Scraping Task

Initialisation

Loading libraries and id/password from environment

```
from requests import session
import dotenv
import json
import os
from bs4 import BeautifulSoup
import time

dotenv.load_dotenv()
```

```
True
```

Functions

Create folder function

This function just creates a folder with the correct attributes for the files to be saved in

```
def create_img_folder(school, img_size):
    img_folder = "imgs"
    folder_name = "_".join([school["name"], str(school["year"]), img_size])
    folder_name = "/".join([img_folder, folder_name])
    if not os.path.exists(img_folder):
        os.mkdir(img_folder)
    if not os.path.exists(folder_name):
        os.mkdir(folder_name)
    return folder_name
```

Save data dict

This function iterates over the data dictionary, and chooses the size of the image (defaults to large)

```
def save_data_dict(session, school, data_dict, img_size="largeSizeUrl"):
    folder_name = create_img_folder(school, img_size)
    for page in data_dict["yearbookPages"]:
        save_page(session, page["pageNumber"], page[img_size], img_size, folder_name)
```

Save page

This function takes the url for each page, and saves its in the previously created folder

```
def save_page(session, pg, img_url, img_size, folder_name):
    file_name = "_".join(["pg", str(pg), "img", img_size.replace("Url", "")])
    file_name = ".".join([file_name, "jpeg"])
    file_name = "/".join([folder_name, file_name])
    img = session.get(img_url)
    if img.status_code == 200:
        with open(file_name, "wb") as f:
            f.write(img.content)
```

Login

This function logs us in by using the id/password, and then returns the saved session

```
def login(session, page_url):
    base_page = session.get(page_url)
    print(base_page.url, base_page)
    login_link_tag = [
        linktag
        for linktag in BeautifulSoup(base_page.content, features="lxml").findAll("a")
        if linktag.text == "Sign in"
    ]
    login_url = login_link_tag[0]["href"]
    login_page = session.get(login_url)
    print(login_page.url, login_page)
    login_soup = BeautifulSoup(login_page.content, features="lxml")
    csrf_token = login_soup.find("input", {"name": "_csrf"})["value"]
    form_data = {
        "_csrf": csrf_token,
        "emailOrRegId": os.environ["USER_ID"],
        "password": os.environ["USER_PW"],
```

```
        "rememberme": "yes",
        "successUrl": page_url,
        "g-recaptcha-response": "",
    }
    resp = session.post(login_url, data=form_data)
    print(resp.url, resp)
    return session
```

Main

Here, we set the page url, the headers, and then login. Post login, we use the api call from the same session. The call needs the current time in nanoseconds, the yearbook id, the page from where to start, and how many pages

```
page_url = "https://www.classmates.com/yearbooks/Alameda-High-School/4182755124?page=1"
headers = {
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) "
    "AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36",
}
cur_session = session()
cur_session.headers = headers
cur_session = login(cur_session, page_url)

# API CALL
cur_session.headers["version"] = "v2"
school = {"name": "Alameda High School", "year": "2010"}
page = 0
limit = 100
api_base = "https://www.classmates.com/node/api/yearbookPages/"
while True:
    api_params = {
        "limit": limit,
        "offset": page * limit,
        "yearbookId": "4182755124",
        "_": str(time.time_ns()),
    }
    api_resp = cur_session.get(api_base, params=api_params)
    if not api_resp.status_code == 200:
        break
    print(f"saving pages {(page*limit)+1} to {(page+1)*limit}")
```

```
        save_data_dict(cur_session, school, json.loads(api_resp.text))
        page += 1
```

```
https://www.classmates.com/yearbooks/Alameda-High-School/4182755124?page=1 <Response [200]>
https://secure.classmates.com/auth/login <Response [200]>
https://secure.classmates.com/auth/login <Response [200]>
saving pages 1 to 100
saving pages 101 to 200
saving pages 201 to 300
```

Notes

The website is quite tricky as it renders HTML only when you are logged out, but loads everything by JS when you login. It required a bit of back and forth to figure out the API call.
I started without logging in, and then you can easily acess the small version of images. Followed by this, I tried to login, and that led to rethinking my approach. I saw the API calls that the JS on loading was making, and tried making those with the logged in session. This failed initially, until I realised two important changes - firstly, the header needed the version to be added, and secondly, one of the params was just the current time in ns. Post that it worked without any issues!
Happy to talk about it further on a call!

repo at https://github.com/divij-sinha/miie-task/tree/main