Coursera

Title	Tags
Medium	Web
Type of scraper	HTML Parsing, HTTP Programming
Type of crawling	Generic
Depth and Scraping Strategy	DFS, Nested
Difficulty	Intermediate
Technology	HTTP Toolkit, Python, Selenium

Coursera is a U.S.-based massive open online course provider. Coursera works with universities and other organizations to offer online courses, certifications, and degrees in a variety of subjects. In 2021 it was estimated that about 150 universities offered more than 4,000 courses through Coursera.

Our task is to extract this data. We will achieve data acquisition by scraping the Coursera website. The following is a detailed process of how we acquire the raw material to implement the task.

Process

1. Visit the Coursera website.
2. Our first objective is to understand the structure of the website so that we can figure out a way to extract maximum data. After exploring for a while, we found out that Coursera's search gives an interesting result.

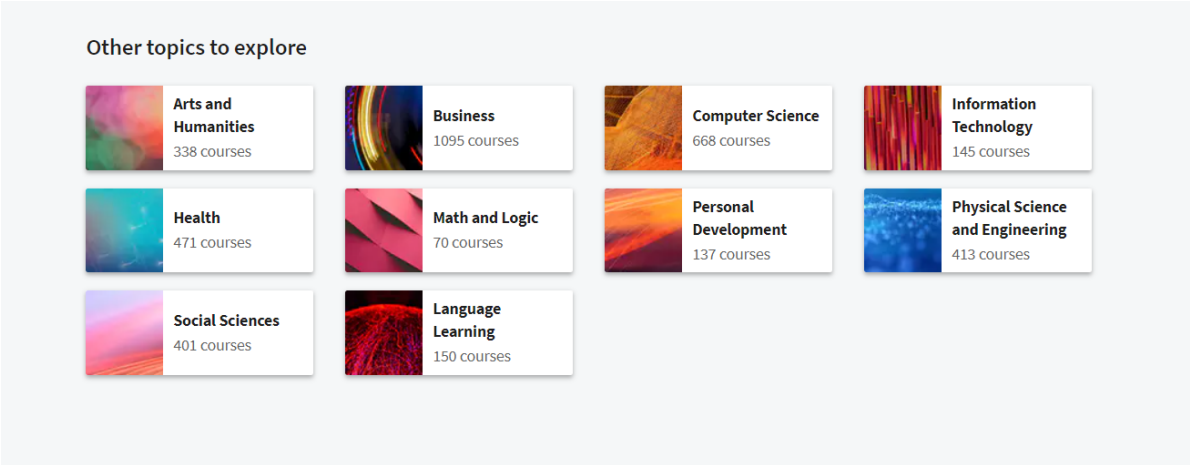
   https://www.coursera.org/search?query=python&page=1&index=prod_all_launched_products_term_optimization

Search is based on 2 parameters, viz., Query, and the Page number. If we can iterate through all the possible combinations, we would be able to scrap all the data.

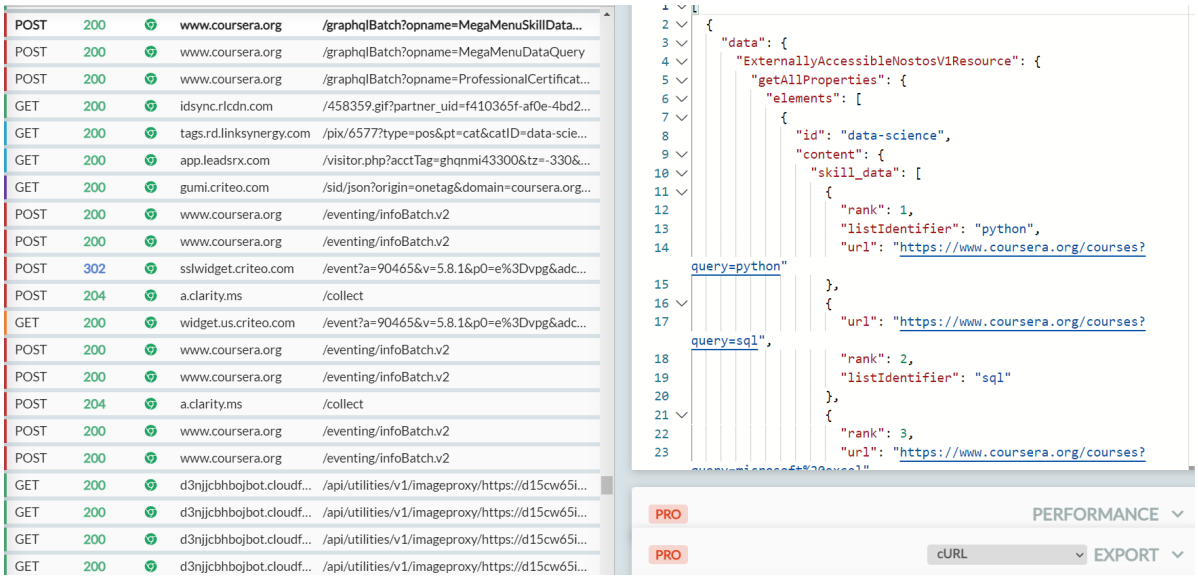
3. Our next step is to find the list of pre-defined queries. Here, we are making an assumption that Coursera has a pre-defined set of tags for courses that are shown using machine learning.

   We will use HTTP Toolkit to explore requests made by the Coursera.

   We found this section called explore other topics in Data Science. Here, we shall assume that these are the only available topics on Coursera.

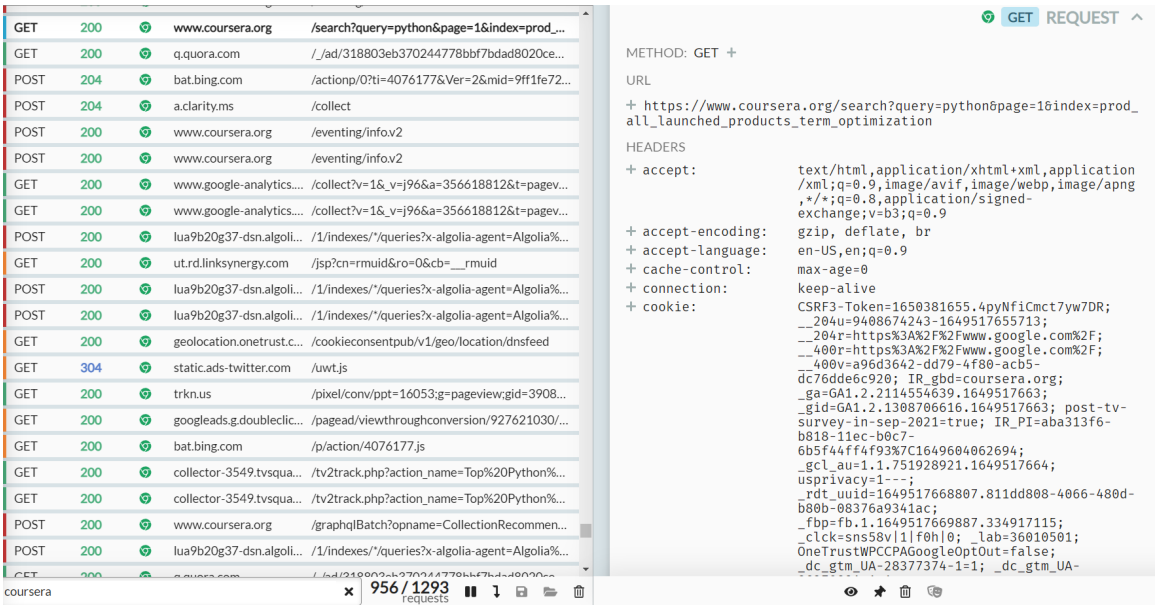


4. Our job is to hunt the API that retrieved the data regarding these topics. After exploration, we found this on HTTP Toolkit.



Here, we can clearly see that Data science is further divided into queries. We will download this JSON file and use it for further exploration.

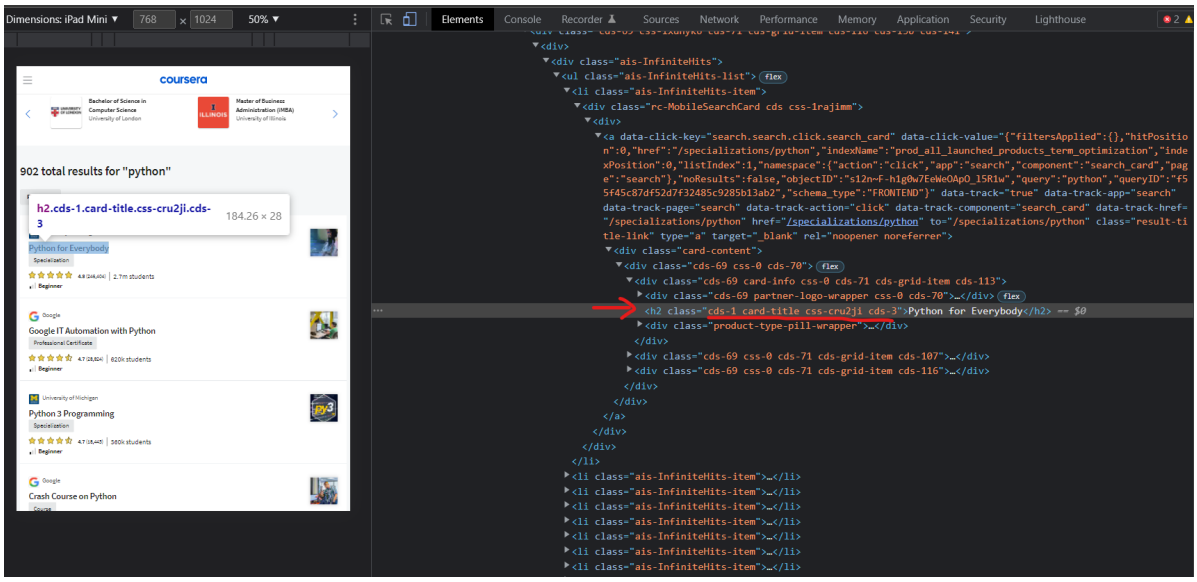
5. Our next task is to understand the structure of the search query page. We will make a search as per point 2. And search for the API in HTTP Toolkit. We will try to understand the response to the request.



Let’s now understand the data retrieved:

We can see that the response is the page itself in HTML format. Thus, our next task is to find text patterns in the HTML file to extract data. Our usage of HTTP Toolkit ends here.

6. For finding the text patterns in HTML response, we shall study the website page source using inspect elements.



Similarly, we shall use the inspect elements to understand the structure of other data points.

7. We shall use the web driver to jot down all “h2” tags with its specific class as shown in the inspect elements. Our final task would be to extract text from the tags.
8. Ultimately, we shall run through all the queries extracted earlier in step 4 and dump the data in an excel sheet using pandas.

Here we commence are raw material extraction process. We have:

- API URLs
- Course Queries

Before moving to the implementation, we shall understand the characteristics of our scraping approach for Coursera.

- As we are interacting with API URLs using HTTP requests, the data scraper is classified into HTTP Programming based scraper. Also, we are further setting a filter on data using text patterns in the link of the images. This sub-classifies the scraper into a Text pattern matcher.

- Our objective is to maximize the data scraped. So, we have set no filters on the crawler. Thus, we classify it as a generic crawling.
- First, we scraped sections (like Data Science) and later course queries (like python) were derived (we'll clarify in the implementation process). This makes the scraper explore 2 different layers in DFS fashion. Thus, the scraper is nested and follows the DFS fashion to scrape data.

Implementation

1. For this implementation, we shall write a python script to understand the JSON extracted in Process Point 4. We'll initiate by assigning the JSON to a variable.

```
data = [
  {
    "data": {
      "ExternallyAccessibleNostosV1Resource": {
        "getAllProperties": {
          "elements": [
            {
              "id": "data-science",
              "content": {
                "skill_data": [
                  {
                    "rank": 1,
                    "listIdentifier": "python",
                    "url": "https://www.coursera.org/courses?query=python"
                  },
```

2. We'll perform analysis to explore data further.

```
In [5]: data[0]['data']['ExternallyAccessibleNostosV1Resource']['getAllProperties']['elements'][0]
```

```
Out[5]: {'id': 'data-science',
         'content': {'skill_data': [{'rank': 1,
            'listIdentifier': 'python',
            'url': 'https://www.coursera.org/courses?query=python'},
           {'url': 'https://www.coursera.org/courses?query=sql',
            'rank': 2,
            'listIdentifier': 'sql'},
           {'rank': 3,
            'url': 'https://www.coursera.org/courses?query=microsoft%20excel',
            'listIdentifier': 'microsoft excel'},
           {'listIdentifier': 'excel',
            'rank': 4,
            'url': 'https://www.coursera.org/courses?query=excel'},
           {'rank': 5,
            'listIdentifier': 'machine learning',
            'url': 'https://www.coursera.org/courses?query=machine%20learning'},
           {'url': 'https://www.coursera.org/courses?query=data%20science',
            'listIdentifier': 'data science',
            'rank': 6},
```

3. While looping over the JSON data, we could extract all the Sections.

```
for i in range(0, 11):
    print(data[0]['data']['ExternallyAccessibleNostosV1Resource']['getAllProperties']['elements'][i]['id'])
```

```
data-science
business
computer-science
information-technology
language-learning
life-sciences
personal-development
physical-science-and-engineering
social-sciences
arts-and-humanities
math-and-logic
```

4. On further exploration and analysis of the JSON response, we could extract all the course queries.

```
In [10]: scrapUrl = []


for i in range(0, len(data[0]['data']['ExternallyAccessibleNostosV1Resource']['getAllProperties']['elements'])):
    for j in range(0, len(data[0]['data']['ExternallyAccessibleNostosV1Resource']['getAllProperties']['elements'][0]['content'][
        scrapUrl.append(data[0]['data']['ExternallyAccessibleNostosV1Resource']['getAllProperties']['elements'][i]['content']['s
```

```
In [11]: scrapUrl

Out[11]: ['https://www.coursera.org/courses?query=python',
          'https://www.coursera.org/courses?query=sql',
          'https://www.coursera.org/courses?query=microsoft%20excel',
          'https://www.coursera.org/courses?query=excel',
          'https://www.coursera.org/courses?query=machine%20learning',
          'https://www.coursera.org/courses?query=data%20science',
          'https://www.coursera.org/courses?query=data%20analytics',
          'https://www.coursera.org/courses?query=power%20bi',
          'https://www.coursera.org/courses?query=artificial%20intelligence',
          'https://www.coursera.org/courses?query=statistics',
          'https://www.coursera.org/courses?query=project%20management',
          'https://www.coursera.org/courses?query=microsoft%20excel',
          'https://www.coursera.org/courses?query=excel',
          'https://www.coursera.org/courses?query=blockchain',
          'https://www.coursera.org/courses?query=digital%20marketing',
          'https://www.coursera.org/courses?query=data%20analytics',
          'https://www.coursera.org/courses?query=power%20bi',
          'https://www.coursera.org/courses?query=design',
          'https://www.coursera.org/courses?query=communication%20skills',
```

```
In [119]: len(scrapUrl)

Out[119]: 110
```

5. We can conclude from Process Point 5 that the response from Coursera is in HTML format, i.e., on a GET request, the backend returns the whole page. Our job is to extract and match values to the parameters that we want to extract.

   We will use Beautiful Soup to perform pattern matcher for extraction. To get specific information, we need: 1) HTML Tag of the information 2) Class name to apply the filter.

   eg:-
   `course_title_tag = 'h2'`
   `course_title_class_name = 'cds-1 card-title css-cru2ji cds-3'`

   In the same fashion, we will assign information to other variables that we shall use later to extract information.

```
# course title
course_title_tag = 'h2'
course_title_class_name = 'cds-1 card-title css-cru2ji cds-3'

# course organization
course_organization_tag = 'span'
course_organization_class_name = 'cds-1 css-1cxz0bb cds-3'

# course certificate type
course_certificate_type_tag = 'span'
course_certificate_type_class_name = 'cds-1 css-yg35ph cds-3'

# course rating
course_rating_tag = 'span'
course_rating_class_name = 'ratings-text'

# course total ratings
course_total_ratings_tag = 'span'
course_total_ratings_class_name = 'ratings-count'

# course difficulty
course_difficulty_tag = 'span'
course_difficulty_class_name = 'cds-1 difficulty css-1vjdgz cds-3'

# course enrollment
course_enrollment_tag = 'span'
course_enrollment_class_name = 'enrollment-number'

# course URL
course_URL_tag = 'a'
course_URL_class_name = 'result-title-link'
```

6. Next, we shall apply our scrapping algorithm that would crawl and explore in the DFS fashion.

```
while query < 10:
    while check:
        driver.get('https://www.coursera.org/search?query=' + scrapUrl[query].split('=')[1] + '&page=' + str(pageNumber) + '&in
        time.sleep(2)

        page_source = driver.page_source
        scraper = BeautifulSoup(page_source, 'html.parser')

        if len(scraper.find_all(course_title_tag,course_title_class_name)) > 0 and pageNumber < 5:
            storeValues[0].append(__append_element__(scraper, course_title_tag, course_title_class_name))
            storeValues[1].append(__append_element__(scraper, course_organization_tag, course_organization_class_name))
            storeValues[2].append(__append_element__(scraper, course_certificate_type_tag, course_certificate_type_class_name))
            storeValues[3].append(__append_element__(scraper, course_rating_tag, course_rating_class_name))
            storeValues[4].append(__append_element__(scraper, course_total_ratings_tag, course_total_ratings_class_name))
            storeValues[5].append(__append_element__(scraper, course_difficulty_tag, course_difficulty_class_name))
            storeValues[6].append(__append_element__(scraper, course_enrollment_tag, course_enrollment_class_name))
            storeValues[7].append(__append_element_href__(scraper, course_URL_tag, course_URL_class_name))
            pageNumber = pageNumber + 1
```

7. Proceeding, we shall save all the entries to a CSV file using CSV write.

```
for i in range(0, len(storeValues[0][0])):
    data = [
        {'Title': storeValues[0][0][i],
        'Organization': storeValues[1][0][i],
        'Certificate': storeValues[2][0][i],
        'Rating': storeValues[3][0][i],
        'Total Ratings': storeValues[4][0][i],
        'Difficulty': storeValues[5][0][i],
        'Enrollment': storeValues[6][0][i],
        'Link': storeValues[7][0][i]}
    ]

    with open('Coursera.csv', 'a', encoding="utf-8") as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=field_names)
        writer.writerows(data)
```

8. Here is a snapshot of the scraped data.

	Title	Organization	Certificate	Rating	Total Ratings	Difficulty	Enrollment	Link
0	Python for Everybody	University of Michigan	Specialization	4.8	(246,447)	Beginner	2.7m	https://www.coursera.org/specializations/python
1	Google IT Automation with Python	Google	Professional Certificate	4.7	(28,848)	Beginner	620k	https://www.coursera.org/professional-certific...
2	Python 3 Programming	University of Michigan	Specialization	4.7	(18,452)	Beginner	360k	https://www.coursera.org/specializations/pytho...
3	Crash Course on Python	Google	Course	4.8	(23,412)	Beginner	540k	https://www.coursera.org/learn/python-crash-co...
4	Data Science Fundamentals with Python and SQL	IBM	Specialization	4.5	(44,400)	Beginner	590k	https://www.coursera.org/specializations/data-...
5	Applied Data Science with Python	University of Michigan	Specialization	4.5	(31,562)	Intermediate	810k	https://www.coursera.org/specializations/data-...
6	IBM Data Science	IBM	Professional Certificate	4.6	(89,364)	Beginner	1.1m	https://www.coursera.org/professional-certific...
7	Programming for Everybody (Getting Started wit...	University of Michigan	Course	4.8	(207,383)	Mixed	2.5m	https://www.coursera.org/learn/python
8	Python for Data Science, AI & Development	IBM	Course	4.6	(25,672)	Beginner	390k	https://www.coursera.org/learn/python-for-appl...
9	Introduction to Programming with Python and Java	University of Pennsylvania	Specialization	4.4	(699)	Beginner	39k	https://www.coursera.org/specializations/progr...

A mini version of the above dataset can be accessed here.

Thank you!