1) GET DATA FROM API:

```
"""
07 LookupTournamentById
Script for esportsearning.com API: LookupTournamentById
http://api.esportsearnings.com/v0/LookupTournamentById?apikey=<apikey>&tournamentid=<tournamentid>
Documentation: https://www.esportsearnings.com/apidocs
Timeout required: 1 second (BAN)
Output format: json + csv

This is my first attempt at coding in Python. If you have any suggestions on how to improve this script, I would love to read them.

During the execution of the script, the following warnings are generated:
InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.esportsearnings.com'. Adding certificate verification is
strongly advised.
See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
warnings.warn("
This warning means that the site does not have a valid SSL certificate (i.e., https://).
Since we are not sending any personal information to the site, this is not a problem. The code will still execute despite the warning.
The warnings are therefore suppressed.

The code contains ANSI color formatting for terminal output, e.g., "\u001b[35m" (see print(f"\u001b[35m{url}\u001b[0m")),
so that it can be seen from a distance whether the request is proceeding normally (the download may take xx hours).

TO MAKE THIS SCRIPT WORK, YOU MUST HAVE A VALID API KEY. YOU CAN GET AN API KEY BY REGISTERING HERE:
https://www.esportsearnings.com/dev

"""
# load modules
import requests
import json
import csv
from datetime import datetime
import time
# for session creation and treatment of http status codes
```

```python
from requests.adapters import HTTPAdapter
from requests.packages.urllib3.util.retry import Retry
# disables the warning of an invalid SSL certificate:
from urllib3.exceptions import InsecureRequestWarning
from urllib3 import disable_warnings
disable_warnings(InsecureRequestWarning)

# setting global variables
apikey = 'your api key here' # YOU CAN GET AN API KEY BY REGISTERING HERE: https://www.esportsearnings.com/dev
tournamentid = 1000 # !!! ID You want to start on
numberofID = 3 # !!! # !!! number of IDs You want to process
file_name = f"LookupTournamentById_{tournamentid}-{tournamentid+numberofID-1}"
# for overview at the end of the script it saves the error messages that occurred during processing
errorCodes = []

# displays the start time and end time of the script execution to estimate the duration of this script
# to calculate the processing time (the result is the time in seconds):
time_start = time.time()
time_begin = datetime.fromtimestamp(time_start)
# display the time on the terminal in blue color (ansi code)
print(f"\u001b[36;1mTime at the beginning of the script: {time_begin}\u001b[0m")

# to estimate the approximate time you can arrive at a processed script:
# you can add something extra to the estimated processing time (correct according to real times)
estim_time_of_execution = numberofID * 1.7
print(f"\u001b[36;1mEstimated script processing time: {(estim_time_of_execution//60)//60} hours, {(estim_time_of_execution//60)%60}
minutes, {((estim_time_of_execution%60)%60):.2f} seconds\u001b[0m") # in blue color

# creating a session to handle HTTP exceptions
# If the API returns a code 502 (bad gateway) or 429 (too many requests) instead of the data,
# the "retry" tries to query the same data again with increasing delay (backoff_factor)
session = requests.Session()
retry = Retry(
    total=5,
    backoff_factor=0.5,
```

```
    status_forcelist=(429, 500, 502, 504))
adapter = HTTPAdapter(max_retries=retry)
session.mount("http://", adapter)
session.mount("https://", adapter)

# API data retrieval functions
def get_LookupTournamentById(tournamentid):
    # API requires min 1 second between queries, min 1 second between requests
    time.sleep(1)
    # without Verify=False the request will not be executed due to an invalid SSL certificate
    url = f'https://api.esportsearnings.com/v0/LookupTournamentById?apikey={apikey}&tournamentid={tournamentid}'
    response = session.get(url, verify=False)
     # on the terminal You can see which address You are currently at. For quick orientation, how far You are in the request, in pink
color
    print(f"\u001b[35m{url}\u001b[0m")

    # handling of Errors returned by the API (200 = success)
    if response.status_code != 200:
        # If the API reports an Error, we display it and also save it to the list, in red color
        print(f"\u001b[31;1mError: HTTP {response.status_code}\u001b[0m")
        errorCodes.append(f"tournamentid:{tournamentid}-error:{response.status_code}")
        data = [{"Error": f"HTTP {response.status_code}"}]

    # if the API returns "NULL"/none:
    elif not response.text:
        print(f"\u001b[31;1mError: Empty response\u001b[0m")
        errorCodes.append(f"TournamentId:{tournamentid}-Error:Empty response")
        data = [{"Error": "Empty response"}]
    else:
        try:
            data = response.json()
        except:
            print(f"\u001b[31;1mError: Invalid response: {url}\u001b[0m")
            print(response.status_code)
            print(response.text)
```

```
            data = [{"Error": "Invalid response"}]

    # Add a new key and ID value
    data.update({"tournamentid": tournamentid})

    return data

# saving data to a file
try:
    with open(f"{file_name}.jsonl", mode="w", encoding="utf-8") as file:
        for i in range(0, numberofID):
            # call the function with the parameter corresponding to the first ID where I want to start
            dataA = get_LookupTournamentById(tournamentid)
            tournamentid += 1  # ID increase by 1
            if not dataA:
                continue
            json.dump(dataA, file, ensure_ascii=False)
            file.write("\n") # separate lines
# the code will be executed even if the user interrupts its execution (which sometimes takes hours):
except KeyboardInterrupt:
    print("Interrupted by user")

# convert json file to csv:
def jsonl_to_csv():
    with open(f'{file_name}.csv', 'w', newline='', encoding="utf-8") as fw:
        with open(f'{file_name}.jsonl', 'r', encoding="utf8") as fr:
            first_row = True
            for row in fr:
                data = json.loads(row)
                if first_row:
                    first_row = False
                    headers = list(data.keys()) + ["ErrorCode", "Error"]
                    writer = csv.DictWriter(fw, fieldnames=headers)
                    writer.writeheader()
                writer.writerow(data)
```

```
# call the jsonl_to_csv() function
jsonl_to_csv()

# Request time summary: (When running longer requests (many IDs) we needed to have some overview,
# when to go back to the computer to the downloaded file and run the next one)
# To calculate the processing time (the result is the time in seconds):
time_end = time.time()
time_over = datetime.fromtimestamp(time_end, tz=None)
print(f"\u001b[36;1mTime at start of request: {time_begin}\u001b[0m") # blue
print(f"\u001b[36;1mTime at end of   request: {time_over}\u001b[0m") # blue
print(f"\u001b[32;1mExpected request duration: {(estim_time_of_execution//60)//60} hodin, {(estim_time_of_execution//60)%60} minut,
{((estim_time_of_execution%60)%60):.4f} sekund.\u001b[0m") # blue
print(f"\u001b[32;1mActual request duration: {(((time_end - time_start)//60)//60)} hodin, {(((time_end - time_start)//60)%60)} minut,
{(((time_end - time_start)%60)%60):.4f} sekund.\u001b[0m") # green
print(f"\u001b[31;1mHTTP error codes: {errorCodes}\u001b[0m") # red
print(f"\u001b[33mThe name of the output file is: {file_name}\u001b[0m") # yellow
```

TERMINAL:

2) SCRAPE DATA FROM WEBSITE:

```
# From https://www.esportsearnings.com/ scrapes Date of Birth of individual players
## Most of the players have not Date of Birth filled.
# You need file with all the links, similar to my sample file "links_todo.csv"

# load modules
import requests
import csv
import time
from bs4 import BeautifulSoup
# for session creation and treatment of http status codes
from requests.adapters import HTTPAdapter
from requests.packages.urllib3.util.retry import Retry

## VARIABLES TO CHANGE:
input_file_name_links_todo = "links_todo.csv"
output_file_name_DatesOfBirth = "DatesOfBirth.csv"

# creating a session to handle HTTP exceptions
# If the API returns a code 502 (bad gateway) or 429 (too many requests) instead of the data,
# the "retry" tries to query the same data again with increasing delay (backoff_factor)
session = requests.Session()
retry = Retry(
    total=5,
    backoff_factor=0.5,
    status_forcelist=(429, 500, 502, 504))
adapter = HTTPAdapter(max_retries=retry)
session.mount("http://", adapter)
session.mount("https://", adapter)

## get links to scrape from file
links_todo = []
with open(f"{input_file_name_links_todo}", mode="r", encoding="utf-8") as fr:
    for rows in fr:
```

```
        rows1 = rows.strip()
        rows3 = rows1.split(',', 2)
        collumn = rows3[0].strip('"')
        if collumn != 'Url':
            links_todo.append(collumn)

print(f"Number of links to process: {len(links_todo)}")

dob_list = []
def get_DateOfBirth(url):
    time.sleep(1.0)

    response = session.get(url)
    # on the terminal I can see which address I am currently at for quick orientation, how far I am in the request, in pink
    print(f"\u001b[35m{url}-{i}\u001b[0m")

    if response.status_code != 200:
        print(f"\u001b[31;1mError: HTTP {response.status_code}\u001b[0m")
        data = [{"Error": f"HTTP {response.status_code}", "Url": url}]

    elif not response.text:
        data = [{"Error": "Empty response", "Url": url}]

    else:
        try:
            # Create BeautifulSoup object
            soup = BeautifulSoup(response.text, 'html.parser')

            # Find the element containing the date of birth
            dob_element = soup.find('div', string='Date of Birth:')

            # Pull the date of birth from the following element
            dob = dob_element.find_next_sibling('div').text
            dictionary = {}
            dictionary["Url"]= url
```

```
            dictionary["DateOfBirth"] = dob
            data = dictionary
        except:
            print(f"\u001b[31;1mError: Invalid response: {url}\u001b[0m")
            print(response.status_code)
            print(response.text)
            data = [{"Error": "Invalid response", "Url": url}]

    # Save the date of birth in the list
    dob_list.append(data)
    return dob_list

# saving data to a file
try:
    with open(f"{output_file_name_DatesOfBirth}", "w", encoding="utf-8", newline='') as fw:
        first_row = True
        for i in range(0,len(links_todo)):
            url = links_todo[i]
            DataA = get_DateOfBirth(url)
            if not DataA:
                continue
            for listA in DataA:
                if first_row:
                    first_row = False
                    headers = ["Url", "DateOfBirth", "Error"]
                    writer = csv.DictWriter(fw, fieldnames=headers)
                    writer.writeheader()
            writer.writerow(listA)
# the code will be executed even if the user interrupts its execution (which sometimes takes hours):
except KeyboardInterrupt:
    print("Interrupted by user")

print(f"\u001b[33mThe name of the output file is: {output_file_name_DatesOfBirth}\u001b[0m") # yellow
```

TERMINAL:

```
Number of links to process: 4
https://www.esportsearnings.com/players/2001-beam-iisakki-ahonen-0
https://www.esportsearnings.com/players/2005-smiley1983-zhang-zhe-1
https://www.esportsearnings.com/players/3304-n0tail-johan-sundstein-2
https://www.esportsearnings.com/players/14671-miracle-amer-al-barkawi-3
The name of the output file is: DatesOfBirth.csv
PS C:\Users\iveta\Desktop\GitHub\python\SCRAPE>
```