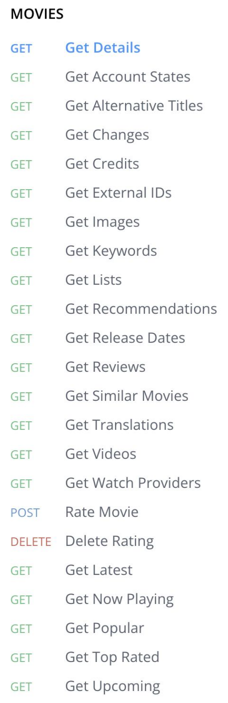
Example 3 – TMDB

- The Movie Database (**TMDB**) is a community built movie and TV database.
- 

  768,791 Movies*
  142,273 TV Shows*
  229,215 TV Seasons
  3,529,298 TV Episodes
- TMDB’s API provides data about
  - Movies
    - Plot
    - Cast
    - Posters
  - TV Shows
    - Seasons
    - Producers

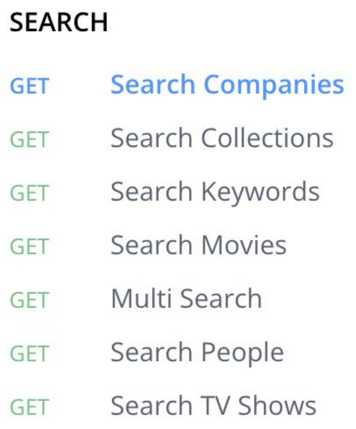
Example 3 – TMDB

- API documentation: https://developers.themoviedb.org/3
- A few methods/endpoints:



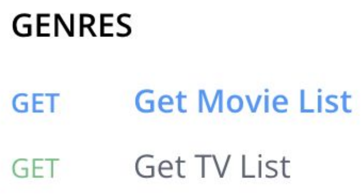
MOVIES

GET	Get Details
GET	Get Account States
GET	Get Alternative Titles
GET	Get Changes
GET	Get Credits
GET	Get External IDs
GET	Get Images
GET	Get Keywords
GET	Get Lists
GET	Get Recommendations
GET	Get Release Dates
GET	Get Reviews
GET	Get Similar Movies
GET	Get Translations
GET	Get Videos
GET	Get Watch Providers
POST	Rate Movie
DELETE	Delete Rating
GET	Get Latest
GET	Get Now Playing
GET	Get Popular
GET	Get Top Rated
GET	Get Upcoming



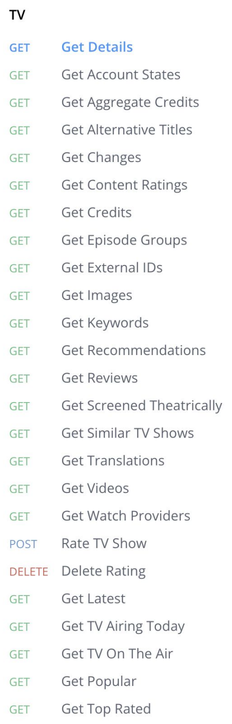
SEARCH

GET	Search Companies
GET	Search Collections
GET	Search Keywords
GET	Search Movies
GET	Multi Search
GET	Search People
GET	Search TV Shows



GENRES

GET	Get Movie List
GET	Get TV List



TV

GET	Get Details
GET	Get Account States
GET	Get Aggregate Credits
GET	Get Alternative Titles
GET	Get Changes
GET	Get Content Ratings
GET	Get Credits
GET	Get Episode Groups
GET	Get External IDs
GET	Get Images
GET	Get Keywords
GET	Get Recommendations
GET	Get Reviews
GET	Get Screened Theatrically
GET	Get Similar TV Shows
GET	Get Translations
GET	Get Videos
GET	Get Watch Providers
POST	Rate TV Show
DELETE	Delete Rating
GET	Get Latest
GET	Get TV Airing Today
GET	Get TV On The Air
GET	Get Popular
GET	Get Top Rated

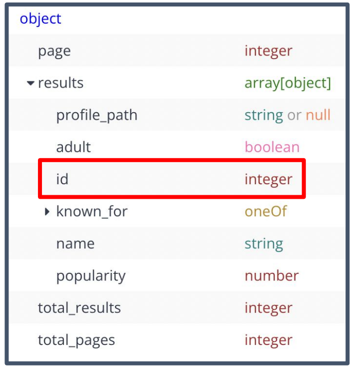
TMDB API Demo!

- **Question:** How many movies has Rajkumar Hirani directed? 🤔
- **Steps:**
  - Register an account on TMDB
  - Create an API Key (https://www.themoviedb.org/settings/api/request)
  - Search the documentation for the appropriate methods and endpoints
  - Query the endpoints to get our answer!
- API base URL: `https://api.themoviedb.org/3/`
- **2 API requests will be made**
  - Find the "ID" of Rajkumar Hirani -
    `GET /search/person&query=Rajkumar+Hirani`
  - Use this "ID" to find movies he has directed -
    `GET /person/{ID}/movie_credits`

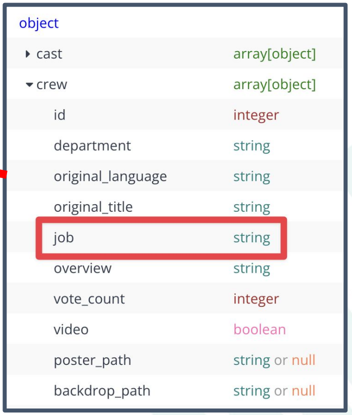
TMDB API Demo!

- **Question:** How many movies has Rajkumar Hirani directed? 🤔
- **Code:**

```python
import requests

API_KEY = "<API-KEY>"
BASE_URL = "https://api.themoviedb.org/3"

def example():

    # Get the person ID for Rajkumar Hirani
    url = f"{BASE_URL}/search/person?query=Rajkumar+Hirani&api_key={API_KEY}"
    data = requests.get(url).json()
    person = data["results"][0]
    id = person["id"]

    # Get the list of movies directed by Rajkumar Hirani
    url = f"{BASE_URL}/person/{id}/movie_credits?api_key={API_KEY}"
    data = requests.get(url).json()
    director_movies = []

    for movie in data["crew"]:
        if movie["job"] == "Director":
            director_movies.append(movie["title"])

    print(len(director_movies))
    print(director_movies)
```



object	
page	integer
▾ results	array[object]
profile_path	string or null
adult	boolean
id	integer
▸ known_for	oneOf
name	string
popularity	number
total_results	integer
total_pages	integer



object	
▸ cast	array[object]
▾ crew	array[object]
id	integer
department	string
original_language	string
original_title	string
job	string
overview	string
vote_count	integer
video	boolean
poster_path	string or null
backdrop_path	string or null

Finding good APIs

- Internet search is the natural choice
- We have shown many popular sites with APIs
- Long list given in: https://github.com/public-apis/public-apis
- Another one: https://pythonrepo.com/repo/public-apis-public-apis-python-third-party-apis-wrappers
- You have to spend time exploring the sites, reading the API documentation
- API based programming can be used to create mashups and interesting applications with data sourced from across the world

Summary

- Websites are now providing data through APIs (programs in the website process requests and return data)
- Programs from anywhere can call APIs and get the data - generally use HTTP GET request
- You can write applications that can use that data - programs have to call the APIs, get the response, and then process the response
- Hugely powerful - whole world can provide data for your programs; there are thousands of websites providing interesting data
-
- Advanced: uses of APIs - POST etc. ; publish APIs

Extra Slides, Examples

These are some more old examples (have not tried it currently - so check them)

Extras

- Pagination - APIs use pagination when there is a lot of data
    - In API request, you can request a page number, as well as size of data you want
- Rate limiting - to prevent servers from attacks, most API providers will have a rate limit - how many calls you can make / time
    - You will get an error if you exceed this
    - Check the limits on the API you want to use

Example 3 – Wolfram Alpha API

- Wolfram Alpha is an expert mathematical system with natural language processing capabilities, and a fact engine to provide real world data.

- In this short demo, we will interact with Wolfram Alpha's Short Answers API.
    - It returns short textual answers for queries
    - The API returns a single plain text result
    - It is implemented in the REST protocol using HTTP GET requests
    - You can read more about the API here

Wolfram Alpha and WikiData

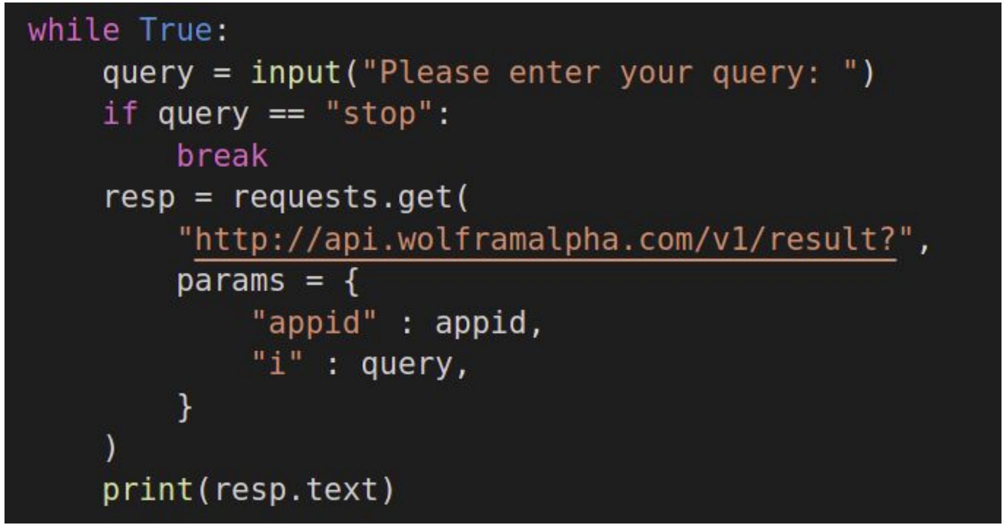
The code and explanation for the following demos on Wolfram Alpha and WikiData is available as a Notebook at: https://colab.research.google.com/drive/1URV5khFQ4Od4uSb586OjUVXwJ3PAaW04?usp=sharing

Example 3 – Wolfram Alpha API

- We need to obtain an APP-ID for authentication to make GET requests to the API
- This can be obtained by creating an account on Wolfram: Create Account
- After creating the account, click on "Get an App-ID", and set the variable appid as the obtained app-id
- The base URL for queries is: http://api.wolframalpha.com/v1/result
- Parameters are:
  - appid we obtained in the previous step
  - "i": The query we want the answer for

Example 3 – Wolfram Alpha API

- Now, we create a loop, in which we will ask the user to enter a query, and pass the query as-is to the Wolfram System.
- To send the query to Wolfram, we use the requests library in Python. More detail on requests can be found by executing help(requests) or visiting requests documentation



```
while True:
    query = input("Please enter your query: ")
    if query == "stop":
        break
    resp = requests.get(
        "http://api.wolframalpha.com/v1/result?",
        params = {
            "appid" : appid,
            "i" : query,
        }
    )
    print(resp.text)
```

Example 3 – Wolfram Alpha API

- What is the size of the moon

  about 1079.6 miles
- What is the integral of x^2 + sin(x)

  x^3/3 - cos(x)
- What is the root of x^2 -1

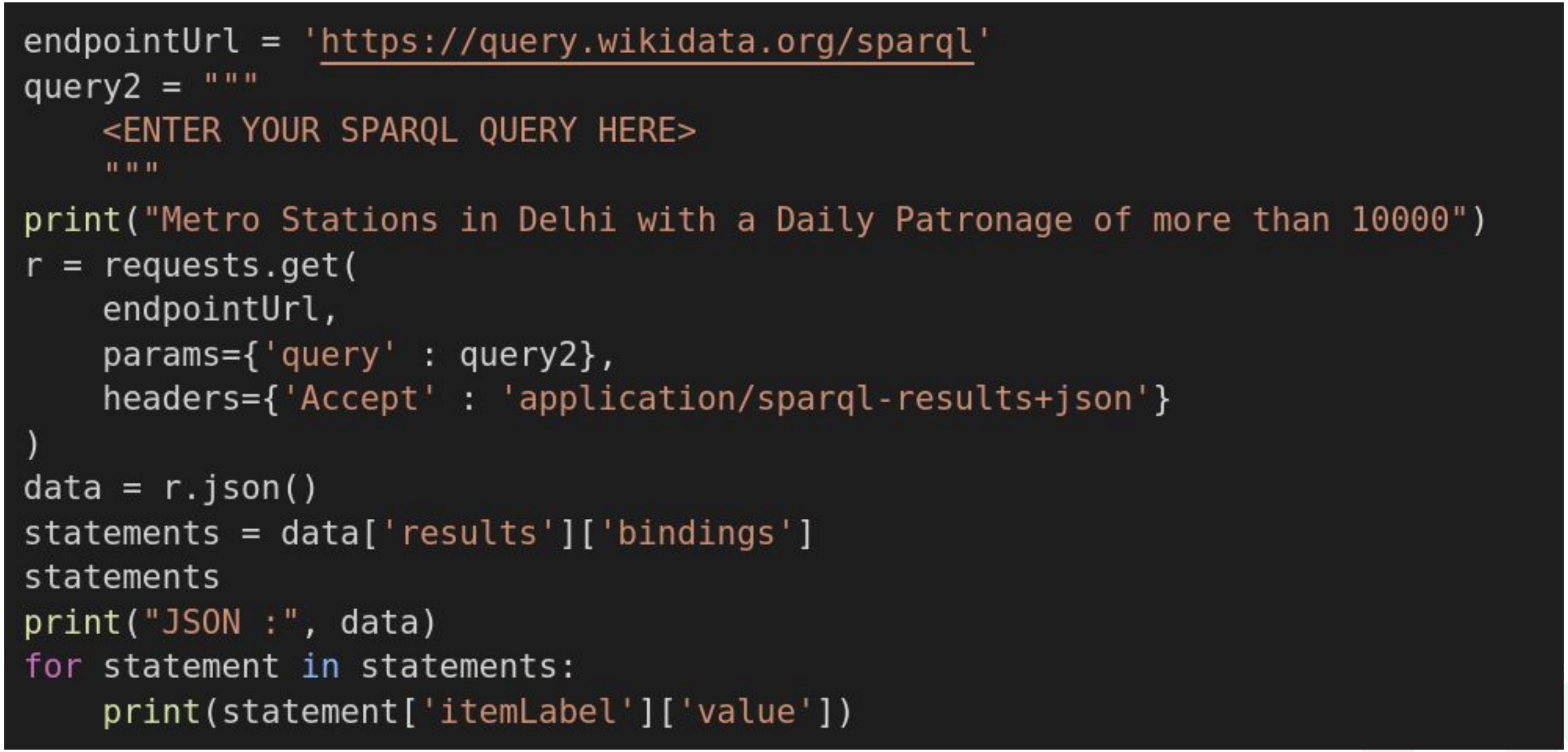
  sqrt(x^2) - 1
- Who is the father of Albert Einstein

  Hermann Einstein

Example 4 – WikiData API

- Wikipedia is mostly text in human languages, making it hard to work with for computer programs.
- WikiData is is a database built using the knowledge stored in Wikipedia, which can be fetched using Query Languages like SQL!
- We will see a small demonstration of the capability of WikiData
- It can be used to query almost the entirety of Wikipedia Knowledge base.

Example 4 – WikiData API



```python
endpointUrl = 'https://query.wikidata.org/sparql'
query2 = """
    <ENTER YOUR SPARQL QUERY HERE>
    """
print("Metro Stations in Delhi with a Daily Patronage of more than 10000")
r = requests.get(
    endpointUrl,
    params={'query' : query2},
    headers={'Accept' : 'application/sparql-results+json'}
)
data = r.json()
statements = data['results']['bindings']
statements
print("JSON :", data)
for statement in statements:
    print(statement['itemLabel']['value'])
```

Example 5 – Currency API

- Currency API is an open source project that provides exchange rates for 150+ currencies, including common cryptocurrencies.
- The API doesn't have a rate limit and is updated daily.
- You get three different type of requests:
    - List of all supported currencies
    - Exchange rate of a currency with all other currencies
    - Specific exchange rates between two currencies

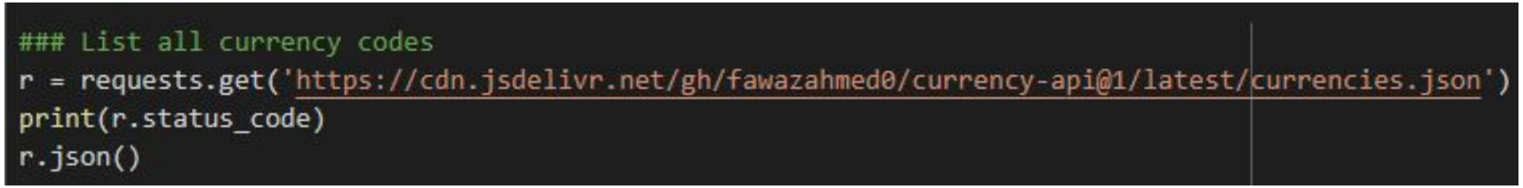
Example 5 – Currency API

- Sample queries that you can run
  - Get the currency list with INR as base currency
  - Get the currency list with BTC as base currency
  - Get the currency value for EUR to JPY
- Can be used for different kinds of applications like mobile and web apps
- URL: https://github.com/fawazahmed0/currency-api

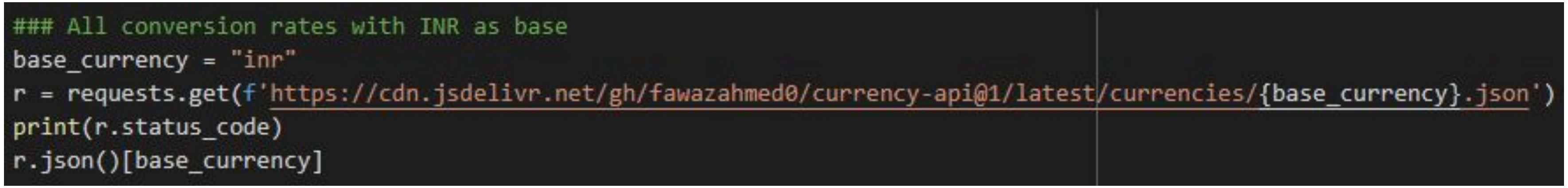
Example 5 – Currency API

- Query 1: https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies.json
- Query 2: https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies/inr.json
- Query 3: https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies/btc.json
- Query 4: https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies/eur/jpy.json

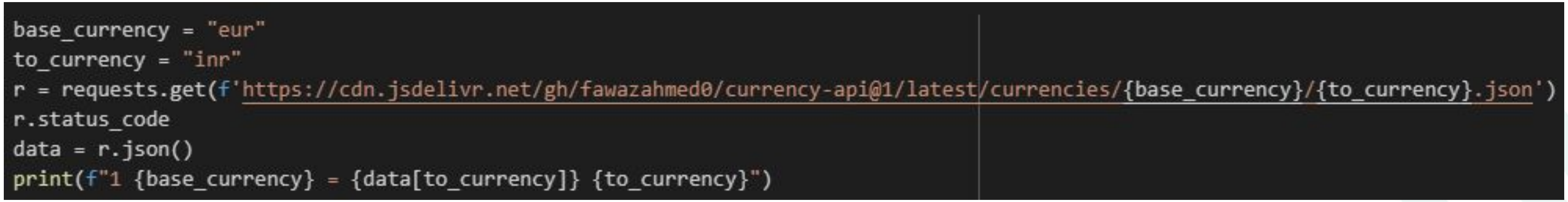
Example 5 – Currency API



```python
### List all currency codes
r = requests.get('https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies.json')
print(r.status_code)
r.json()
```



```python
### All conversion rates with INR as base
base_currency = "inr"
r = requests.get(f'https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies/{base_currency}.json')
print(r.status_code)
r.json()[base_currency]
```



```python
base_currency = "eur"
to_currency = "inr"
r = requests.get(f'https://cdn.jsdelivr.net/gh/fawazahmed0/currency-api@1/latest/currencies/{base_currency}/{to_currency}.json')
r.status_code
data = r.json()
print(f"1 {base_currency} = {data[to_currency]} {to_currency}")
```