21/01/2024

Telecommunication Software

Report 4

Jehanne Sarrazin

230AEM053

For this whole practical exercise, you can find my work in the GitHub repository: https://github.com/Niennaaa/TelecommunicationSoftware

Example 1 to 5:

For this report, you can find my work on the example 1 to 5 and comments in the Annex page 4. You can also find it in the associated PDF file inside the same folder along with the corresponding notebook containing the python code.

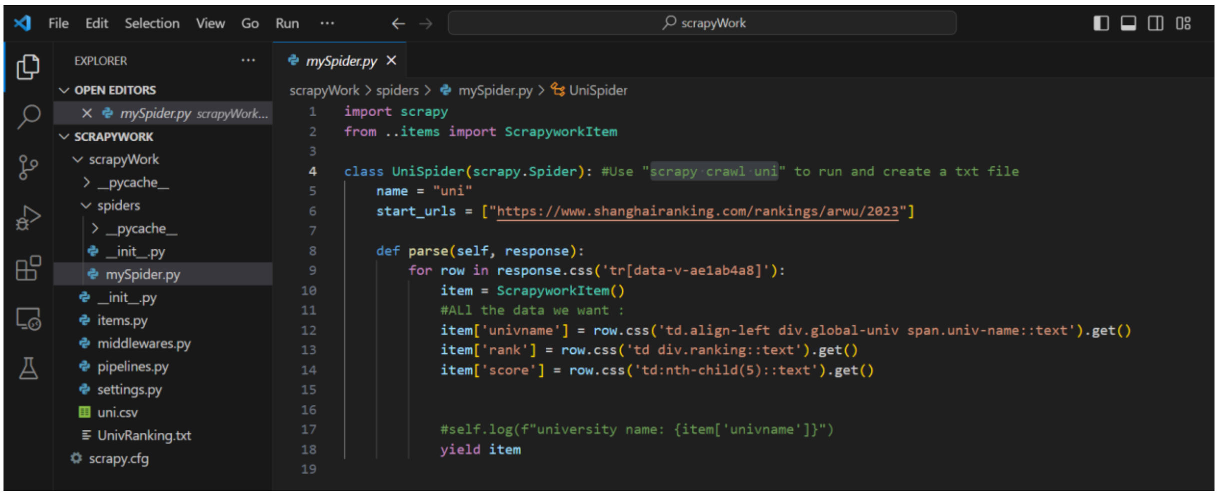
Example 6:

For the Example 6, I chose to complete the University Ranking work. You can find the python project for it in the folder "scrapyWork" and run it by opening it in VSCode and running in its terminal:

cd scrapyWork
scrapy crawl uni

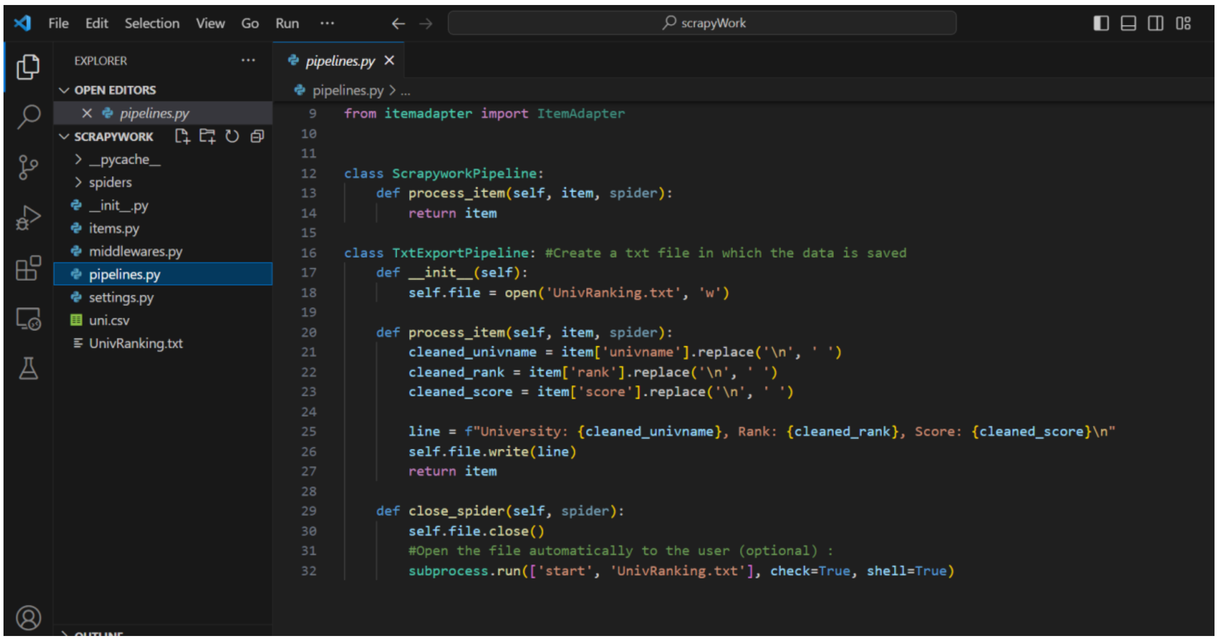
You will see that it will import the data from the university ranking and log it into a text file that will be automatically opened at the end of the run.

Here is the spider used:



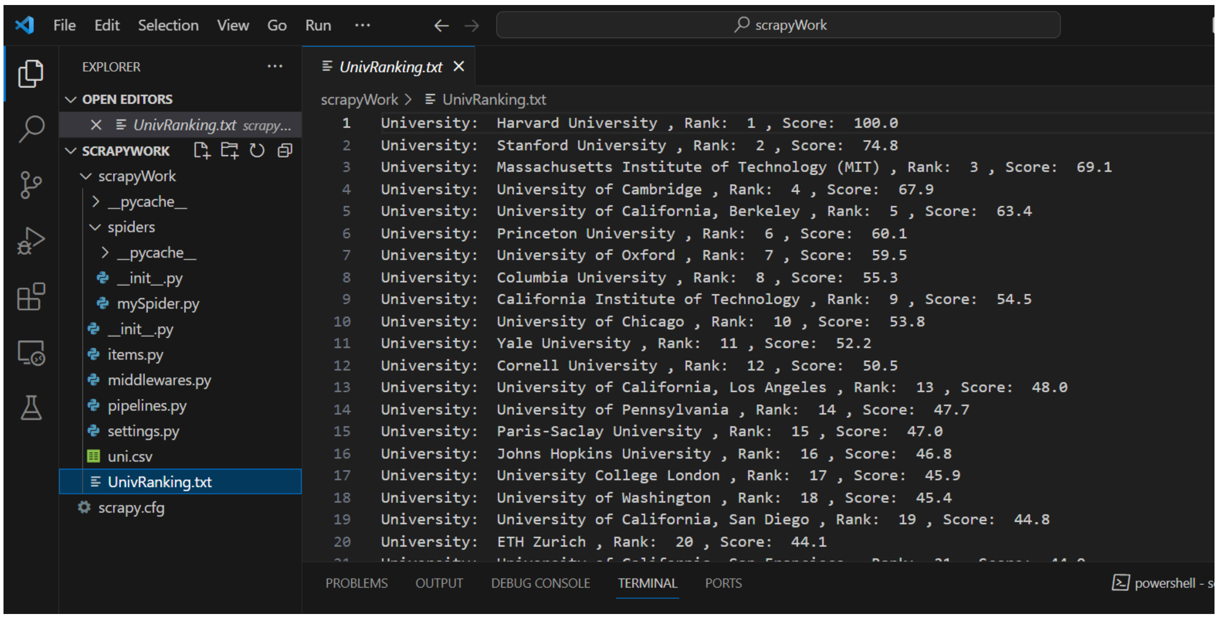
```python
import scrapy
from ..items import ScrapyworkItem

class UniSpider(scrapy.Spider): #Use "scrapy crawl uni" to run and create a txt file
    name = "uni"
    start_urls = ["https://www.shanghairanking.com/rankings/arwu/2023"]

    def parse(self, response):
        for row in response.css('tr[data-v-ae1ab4a8]'):
            item = ScrapyworkItem()
            #ALl the data we want :
            item['univname'] = row.css('td.align-left div.global-univ span.univ-name::text').get()
            item['rank'] = row.css('td div.ranking::text').get()
            item['score'] = row.css('td:nth-child(5)::text').get()


            #self.log(f"university name: {item['univname']}")
            yield item

```

The pipeline.py file used to saved the result as a txt document:



```python
from itemadapter import ItemAdapter


class ScrapyworkPipeline:
    def process_item(self, item, spider):
        return item

class TxtExportPipeline: #Create a txt file in which the data is saved
    def __init__(self):
        self.file = open('UnivRanking.txt', 'w')

    def process_item(self, item, spider):
        cleaned_univname = item['univname'].replace('\n', ' ')
        cleaned_rank = item['rank'].replace('\n', ' ')
        cleaned_score = item['score'].replace('\n', ' ')

        line = f"University: {cleaned_univname}, Rank: {cleaned_rank}, Score: {cleaned_score}\n"
        self.file.write(line)
        return item

    def close_spider(self, spider):
        self.file.close()
        #Open the file automatically to the user (optional) :
        subprocess.run(['start', 'UnivRanking.txt'], check=True, shell=True)
```

And the result:



```
University:  Harvard University , Rank:  1 , Score:  100.0
University:  Stanford University , Rank:  2 , Score:  74.8
University:  Massachusetts Institute of Technology (MIT) , Rank:  3 , Score:  69.1
University:  University of Cambridge , Rank:  4 , Score:  67.9
University:  University of California, Berkeley , Rank:  5 , Score:  63.4
University:  Princeton University , Rank:  6 , Score:  60.1
University:  University of Oxford , Rank:  7 , Score:  59.5
University:  Columbia University , Rank:  8 , Score:  55.3
University:  California Institute of Technology , Rank:  9 , Score:  54.5
University:  University of Chicago , Rank:  10 , Score:  53.8
University:  Yale University , Rank:  11 , Score:  52.2
University:  Cornell University , Rank:  12 , Score:  50.5
University:  University of California, Los Angeles , Rank:  13 , Score:  48.0
University:  University of Pennsylvania , Rank:  14 , Score:  47.7
University:  Paris-Saclay University , Rank:  15 , Score:  47.0
University:  Johns Hopkins University , Rank:  16 , Score:  46.8
University:  University College London , Rank:  17 , Score:  45.9
University:  University of Washington , Rank:  18 , Score:  45.4
University:  University of California, San Diego , Rank:  19 , Score:  44.8
University:  ETH Zurich , Rank:  20 , Score:  44.1
```

Annex:

In [2]:
```
#Exemple 1
#Test requests Python library, including seven types of methods
#and 13 parameters to control access

"""
From the class :
7 METHODS
requests.request() constructs a request that supports the basic methods of the foll
requests.get() The main method for obtaining HTML pages corresponding to HTTP GET
requests.head() The method to obtain the header information of HTML pages, correspo
requests.post() Method for submitting POST requests to HTML pages, corresponding to
requests.put() The method of submitting a PUT request to an HTML page, correspondin
requests.patch() Submit a partial modification request to an HTML page, correspondi
requests.delete() Submit a delete request to the HTML page, corresponding to HTTP D

13 PARAMETERS
params: dictionary or byte sequence, added to the url as a parameter
data: dictionary, byte sequence or file object, as the content of the Request
JSON : data in JSON format, as the content of Request
headers: dictionary, HTTP custom headers
cookies: dictionary or CookieJar, cookies in Request
auth: tuple, support HTTP authentication function
files: dictionary type, transfer files
timeout: Set the timeout time in seconds
proxies: dictionary type, set the access proxy server, you can add login authentica
allow_redirects: True/False, the default is True, redirect switch
stream: True/False, the default is True, get the content immediately download switc
verify: True/False, the default is True, verify the SSL certificate switch
cert: local SSL certificate path

"""

#The 7 methods :
import requests
r1 = requests.request('GET',"http://httpbin.org/get")
r2 = requests.get("http://httpbin.org/get")
r3 = requests.head("http://httpbin.org/get")
r4 = requests.post("http://httpbin.org/post")
r5 = requests.put("http://httpbin.org/put")
r6 = requests.patch("http://httpbin.org/patch")
r7 = requests.delete("http://httpbin.org/delete")
r = [r1, r2, r3, r4, r5, r6, r7]
equi = ["REQUEST", "GET", "HEAD", "POST", "PUT", "PATCH", "DELETE"]
n = 0
for rr in r:
    try :
        rr.raise_for_status()
        rr.encoding = rr.apparent_encoding
        print("\n" + equi[n]+ " : ")
        print(rr.text[0:600])
        print(rr.status_code)
        print(rr.encoding)
        print("\n")
        n=n+1
    except :
        print("Not this one :" + str(rr))
        n=n+1

url = "http://httpbin.org/get"
kv = {'key1':'value1', 'key2':'value2'}
body = "content"

#The 13 parameters :
r1 = requests.get(url, params = kv)
r2 = requests.get(url, data = kv)
r3 = requests.get(url, json = body)
r4 = requests.get(url, headers = kv)
r5 = requests.get(url, cookies = kv)
r6 = requests.get(url, auth = ('user', 'pass'))
r7 = requests.get(url, files = kv)
r8 = requests.get(url, timeout = 100)
r9 = requests.get(url, proxies = kv)
r10 = requests.get(url, allow_redirects = False)
r11 = requests.get(url, stream = False)
r12 = requests.get(url, verify = False)
#r13 = requests.get(url, cert = ('client.cert', 'client.key'))
r = [r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12]
equi = ["PARAMS", "DATA", "JSON", "HEADERS", "COOKIES", "AUTH", \
        "FILES", "TIMEOUT", "PROXIES", "ALLOW_REDIRECTS", "STREAM", "VERIFY", "CERT
n = 0
for rr in r:
    try :
        rr.raise_for_status()
        rr.encoding = rr.apparent_encoding
        print("\n" + equi[n]+ " : ")
        print(rr.text[0:600])
        print(rr.status_code)
        print(rr.encoding)
        print("\n")
        n=n+1
    except :
        print("Not this one :" + str(rr))
        n=n+1
```

```
REQUEST :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2bf-26d8457035e0f3753fa23f80"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



GET :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c0-635c5c5f71813c467bf73a62"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



HEAD :

200
None



POST :
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Length": "0",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c0-71303cbf6ddaccc03662c892"
  },
  "json": null,
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/post"
}

200
ascii



PUT :
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Length": "0",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c0-04decb382eaac969034cf598"
  },
  "json": null,
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/put"
}

200
ascii



PATCH :
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Length": "0",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c0-0f97d29c25ed5fc8000d6e28"
  },
  "json": null,
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/patch"
}
```

```
200
ascii



DELETE :
{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Length": "0",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c1-435216581805d3ec32235b5a"
  },
  "json": null,
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/delete"
}

200
ascii



PARAMS :
{
  "args": {
    "key1": "value1",
    "key2": "value2"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c1-6a18bea64395b5be20591cc0"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get?key1=value1&key2=value2"
}

200
ascii



DATA :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Length": "23",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c2-1d6d22e338d0d2fc33819d82"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



JSON :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Length": "9",
    "Content-Type": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c2-19434aa4427922147ad84019"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



HEADERS :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "Key1": "value1",
    "Key2": "value2",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c2-1147fd977327c3f2398f43f7"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii
```

```
COOKIES :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Cookie": "key1=value1; key2=value2",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c2-619f18db286258f97e5cd17c"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



AUTH :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Authorization": "Basic dXNlcjpwYXNz",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c3-27908f461e708adc7120354c"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



FILES :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Content-Length": "254",
    "Content-Type": "multipart/form-data; boundary=73e3b624c4822ba47fadb8cd6ffa8a5
5",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c3-4f491ab302d79f146745264a"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



TIMEOUT :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c3-059f9b027ff33a4b17eb1b82"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



PROXIES :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c3-796c0e0a4927d8cb41a0448a"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



ALLOW_REDIRECTS :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c4-5a67cea645ca368b323ab8f9"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



STREAM :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c4-2bc46d3632957bfd753a0f33"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii



VERIFY :
{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate, br",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-65a7b2c5-6e699a1829675a4d60c46f85"
  },
  "origin": "85.254.221.245",
  "url": "http://httpbin.org/get"
}

200
ascii
```

In [3]:
```
#Exemple 2
#Search engine keyword submission interface with requests python library.

import requests
keyword = "meme"
try :
    kv = {'q':keyword} #I followed the same logic as the one we've seen in the lect
    r = requests.get("http://www.bing.com/search", params = kv)
    print(r.request.url)
    r.raise_for_status()
    print(len(r.text))
except :
    print("Abnormal detected")

print(r.status_code)
print(r.encoding)
```

```
http://www.bing.com/search?q=meme
81755
200
utf-8
```

In [4]:
```
#Exemple 3
#  Image crawling.
import requests
import os
url = "https://i.imgflip.com/2ghngj.jpg"
root = "C:/Users/Jehanne/Downloads/"
path = root+url.split("/")[-1]

#I followed the same logic as seen in class
if not os.path.exists(root):
    os.nkdir(root)
if not os.path.exists(path):#Get image
    r = requests.get(url)
    with open(path, "wb") as f :
        f.write(r.content) #Save it
        f.close()
        print("File saved")
else :
    print("File already exist")
```

```
File already exist
```

In [14]:
```
#Exemple 4
#University ranking print

"""

#Input: University ranking URL link
Link: https://www.shanghairanking.com/rankings/arwu/2023

#Output: Screen output of university ranking information (rank, #university name, t

#Technical route: requests-bs4 or requests-lxml

#Step 1: Obtain university ranking webpage content from the web
#Step 2: Extract the information in the web page content into a suitable data struc
#Step 3: Use data structures to display and output results

"""

import requests
from bs4 import BeautifulSoup
import bs4
import re
import pandas as pd

#We will output the reult as a pandas dataframe
header = ["Rank", "University", "Score"]
result = pd.DataFrame(columns = header)
r = requests.get("https://www.shanghairanking.com/rankings/arwu/2023")
demo = r.text
soup = BeautifulSoup(demo, 'html.parser')

for tr in soup.find('tbody').children:
    if isinstance(tr, bs4.element.Tag):
        tds = tr('td')
        #First I get the data :
        rank =tds[0].find_all(string = re.compile(''))[-1].replace("\n","")
        university = tds[1].find_all(string = re.compile(''))[-1]
        score = tds[4].find_all(string = re.compile(''))[-1].replace("\n","")
        #Then I add it to my result
        result.loc[len(result)] = [rank, university, score]
print(result)
```

```
   Rank                                 University  Score
0     1                         Harvard University  100.0
1     2                        Stanford University   74.8
2     3  Massachusetts Institute of Technology (MIT)   69.1
3     4                    University of Cambridge   67.9
4     5         University of California, Berkeley   63.4
5     6                       Princeton University   60.1
6     7                       University of Oxford   59.5
7     8                        Columbia University   55.3
8     9          California Institute of Technology   54.5
9    10                      University of Chicago   53.8
10   11                            Yale University   52.2
11   12                         Cornell University   50.5
12   13       University of California, Los Angeles   48.0
13   14                University of Pennsylvania   47.7
14   15                    Paris-Saclay University   47.0
15   16                   Johns Hopkins University   46.8
16   17                  University College London   45.9
17   18                   University of Washington   45.4
18   19        University of California, San Diego   44.8
19   20                                ETH Zurich   44.1
20   21    University of California, San Francisco   44.0
21   22                        Tsinghua University   40.3
22   23                    Imperial College London   39.9
23   24                     University of Toronto   39.7
24   25         Washington University in St. Louis   39.0
25   26           University of Michigan-Ann Arbor   37.8
26   27                    The University of Tokyo   37.7
27   28                        New York University   37.3
28   29                         Peking University   36.7
29   30                    Northwestern University   35.8
```

```
In [21]: #Example 5:
         #Crawling of goods web pages and printing relevant numbers, goods names, and prices

         import requests
```

```python
from bs4 import BeautifulSoup
import bs4
import re
import pandas as pd

#I created this function to work with aliexpress.com
#It was a challenge because the place of the price changes based on the reductions
keyword = "plushie"
url = "https://www.aliexpress.com/w/wholesale-"+keyword+".html?spm=a2g0o.productlis


try :
    r = requests.get(url)
    print(r.request.url)
    soup = BeautifulSoup(r.text, 'html.parser')
    prettified= soup.prettify()

except :
    print("Abnormal detected")

soup0 = soup.find_all(class_="multi--content--11nFIBL")

alllist = []
index = 0
name = []
prices = []
rating = []
headers = ["Name", "Price", "Rating"]
results = pd.DataFrame(columns =headers)
for div in soup0:
    #I get all the data i need :
    alllist.append(div.get_text(separator=';', strip=True).split(";"))
    name.append(alllist[index][0])

    price = alllist[index][2]+alllist[index][3]+alllist[index][4]+alllist[index][5]
    prices.append(price)

    if len(alllist[index][-3])>3:
        rating.append(None)
    else :
        rating.append(alllist[index][-3])

    index+=1


    #And add it to my result

for i in range(len(name)):
    results.loc[i] = [name[i], prices[i], rating[i]]
print(results)
```

```
https://www.aliexpress.com/w/wholesale-plushie.html?spm=a2g0o.productlist.search.0
                                               Name   Price Rating
0  Bat Plush Toy manta Kawaii Animal Creative Plu...   €0.63    93%
1  18cm FNAF Stuffed Plush Toys Freddy Fazbear Be...   €1.27    92%
2  Original Sanrio Plushies Hello Kitty Cinnamonr...   €3.08    75%
3  Skzoo Plush Toys 20cm Stray Kids Plush Wolf Ch...   €0.46    95%
4  Gaint White Goose Plush Toy Super Soft Goose S...   €20.7   None
5  40-50cm Sanrio Pompompurin Stuffed Plush Toys ...  €23.54    68%
6  Genshin Game Anime Figure Doll Fluffy Cat Plus...   €1.31    87%
7  20cm We Bare Bears Cartoon Plush Toys Standing...    €0.7    95%
8  Cuddly Little Cat Plush Pendant Toy Fluffy Kit...   €0.46    94%
9  Angry Blob Seal Pillow Popular Soft Chubby 3D ...    €3.2    59%
```

In [ ]:

```
#Example 6:

#See in project report, this was done in VScode
```