AI VIETNAM
All-in-One Course

Project: Image Retrieval (TA Session)

Nguyễn Đăng Nhã

Year 2024

Objectives

Scraping URL of Images

Handle Request- Response

Execute HTML content

Interact with Web elements

Getting Images from URLs

Multi-Threading for Efficient Downloading

Polite Delay in Multiple Requests

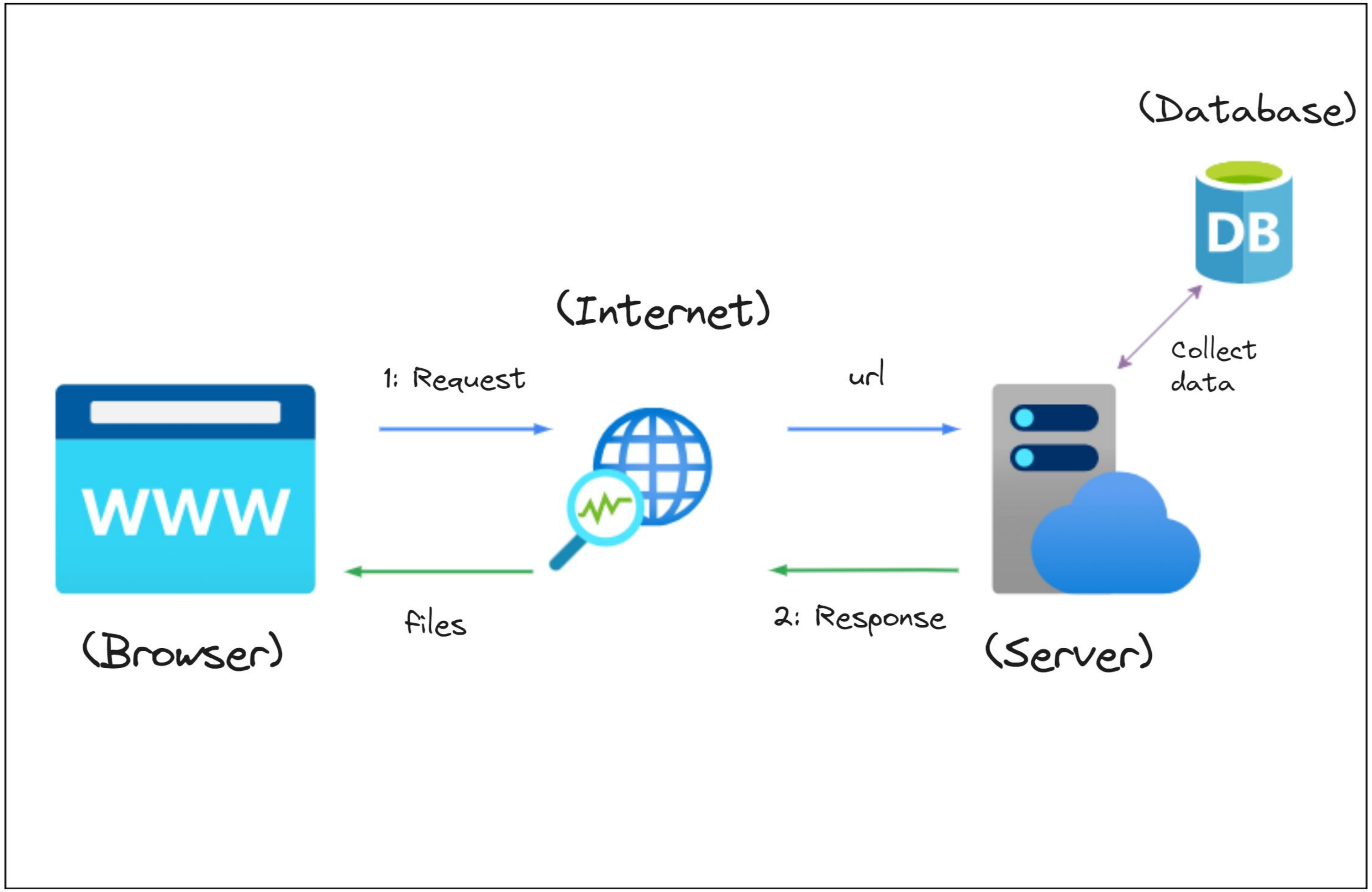
Process data to create a Dataset

Outline

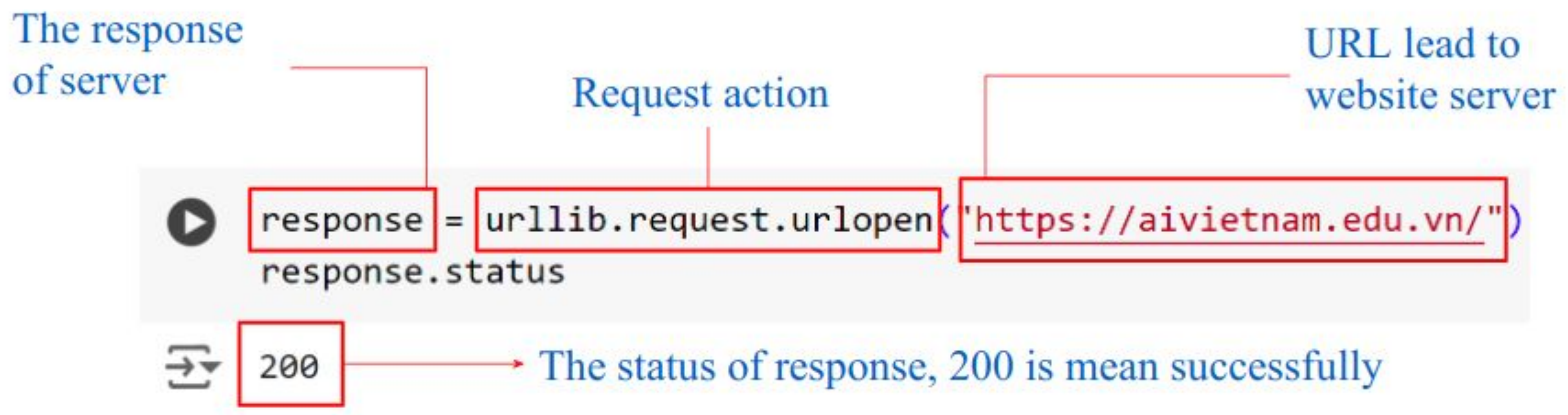
- How can we scrape image urls for single class?
- How can we scrape image urls for multiple classes?
- Downloading images via urls with multi-threading
- Why we need polite delay?
- Clean and Organize our final dataset

How can we scrape image urls for single class?

Client – Server Protocol

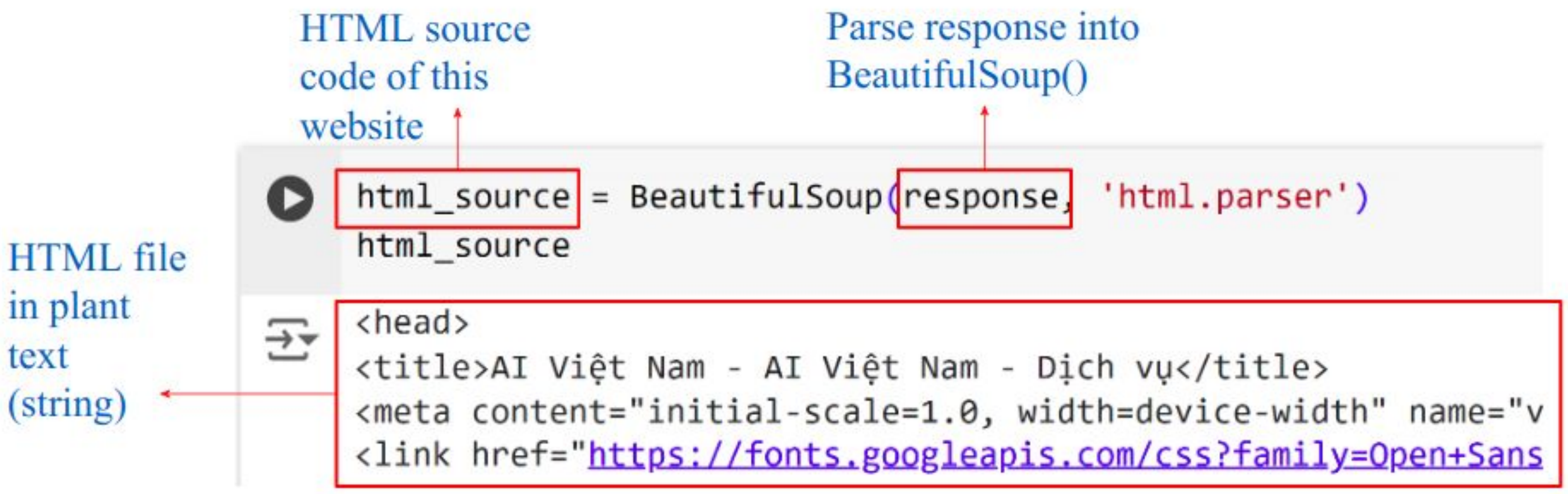


Getting HTML from URL


The response of server
Request action
URL lead to website server
The status of response, 200 is mean successfully

```
response = urllib.request.urlopen("https://aivietnam.edu.vn/")
response.status
```

```
200
```

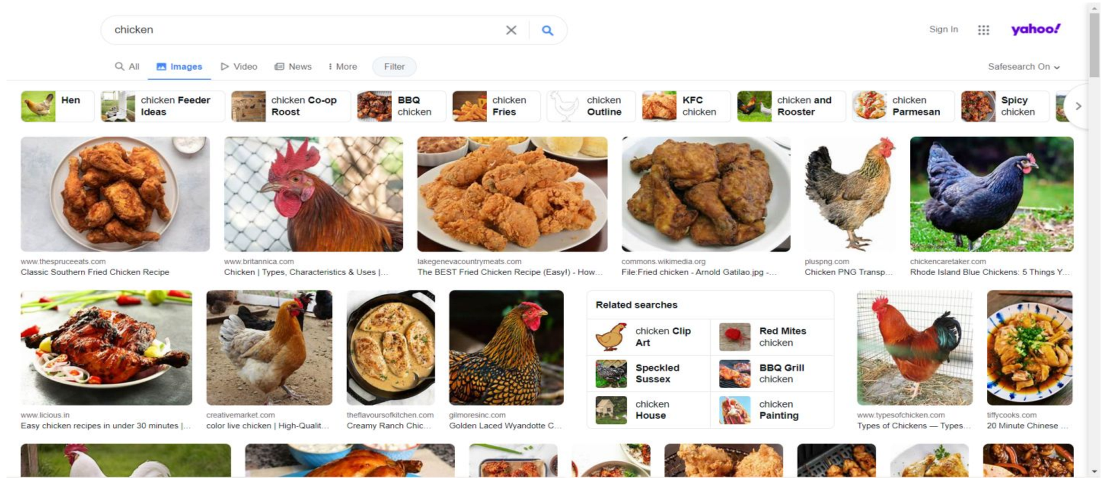

HTML source code of this website
Parse response into BeautifulSoup()
HTML file in plant text (string)

```
html_source = BeautifulSoup(response, 'html.parser')
html_source
```

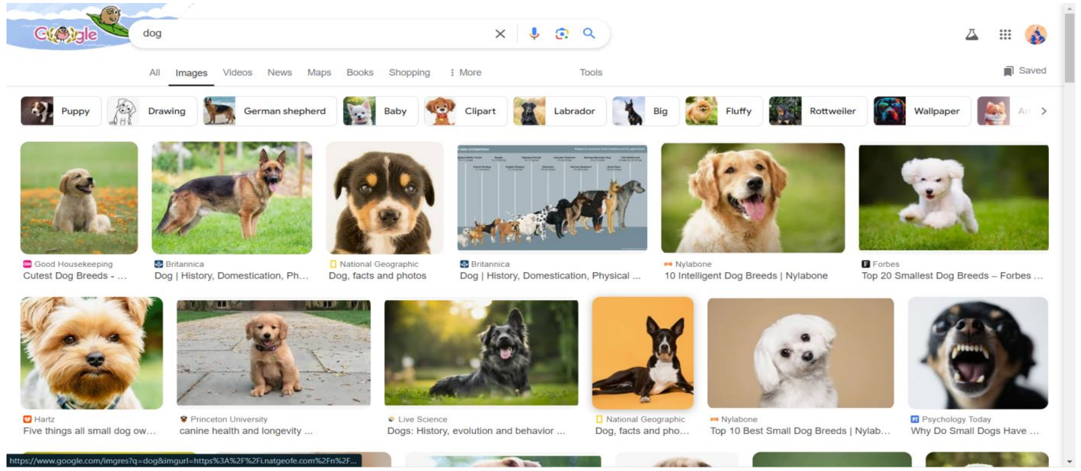
```
<head>
<title>AI Việt Nam - AI Việt Nam - Dịch vụ</title>
<meta content="initial-scale=1.0, width=device-width" name="v
<link href="https://fonts.googleapis.com/css?family=Open+Sans
```

AI VIETNAM
All-in-One Course

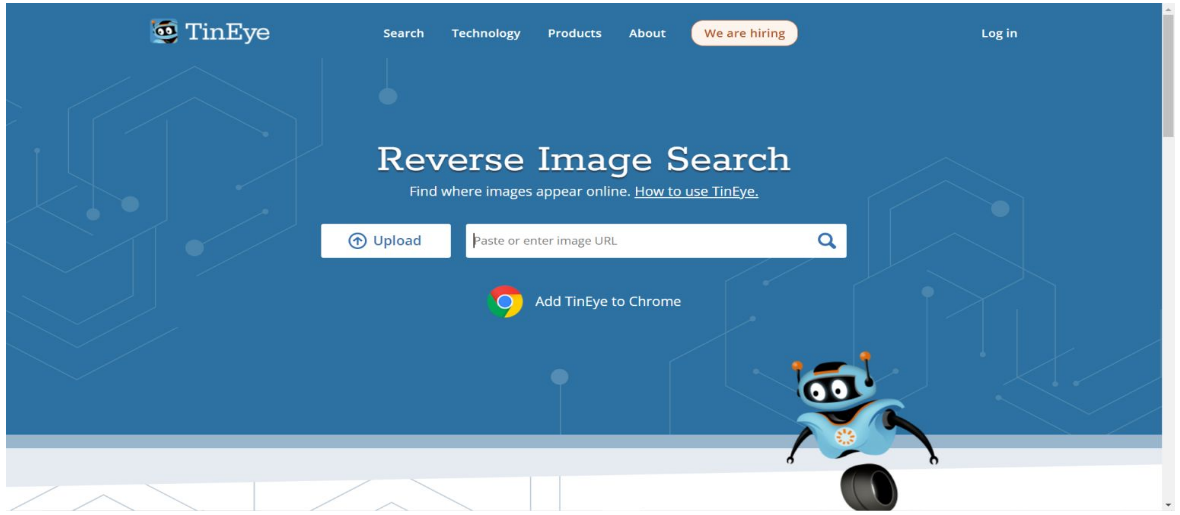
Image search engine



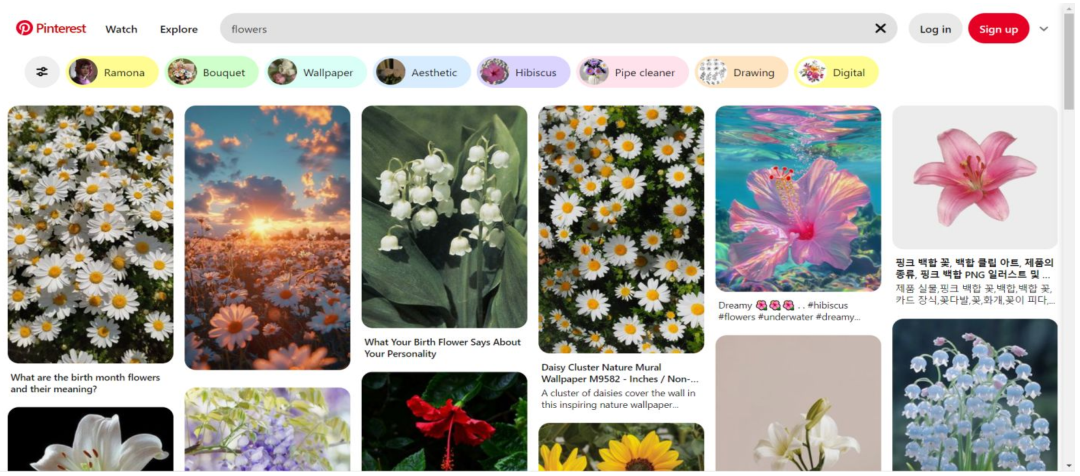
Yahoo Image Search



Google Image Search

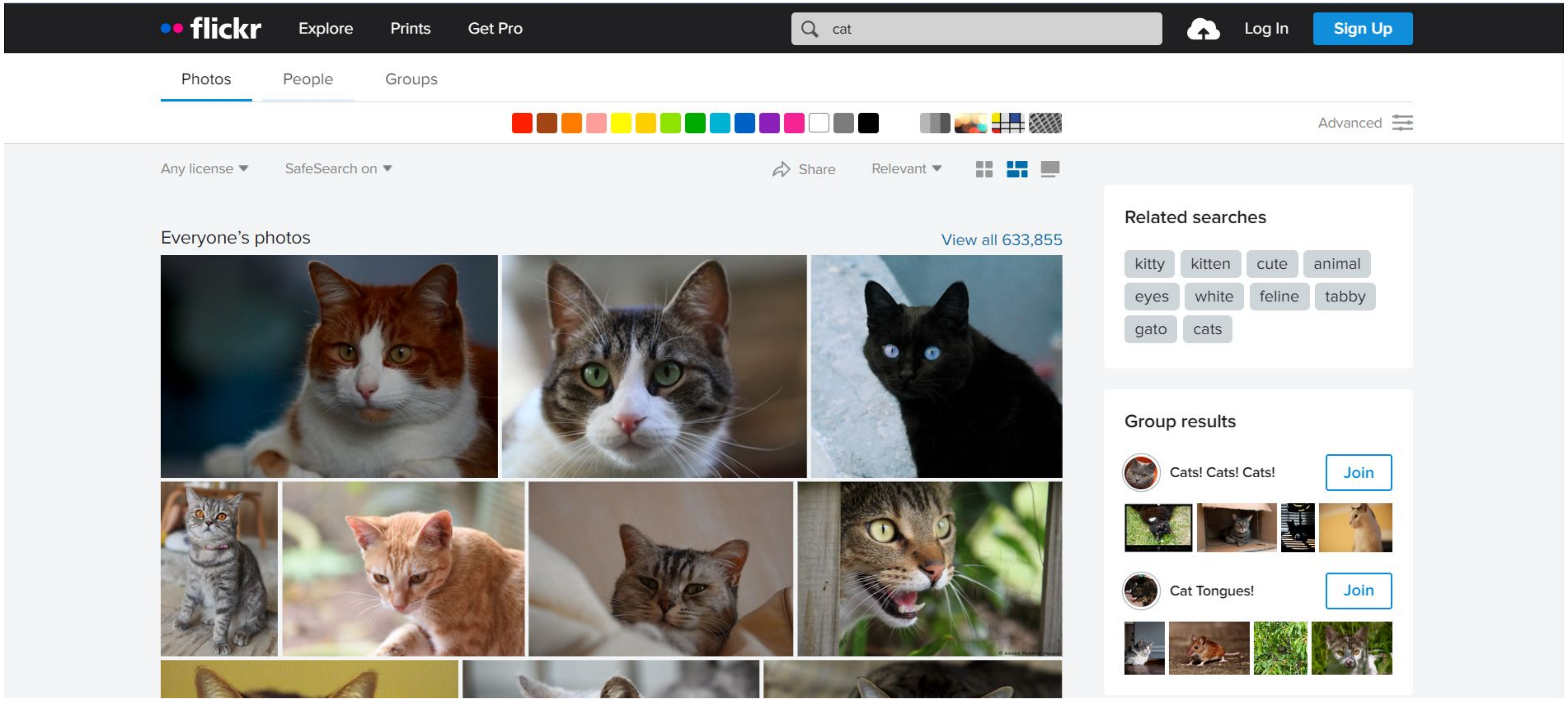


TinyEye



Pinterest

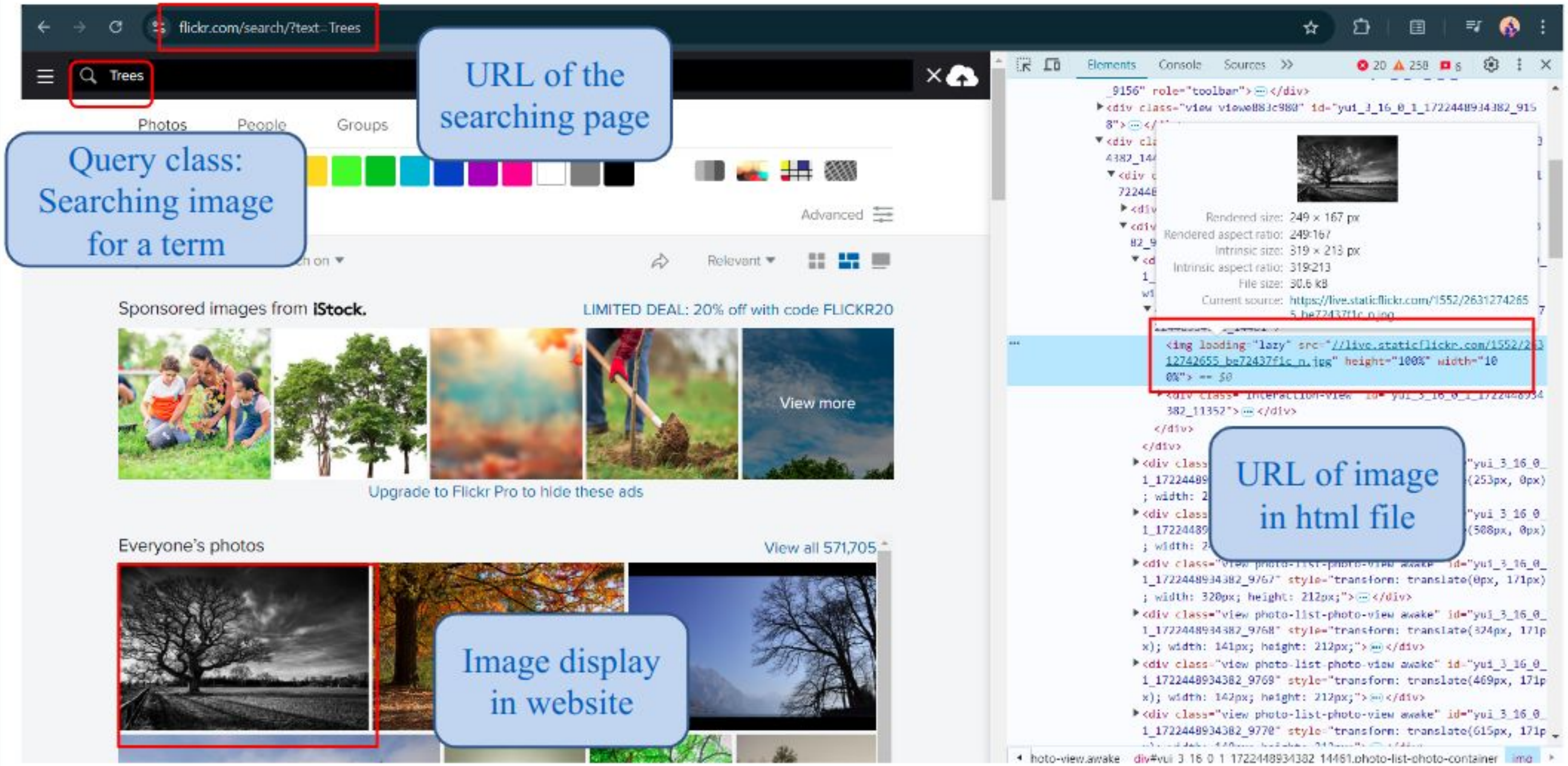
Image search engine



Flickr.com

AI VIETNAM
All-in-One Course

Outline of Flickr.com


flickr.com/search/?text=Trees
Trees
Photos
People
Groups
Advanced
Relevant
Sponsored images from iStock.
LIMITED DEAL: 20% off with code FLICKR20
View more
Upgrade to Flickr Pro to hide these ads
Everyone's photos
View all 571,705
Elements
Console
Sources

URL of the searching page

Query class: Searching image for a term

URL of image in html file

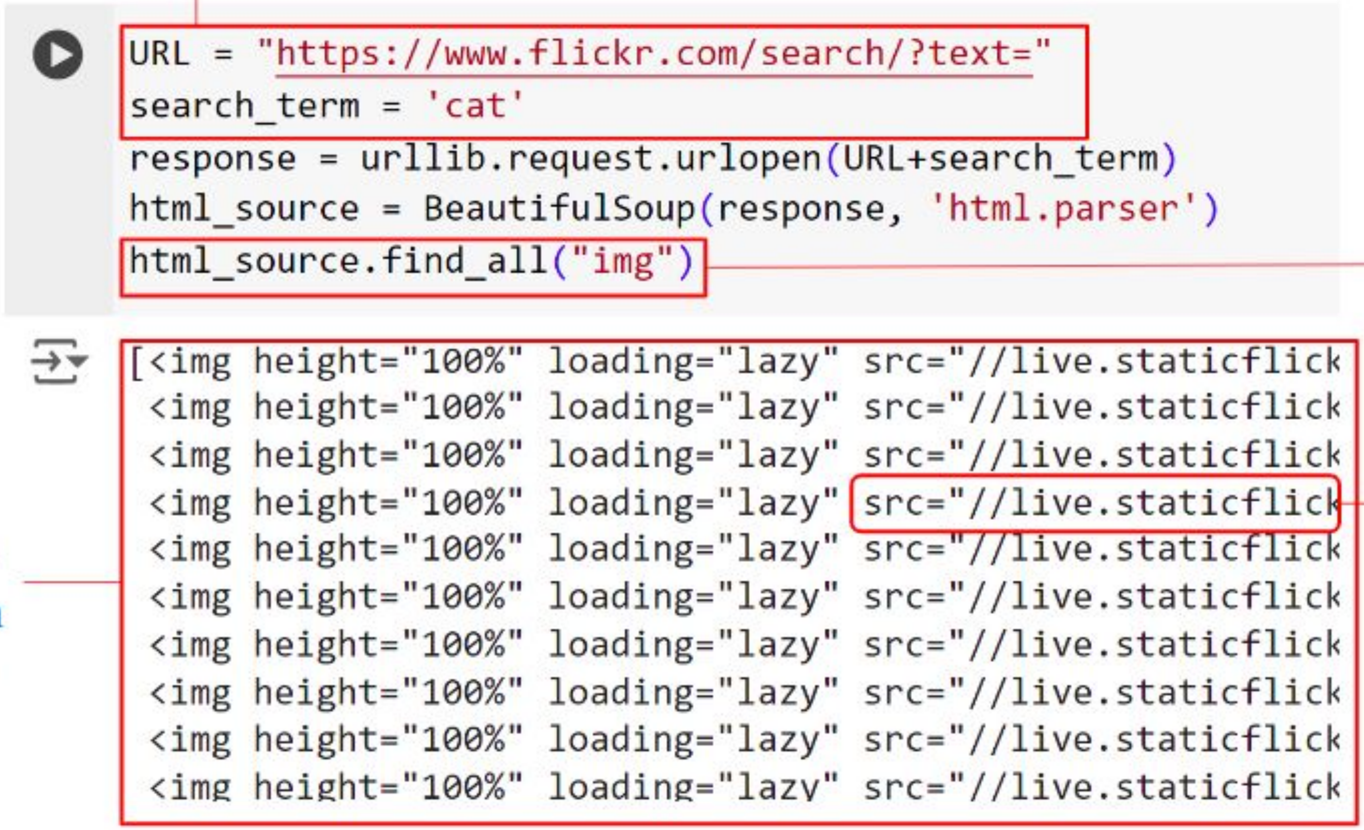
Image display in website

```
_9156" role="toolbar"> ⋯ </div>
▶<div class="view view08c980" id="yui_3_16_0_1_1722448934382_915
8"> ⋯ </
▼<div cl
4382_14
  ▼<div c
  72244
    ▶<div
    ▼<div
    82_9
      ▼<d
      1_
      w1
        ▼
          <img loading="lazy" src="//live.staticflickr.com/1552/263
          12742655_be72437f1c_n.jpg" height="100%" width="10
          0%"> == $0
        ▶<div class="interaction-view" id="yui_3_16_0_1_17224489334
        382_11352"> ⋯ </div>
      </div>
    </div>
  ▶<div class                                    "yui_3_16_0_
  1_17224489                                     (253px, 0px)
  ; width: 2
  ▶<div class                                    "yui_3_16_0_
  1_17224489                                     (508px, 0px)
  ; width: 2
  ▶<div class="view photo-list-photo-view awake" id="yui_3_16_0_
  1_1722448934382_9767" style="transform: translate(0px, 171px)
  ; width: 320px; height: 212px;"> ⋯ </div>
  ▶<div class="view photo-list-photo-view awake" id="yui_3_16_0_
  1_1722448934382_9768" style="transform: translate(324px, 171p
  x); width: 141px; height: 212px;"> ⋯ </div>
  ▶<div class="view photo-list-photo-view awake" id="yui_3_16_0_
  1_1722448934382_9769" style="transform: translate(469px, 171p
  x); width: 142px; height: 212px;"> ⋯ </div>
  ▶<div class="view photo-list-photo-view awake" id="yui_3_16_0_
  1_1722448934382_9770" style="transform: translate(615px, 171p
```

```
Rendered size: 249 × 167 px
Rendered aspect ratio: 249:167
Intrinsic size: 319 × 213 px
Intrinsic aspect ratio: 319:213
File size: 30.6 kB
Current source: https://live.staticflickr.com/1552/2631274265
5_be72437f1c_n.jpg
```

hoto-view.awake div#yui_3_16_0_1_1722448934382_14461.photo-list-photo-container img

Collect image's URL



```
URL = "https://www.flickr.com/search/?text="
search_term = 'cat'
response = urllib.request.urlopen(URL+search_term)
html_source = BeautifulSoup(response, 'html.parser')
html_source.find_all("img")
```

```
[<img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
 <img height="100%" loading="lazy" src="//live.staticflick
```

URL route directly to the searching page of class

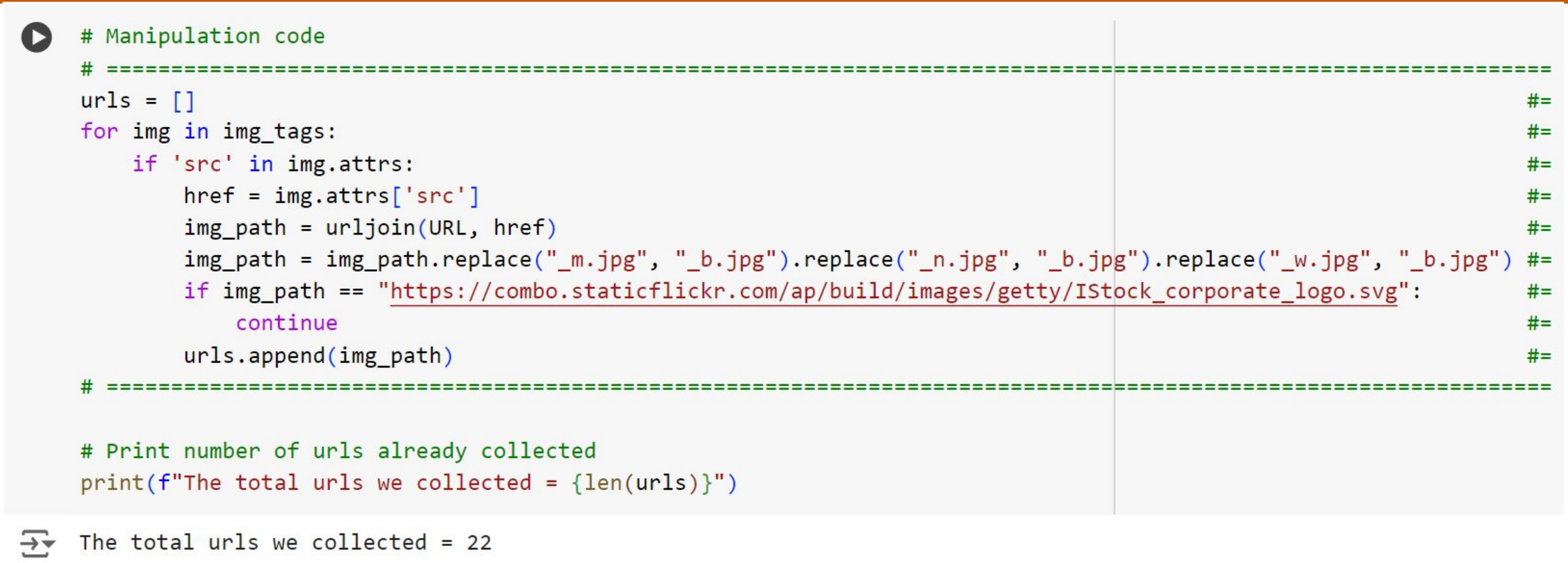
Extract the <img/> tag in html source

All img tags contain url of all images display in searching page

The image's url we can collect by **string manipulation**

Read detail manipulation in Colab (Brute force part)

Quiz time



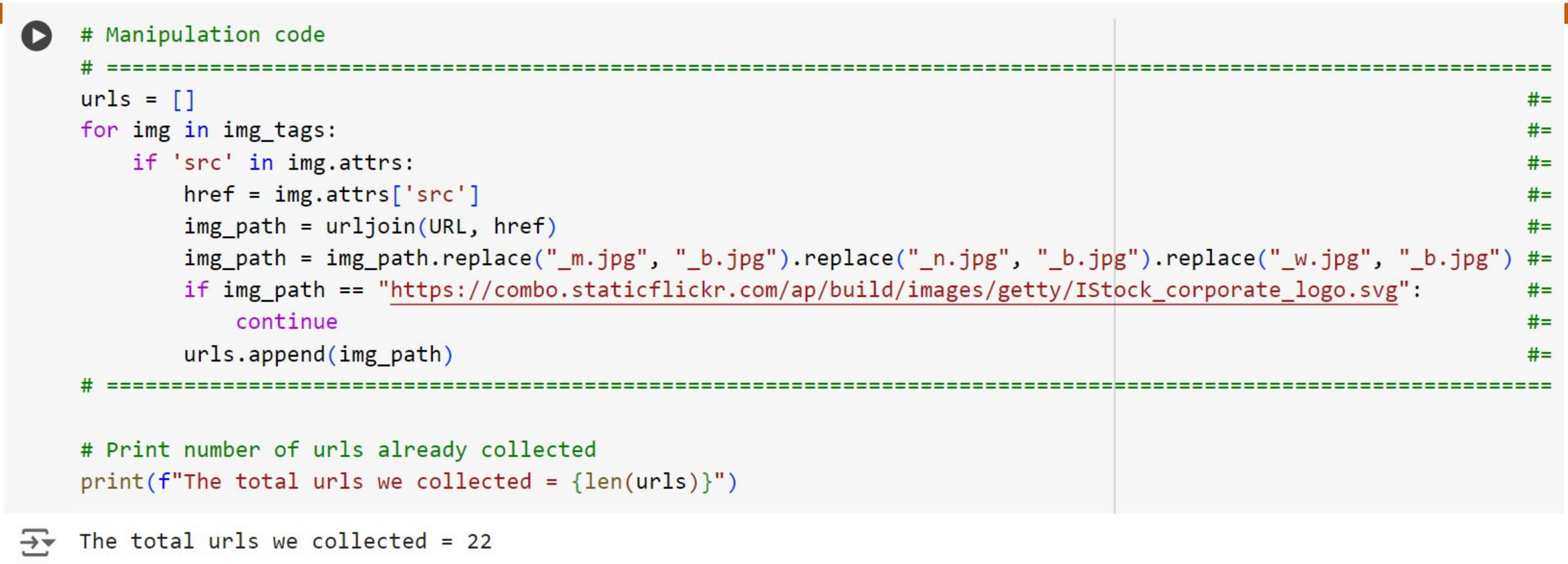
```
# Manipulation code
# ==================================================================================================================
urls = []                                                                                                          #=
for img in img_tags:                                                                                               #=
    if 'src' in img.attrs:                                                                                         #=
        href = img.attrs['src']                                                                                    #=
        img_path = urljoin(URL, href)                                                                              #=
        img_path = img_path.replace("_m.jpg", "_b.jpg").replace("_n.jpg", "_b.jpg").replace("_w.jpg", "_b.jpg") #=
        if img_path == "https://combo.staticflickr.com/ap/build/images/getty/IStock_corporate_logo.svg":           #=
            continue                                                                                               #=
        urls.append(img_path)                                                                                      #=
# ==================================================================================================================

# Print number of urls already collected
print(f"The total urls we collected = {len(urls)}")
```

```
The total urls we collected = 22
```

- ❖ Why we only collected 22 urls?
  - Answer: …
- ❖ How can we collect more urls as we want?
  - Answer: …

Quiz time



```
# Manipulation code
# ==========================================================================================
urls = []                                                                                 #=
for img in img_tags:                                                                      #=
    if 'src' in img.attrs:                                                                #=
        href = img.attrs['src']                                                           #=
        img_path = urljoin(URL, href)                                                     #=
        img_path = img_path.replace("_m.jpg", "_b.jpg").replace("_n.jpg", "_b.jpg").replace("_w.jpg", "_b.jpg") #=
        if img_path == "https://combo.staticflickr.com/ap/build/images/getty/IStock_corporate_logo.svg":       #=
            continue                                                                      #=
        urls.append(img_path)                                                             #=
# ==========================================================================================

# Print number of urls already collected
print(f"The total urls we collected = {len(urls)}")
```

```
The total urls we collected = 22
```

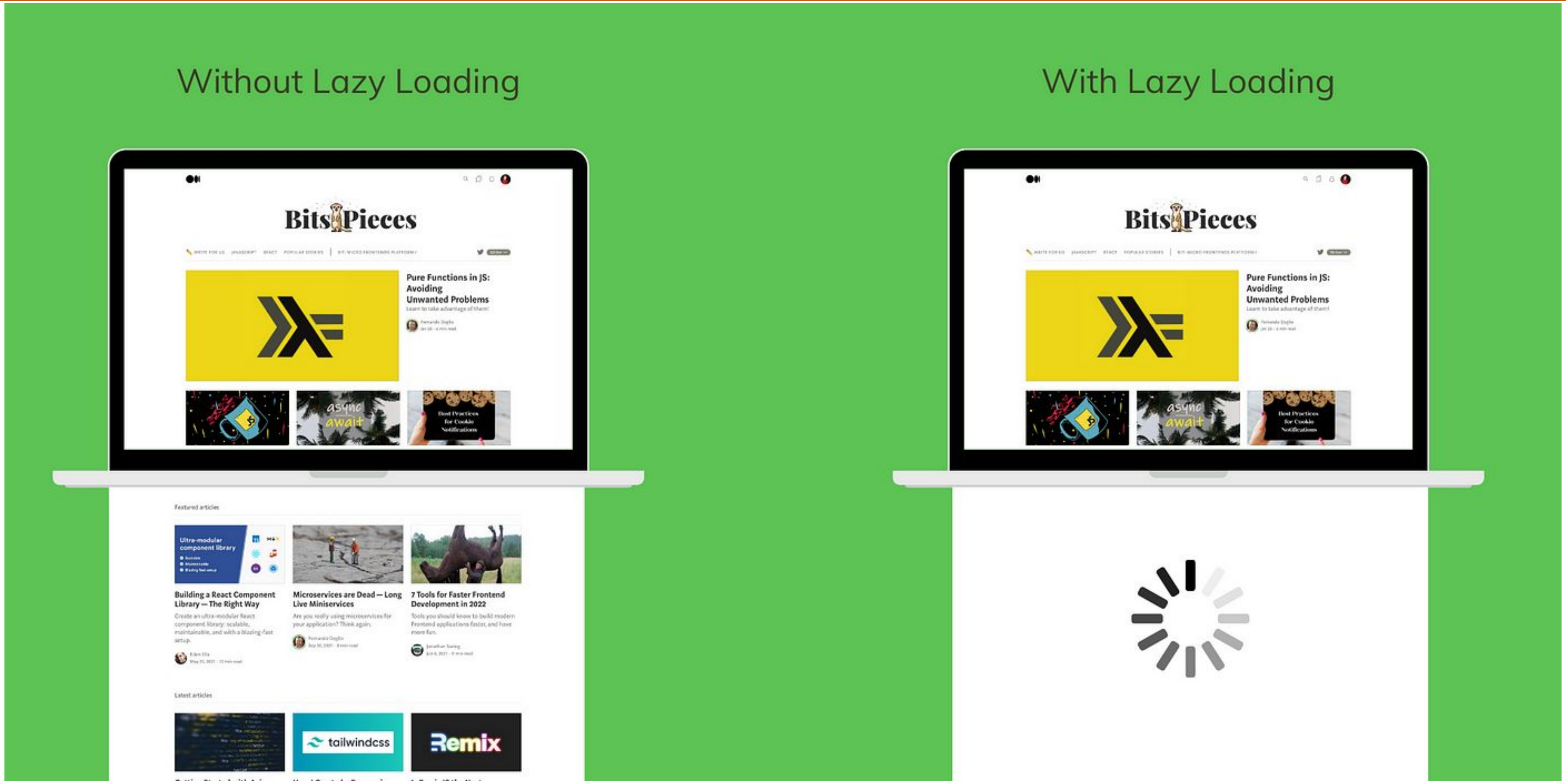
❖ Why we only collected 22 urls?

- Answer: Lazy loading and restrict loading more image.

❖ How can we collect more urls as we want?

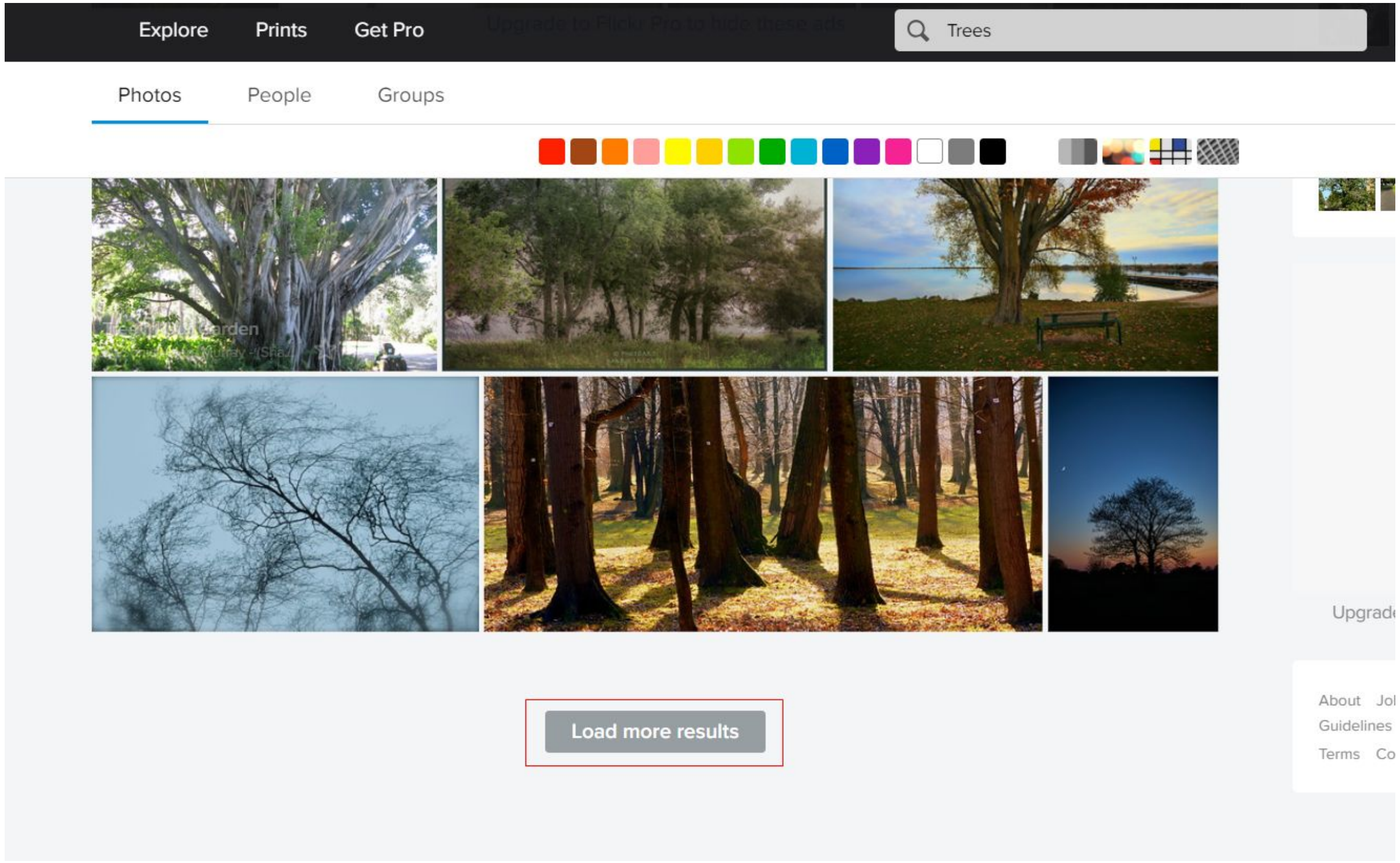
- Answer: Use Selenium loading more content in searching page.

Lazy loading



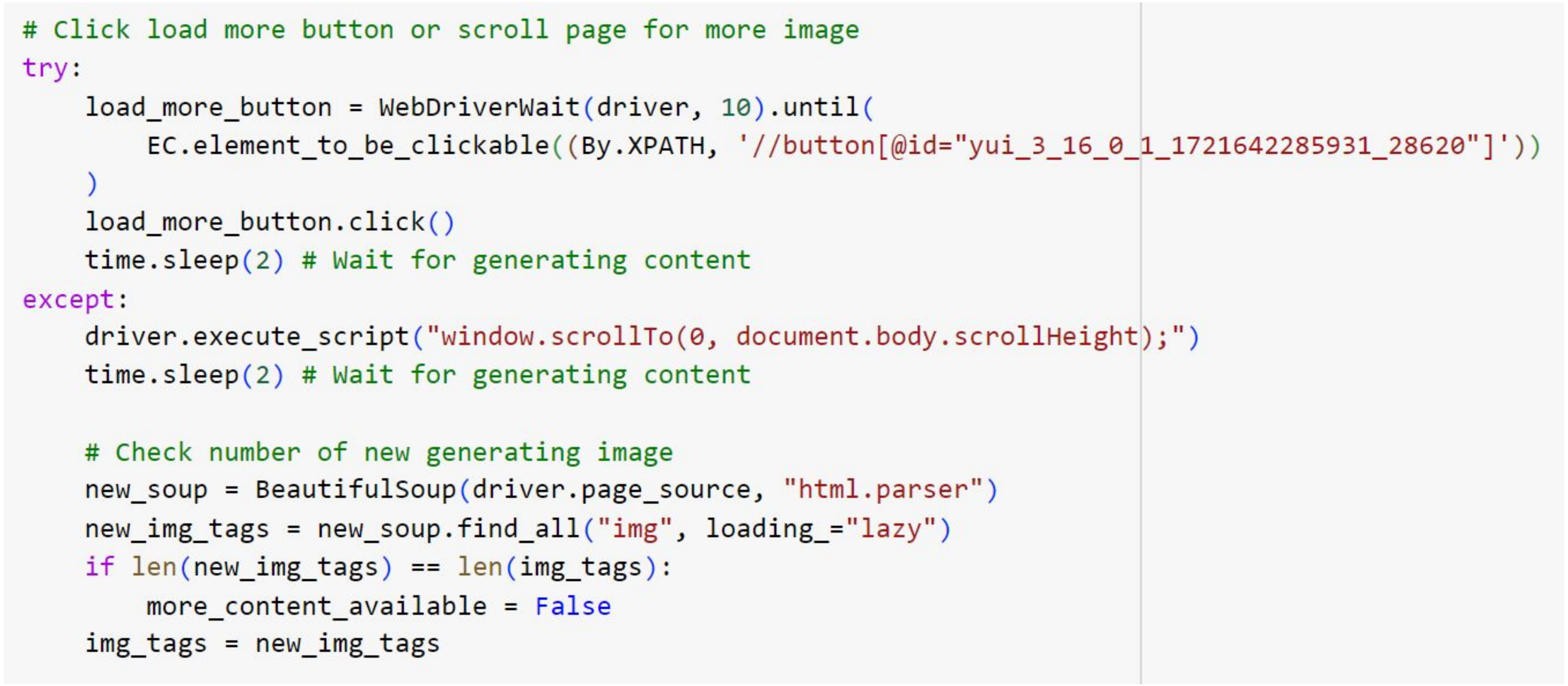
Lazy loading is a design pattern commonly used in programming, especially in web development, to delay the loading of resources until they are actually needed.

Load more result



We need to click this button in code view to load more content until we collect enough urls

Handle with Selenium

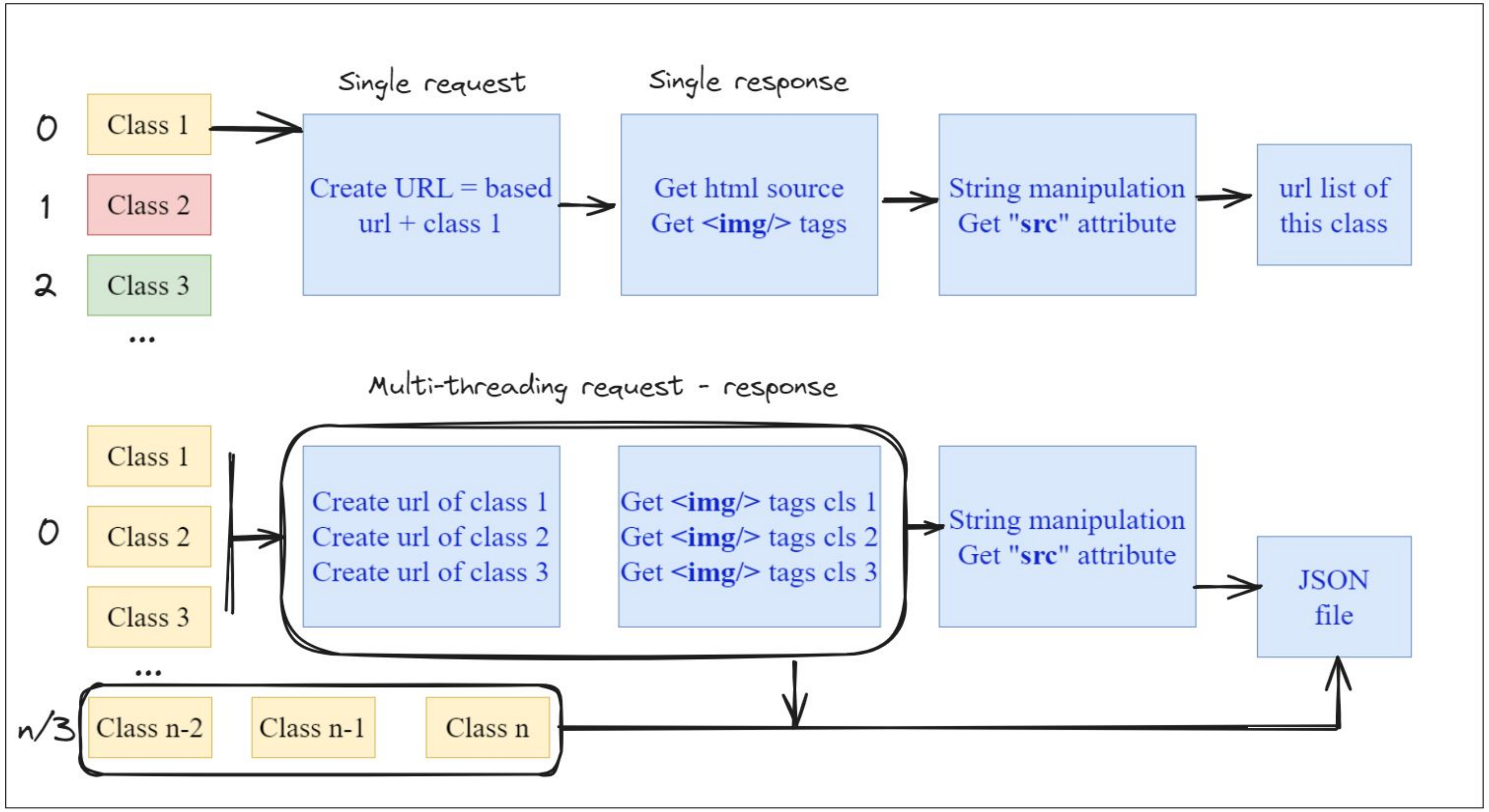


```python
# Click load more button or scroll page for more image
try:
    load_more_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.XPATH, '//button[@id="yui_3_16_0_1_1721642285931_28620"]'))
    )
    load_more_button.click()
    time.sleep(2) # Wait for generating content
except:
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(2) # Wait for generating content

    # Check number of new generating image
    new_soup = BeautifulSoup(driver.page_source, "html.parser")
    new_img_tags = new_soup.find_all("img", loading_="lazy")
    if len(new_img_tags) == len(img_tags):
        more_content_available = False
    img_tags = new_img_tags
```

How can we scrape image urls for multiple classes?

Collect urls of all class



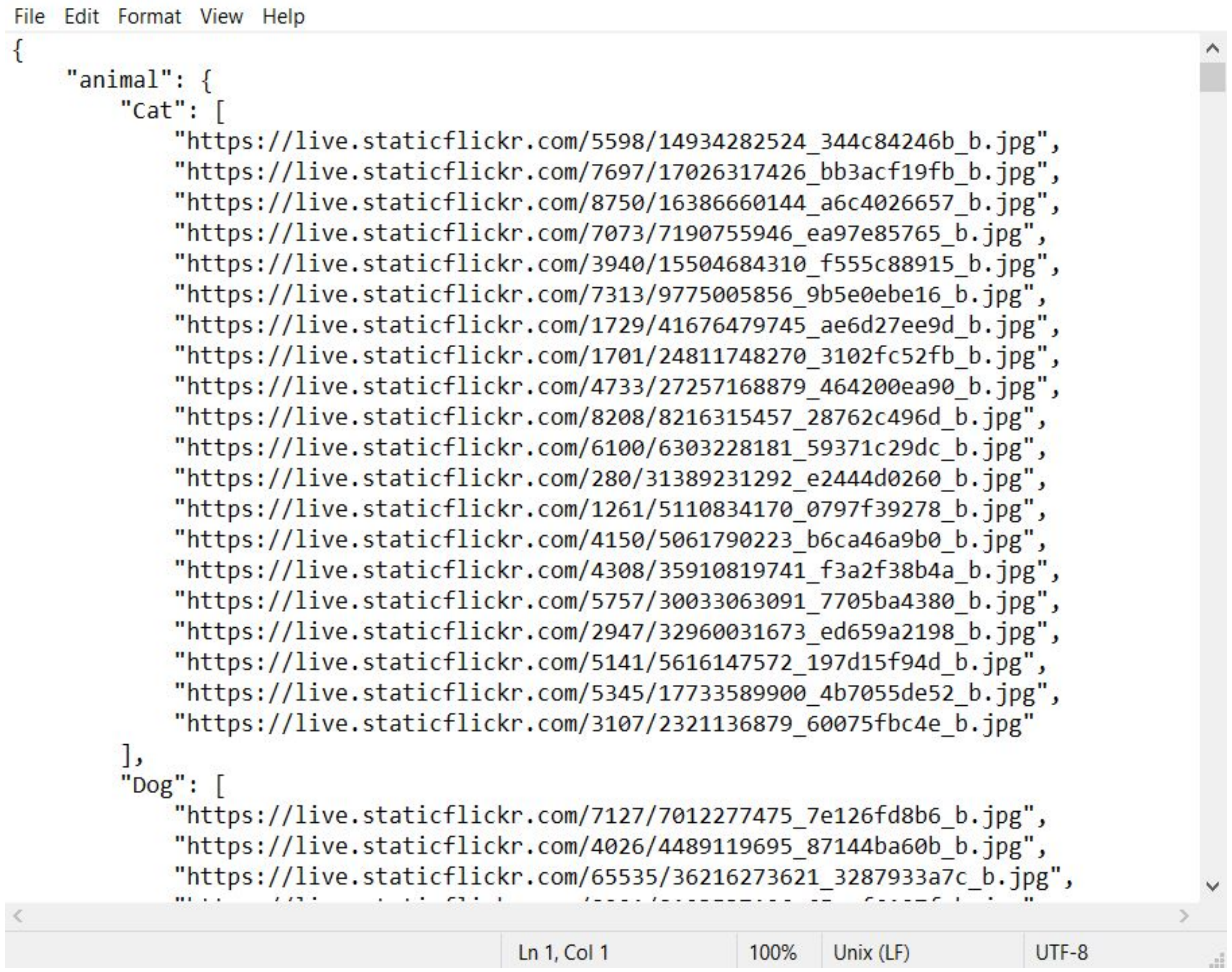
Combine process
in Class
UrlScraper

UrlScraper

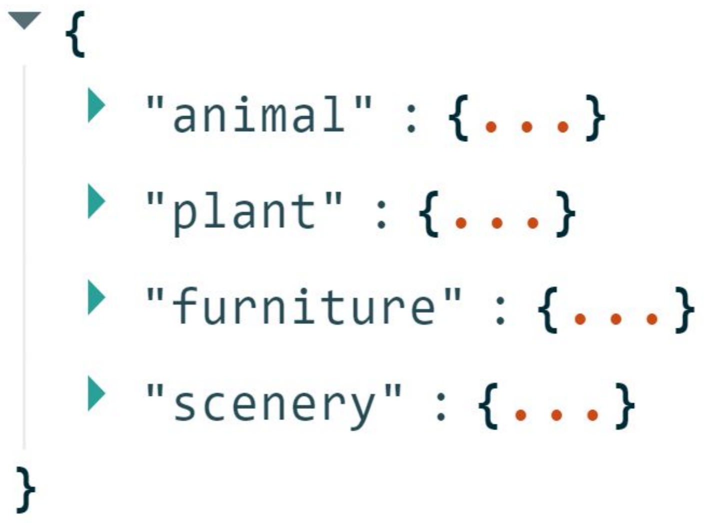
- url_template: str
- max_images: int
- max_workers: int

+ __init__(url_template, max_images,
+ setup_environment()
+ get_url_images(term: str) -> list
+ scrape_images(categories: dict) -> dict
+ save_to_file(data: dict, filename: str) -> None

Final JSON file



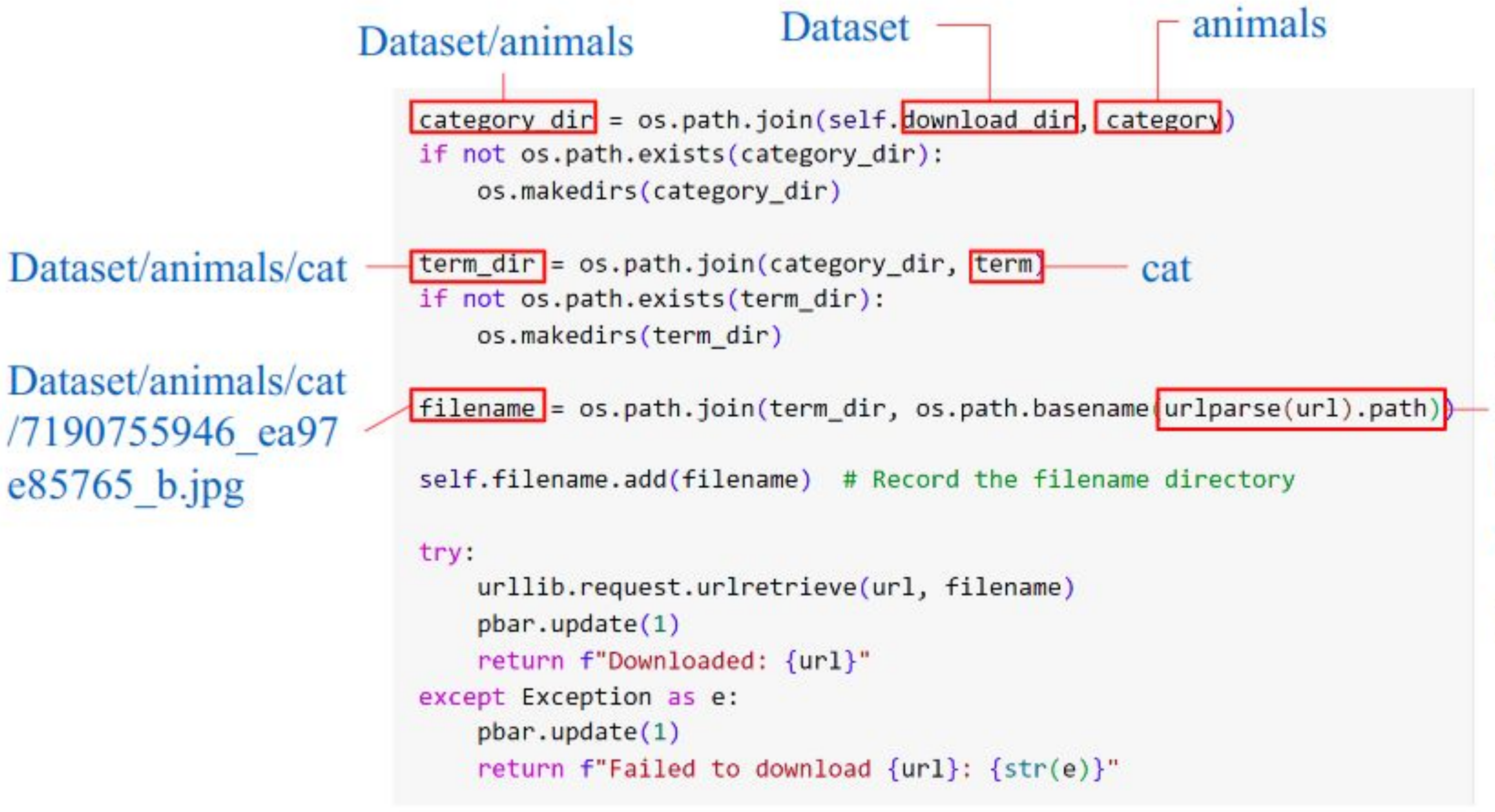
```
{
    "animal": {
        "Cat": [
            "https://live.staticflickr.com/5598/14934282524_344c84246b_b.jpg",
            "https://live.staticflickr.com/7697/17026317426_bb3acf19fb_b.jpg",
            "https://live.staticflickr.com/8750/16386660144_a6c4026657_b.jpg",
            "https://live.staticflickr.com/7073/7190755946_ea97e85765_b.jpg",
            "https://live.staticflickr.com/3940/15504684310_f555c88915_b.jpg",
            "https://live.staticflickr.com/7313/9775005856_9b5e0ebe16_b.jpg",
            "https://live.staticflickr.com/1729/41676479745_ae6d27ee9d_b.jpg",
            "https://live.staticflickr.com/1701/24811748270_3102fc52fb_b.jpg",
            "https://live.staticflickr.com/4733/27257168879_464200ea90_b.jpg",
            "https://live.staticflickr.com/8208/8216315457_28762c496d_b.jpg",
            "https://live.staticflickr.com/6100/6303228181_59371c29dc_b.jpg",
            "https://live.staticflickr.com/280/31389231292_e2444d0260_b.jpg",
            "https://live.staticflickr.com/1261/5110834170_0797f39278_b.jpg",
            "https://live.staticflickr.com/4150/5061790223_b6ca46a9b0_b.jpg",
            "https://live.staticflickr.com/4308/35910819741_f3a2f38b4a_b.jpg",
            "https://live.staticflickr.com/5757/30033063091_7705ba4380_b.jpg",
            "https://live.staticflickr.com/2947/32960031673_ed659a2198_b.jpg",
            "https://live.staticflickr.com/5141/5616147572_197d15f94d_b.jpg",
            "https://live.staticflickr.com/5345/17733589900_4b7055de52_b.jpg",
            "https://live.staticflickr.com/3107/2321136879_60075fbc4e_b.jpg"
        ],
        "Dog": [
            "https://live.staticflickr.com/7127/7012277475_7e126fd8b6_b.jpg",
            "https://live.staticflickr.com/4026/4489119695_87144ba60b_b.jpg",
            "https://live.staticflickr.com/65535/36216273621_3287933a7c_b.jpg",
```



```
{
    "animal" : {...}
    "plant" : {...}
    "furniture" : {...}
    "scenery" : {...}
}
```

Downloading images via urls with multi-threading

Download 1 image



```python
category_dir = os.path.join(self.download_dir, category)
if not os.path.exists(category_dir):
    os.makedirs(category_dir)

term_dir = os.path.join(category_dir, term)
if not os.path.exists(term_dir):
    os.makedirs(term_dir)

filename = os.path.join(term_dir, os.path.basename(urlparse(url).path))

self.filename.add(filename)  # Record the filename directory

try:
    urllib.request.urlretrieve(url, filename)
    pbar.update(1)
    return f"Downloaded: {url}"
except Exception as e:
    pbar.update(1)
    return f"Failed to download {url}: {str(e)}"
```

scheme='https',

netloc='live.staticflick
r.com'
path='/7073/71907559
46_ea97e85765_b.jpg'
params=''

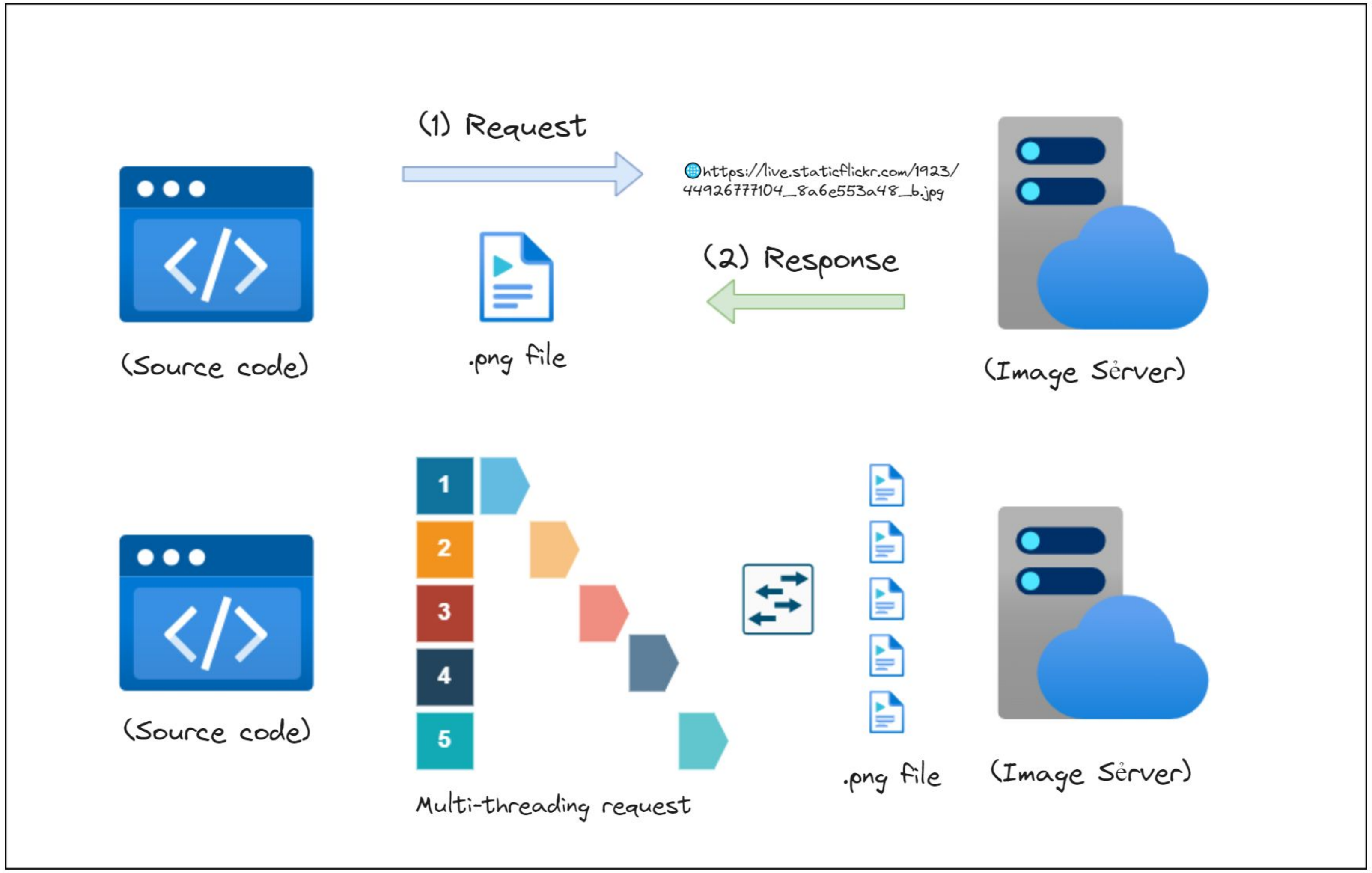
query=''

fragment=''

url: "https://live.staticflickr.com/7313/9775005856_9b5e0ebe16_b.jpg",

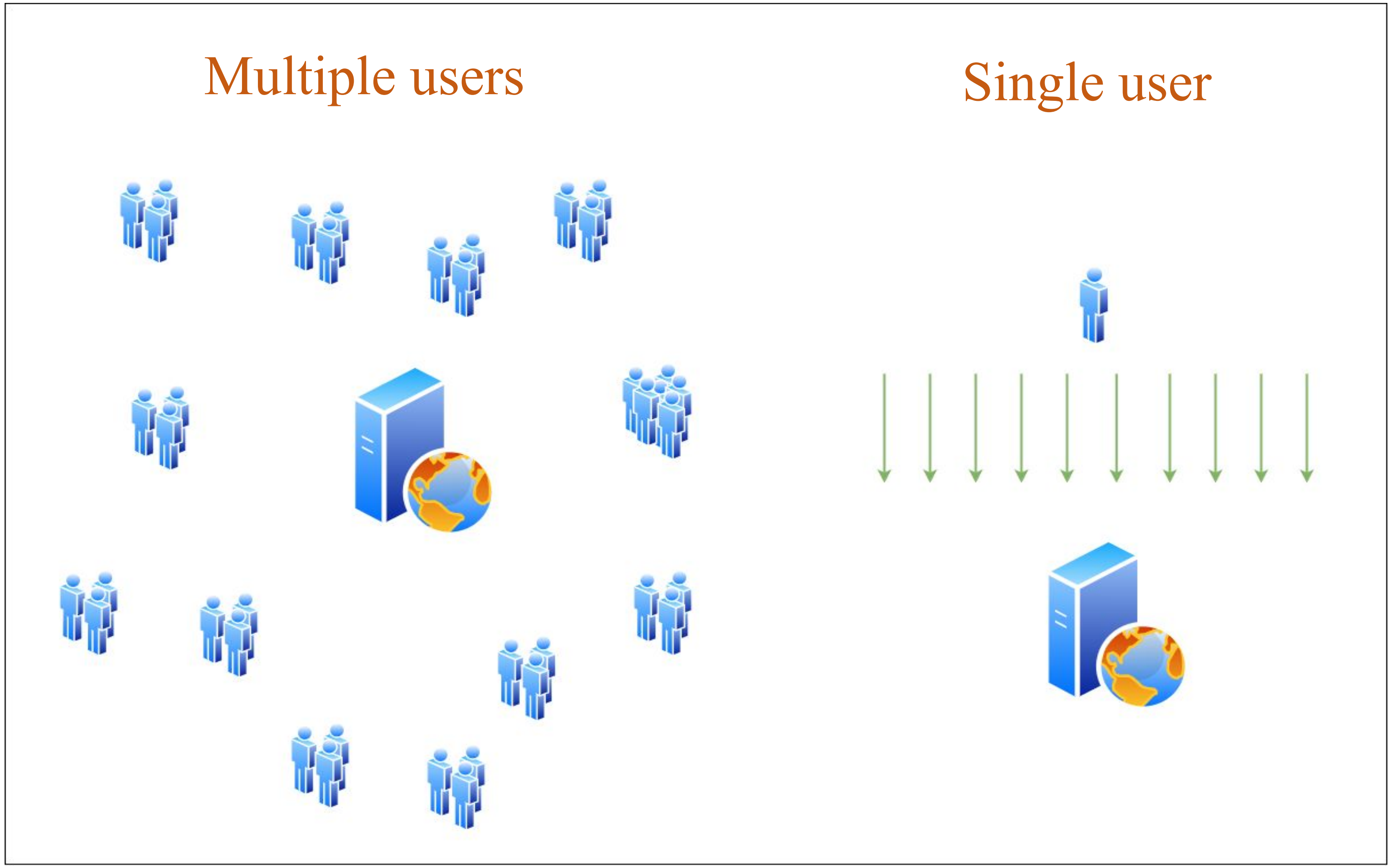
AI VIETNAM
All-in-One Course

Download images with multi-threading



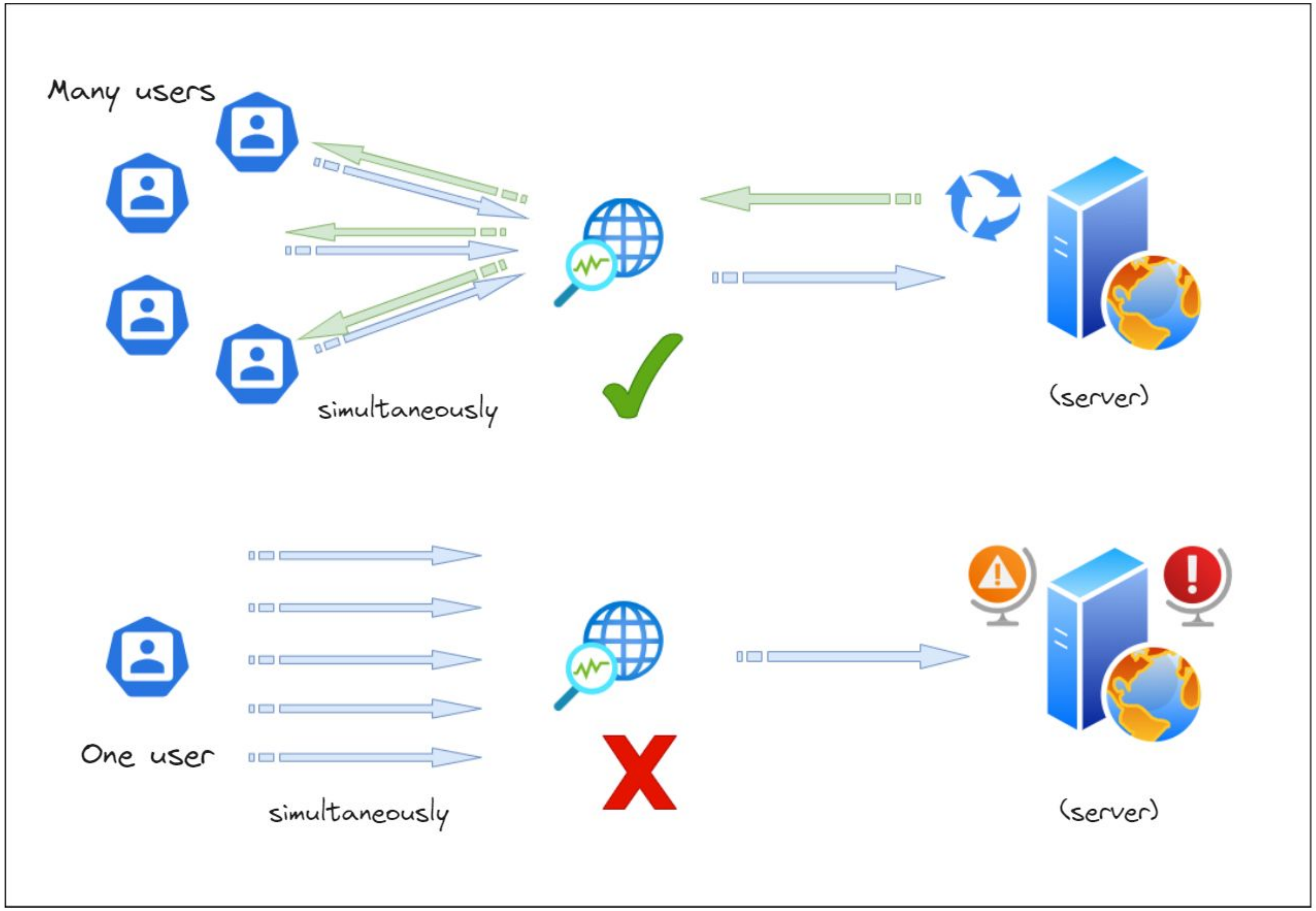
Quiz time

- ❖ What is difference between 100.000 request from many users and 1000 requests from one user? (in 5 seconds)

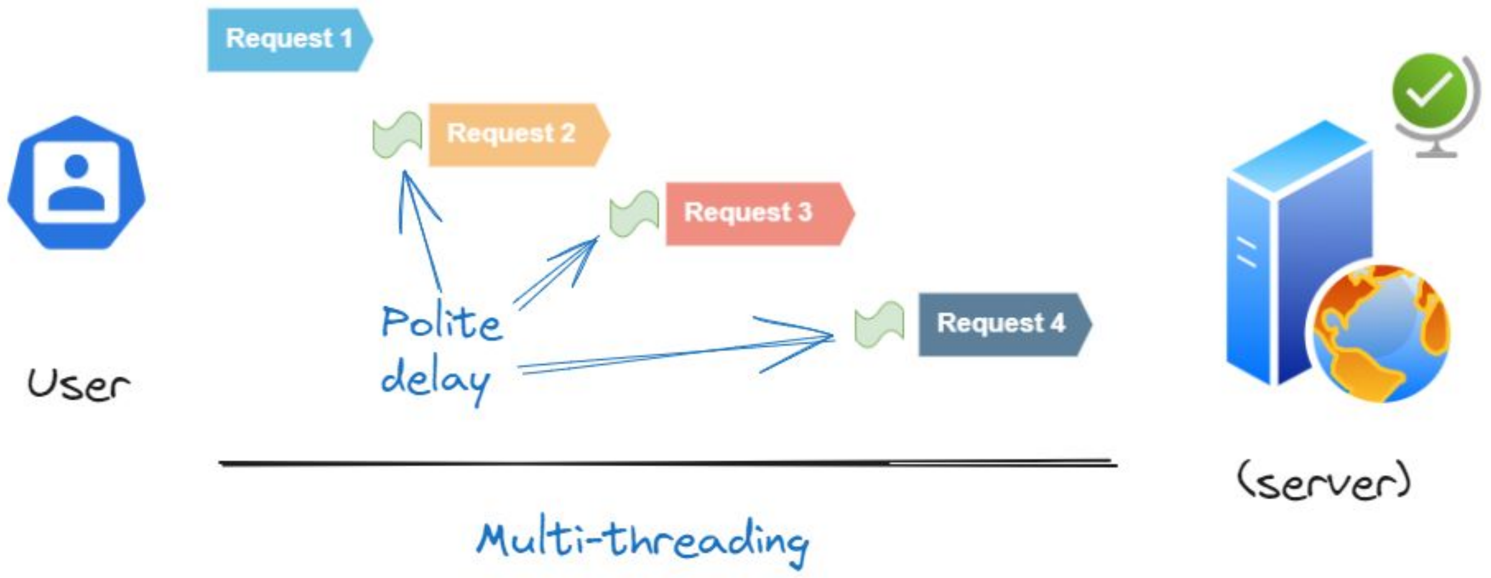


Why we need polite delay?

Server Overloading



Polite delay

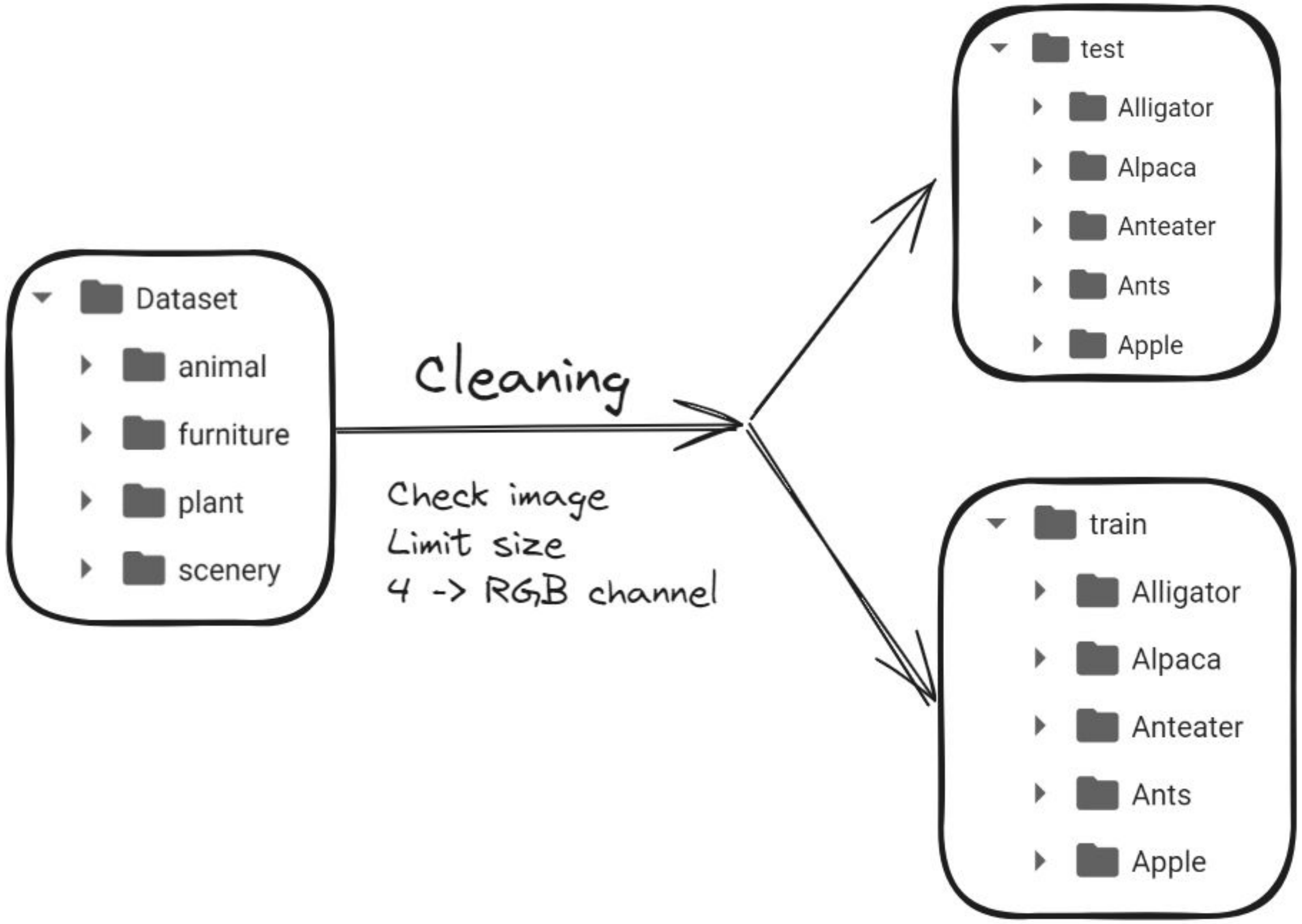


❖ Polite delay in web scraping is the practice of adding a pause between consecutive requests to a website to avoid overloading the server and to respect the website's resources.

Clean and Organize our final dataset

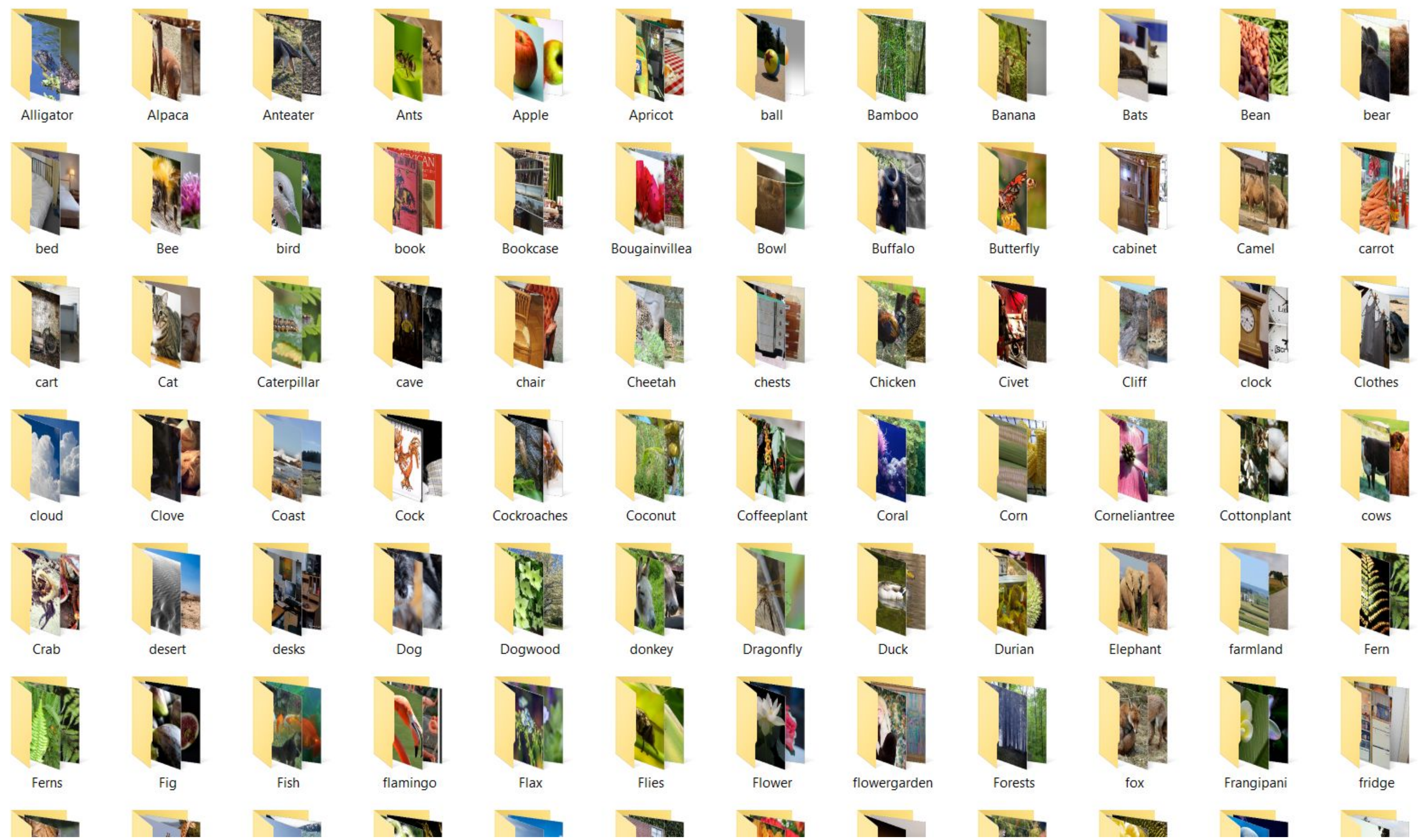
AI VIETNAM
All-in-One Course

Build final data folder



AI VIETNAM
All-in-One Course

Final dataset


Alligator
Alpaca
Anteater
Ants
Apple
Apricot
ball
Bamboo
Banana
Bats
Bean
bear
bed
Bee
bird
book
Bookcase
Bougainvillea
Bowl
Buffalo
Butterfly
cabinet
Camel
carrot
cart
Cat
Caterpillar
cave
chair
Cheetah
chests
Chicken
Civet
Cliff
clock
Clothes
cloud
Clove
Coast
Cock
Cockroaches
Coconut
Coffeeplant
Coral
Corn
Corneliantree
Cottonplant
cows
Crab
desert
desks
Dog
Dogwood
donkey
Dragonfly
Duck
Durian
Elephant
farmland
Fern
Ferns
Fig
Fish
flamingo
Flax
Flies
Flower
flowergarden
Forests
fox
Frangipani
fridge