### BUY RENT KENYA PROJECT (SCRAPING)

**1. Import Libraries**

import requests

from bs4 import BeautifulSoup

import pandas as pd

* **requests**: Used to send HTTP requests to the website and retrieve the content of each page.
* **BeautifulSoup**: A powerful library for parsing HTML content and extracting specific data (in this case, property details).
* **pandas**: Used to store the scraped data in a structured way (a DataFrame) and later save it to a CSV file.

**2. Function Definition - scrape\_pages(start\_page, end\_page)**

This function is responsible for scraping properties between a specified range of pages (from start\_page to end\_page).

**Input Parameters:**

* **start\_page**: The first page number to scrape (e.g., 1).
* **end\_page**: The last page number to scrape (e.g., 4).

**3. Base URL and Headers**

base\_url = 'https://www.buyrentkenya.com/houses-for-sale'

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'

}

* **base\_url**: The root URL where the property listings are located.
* **headers**: The headers passed in the request to mimic a real web browser, preventing the server from blocking the request due to it being sent by a bot.

**4. Initialize List to Store Data**

properties = []

* **properties**: A list that will hold dictionaries for each property. Each dictionary will contain details like the property title, price, location, etc.

**5. Loop Through Pages**

for page\_num in range(start\_page, end\_page + 1):

url = f'{base\_url}?page={page\_num}'

print(f"Scraping page {page\_num}: {url}")

response = requests.get(url, headers=headers)

* The loop runs from start\_page to end\_page, constructing the URL for each page by appending ?page={page\_num} to the base URL.
* It then sends an HTTP GET request for each page using the requests library.
* The script prints the current page being scraped for debugging purposes.

**6. Check Response Status**

if response.status\_code != 200:

print(f"Failed to retrieve the page. Status code: {response.status\_code}")

continue

* If the response status code is not 200 (which indicates a successful request), the page is skipped, and an error message is printed.

**7. Parse HTML with BeautifulSoup**

soup = BeautifulSoup(response.content, 'html.parser')

* The HTML content of the page is parsed using BeautifulSoup so that it can be easily searched and manipulated.

**8. Find Property Listings**

listings = soup.find\_all('div', class\_='listing-card')

* find\_all is used to find all the div elements with the class listing-card, which represent individual property listings on the page.

**9. Extract Property Details**

For each property listing, the code extracts key details like title, price, location, and property attributes (bedrooms, bathrooms, and size):

**Title**:

title\_tag = listing.find('h2')

title = title\_tag.get\_text(strip=True) if title\_tag else 'No title'

* + Looks for the <h2> tag containing the property title. If it exists, it extracts the text; otherwise, it defaults to 'No title'.

**Price**:

price\_tag = listing.find('p', class\_='text-xl font-bold leading-7 text-grey-900')

price = price\_tag.get\_text(strip=True) if price\_tag else 'No price'

* + Extracts the price from the <p> tag with the class text-xl font-bold leading-7 text-grey-900. If the price tag is not found, it defaults to 'No price'.

**Location**:

location\_tag = listing.find('p', class\_='ml-1 truncate text-sm font-normal capitalize text-grey-650')

location = location\_tag.get\_text(strip=True) if location\_tag else 'No location'

* + Looks for the location in a <p> tag with a specific class. If not found, it defaults to 'No location'.

**Swiper Slide Data (Bedrooms, Bathrooms, Size)**:

swiper\_div = listing.find('div', class\_='scrollable-list')

bedrooms = bathrooms = size = 'N/A'

if swiper\_div:

slides = swiper\_div.find\_all('div', class\_='swiper-slide')

for slide in slides:

text = slide.get\_text(strip=True)

if 'Bedroom' in text:

bedrooms = text

elif 'Bathroom' in text:

bathrooms = text

elif 'm²' in text:

size = text

* + The script finds a div with the class scrollable-list containing multiple swiper-slide elements that store the number of bedrooms, bathrooms, and size (in square meters). It checks each slide's text to assign the appropriate values to bedrooms, bathrooms, and size.

**10. Append Data to List**

properties.append({

'Title': title,

'Price': price,

'Location': location,

'Bedrooms': bedrooms,

'Bathrooms': bathrooms,

'Size': size

})

* Each property’s details are stored in a dictionary, which is appended to the properties list.

**11. Convert to DataFrame**

df = pd.DataFrame(properties)

return df

* After all pages have been scraped, the properties list is converted to a Pandas DataFrame, which organizes the data into a tabular format.

**Final Output:**

The function will return a DataFrame (df) containing the scraped properties, which can then be saved to a CSV file or further analyzed.

**How to Use:**

You can call the function scrape\_pages(start\_page, end\_page) with the desired page range (e.g., from page 1 to page 4) to scrape properties from multiple pages. For example:

df = scrape\_pages(1, 4) # Scrapes from page 1 to page 4

df.to\_csv('properties.csv', index=False) # Save the scraped data to a CSV file