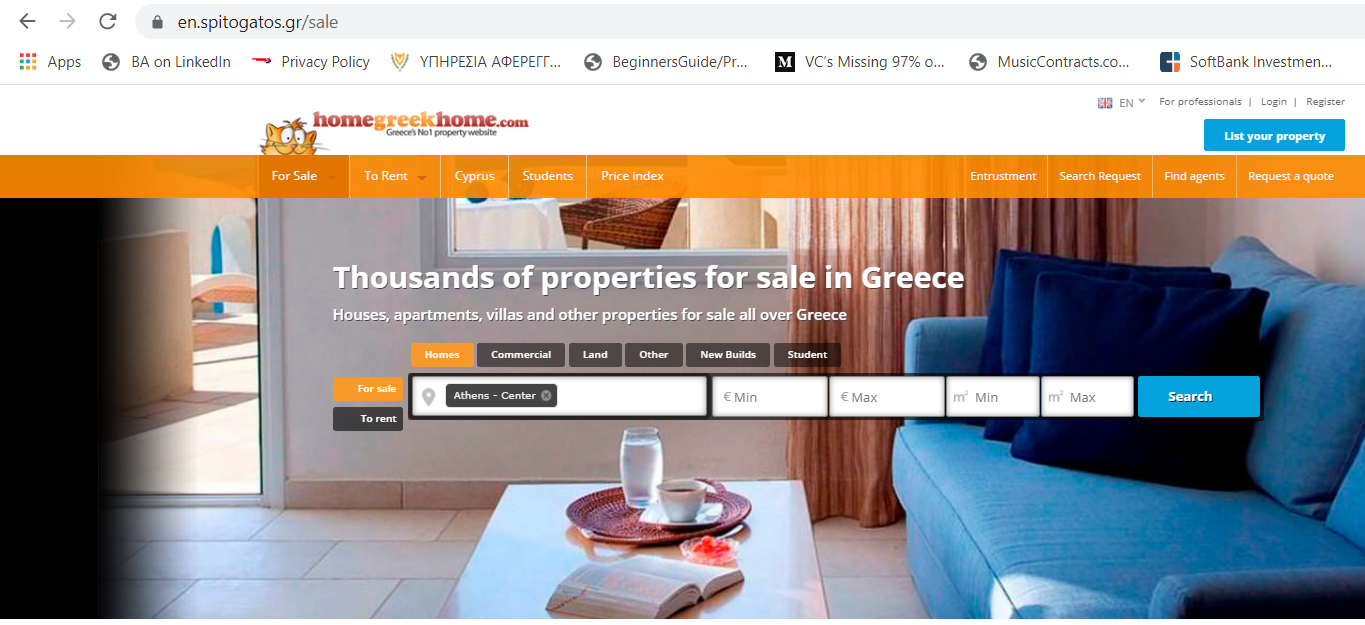
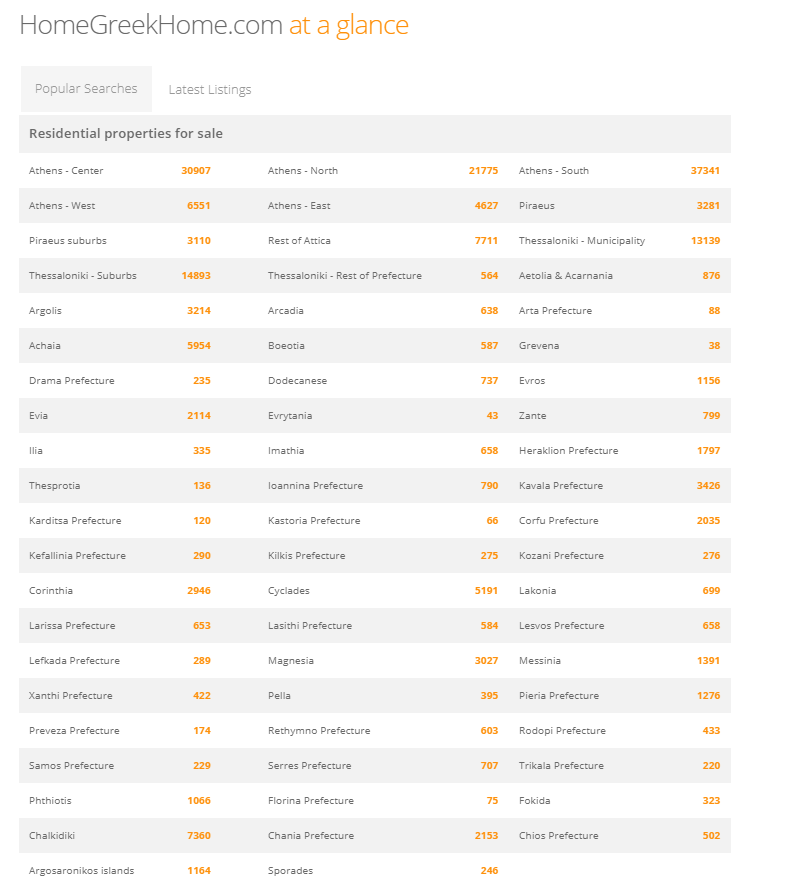
**FINAL PROJECT**

**Step 1:**

Develop code to scrape the website en.spitogatos.gr that has English listings (and Greek) of thousands of properties listed for sale in Greece. For the purpose of this project we are focusing just on residential properties. Project can be adapted for commercial properties as well.

Page: <https://en.spitogatos.gr/sale>





Looking into the json data loaded from the <https://en.spitogatos.gr/sale>, identified the element: div id="latestTabs\_popularSearches" as the pointer to the above container that has the links to real estate offerings for each region of Greece.

If that element exists then need to look for each instance of the div class="css-table-cell padding" and load the json which corresponds to one single region of Greece and has a url link the real estate offering for that region. Loop through this list and run a loop until no more such json’s exist. Need to be careful to exclude from this list the instances where instead of a link to a region we have the number of properties in orange defined by the attibute span class="orange-text”

For each ="css-table-cell padding" json need to read the href url from : a href="https://en.spitogatos.gr/for\_sale-homes/athens-center" and open the url for that region of Greece to further look at each real estate offering.

<div id="latestTabs\_popularSearches" class="margin-top accordion-item ui-tabs-panel ui-widget-content ui-corner-bottom" aria-labelledby="ui-id-1" role="tabpanel" aria-expanded="true" aria-hidden="false">

<h4 class="accordion-title">Popular Searches</h4>

<div class="accordion-body">

<div class="bck grey padding">

<h5 class="text semibold">Residential properties for sale</h5>

</div>

<div class="css-table onelook-table">

<div class="css-table-row">

<div class="css-table-cell padding">

<a href="https://en.spitogatos.gr/for\_sale-homes/athens-center">

Athens - Center

</a>

</div>

But exclude:

<div class="css-table-cell padding">

<a href="https://en.spitogatos.gr/for\_sale-homes/sporades">

<span class="orange-text">246</span>

</a>

</div>

So to achieve that with Python:

def getAllHomes(url):

soup = BeautifulSoup(requests.get(url).content, features='html.parser')

**regionContainer** = soup.find('div', attrs={'id': 'latestTabs\_popularSearches'})

if regionContainer == None:

return

**regionElements** = **regionContainer**.find\_all('div', attrs={'class': 'css-table-cell padding'})

if regionElements == None:

return

for regionElement in regionElements:

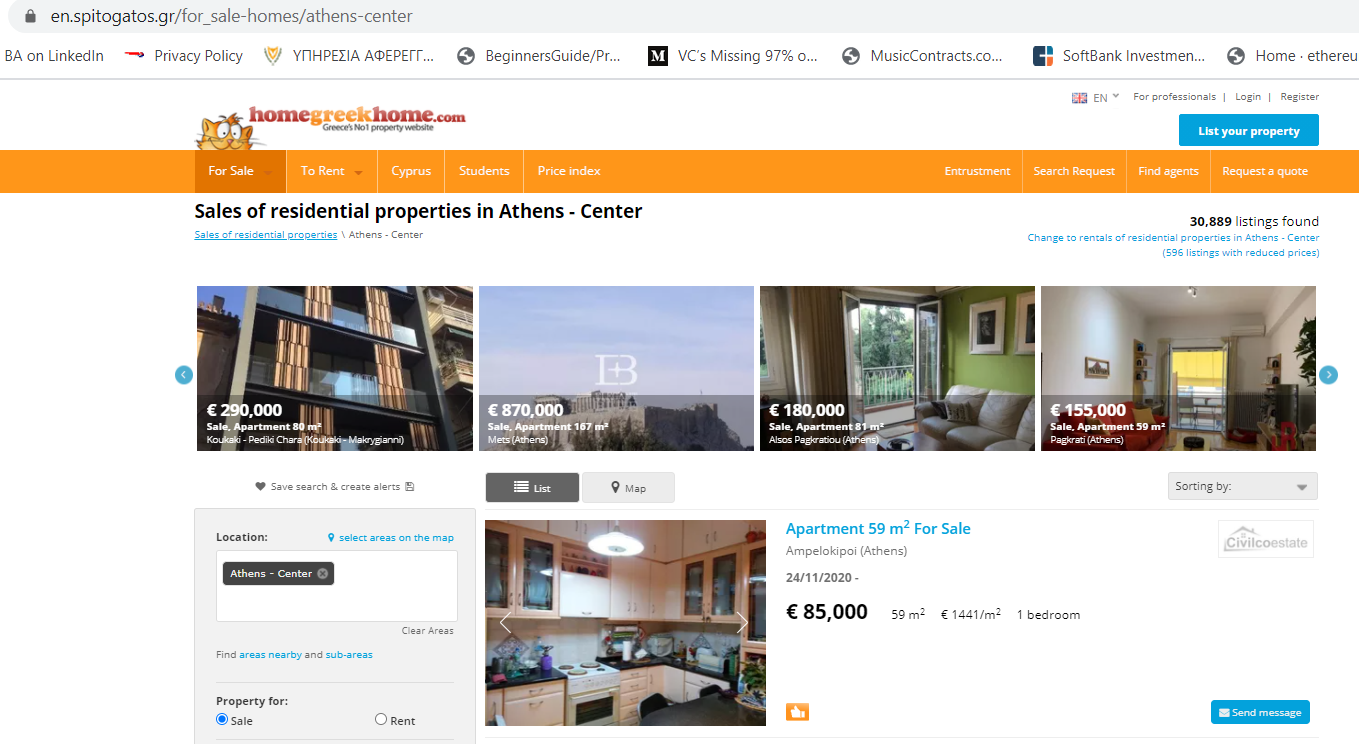
if **regionElement**.find('span', attrs={'class': 'orange-text'}) == None:

# Only pick URLs for region names, not number of homes (orange numbers)

**regionUrl** = **regionElement**.find('a')['href']

getRegionHomes(regionUrl, f, language)

Regional URL pages have only 10 properties listed but at the bottom of the page there is a link to the next 10 properties. So I would need first to extract the info from these properties and then go to the next page.



From this page would need to find the links to the 10 properties that are listed to finally get to the individual property URLs that have the information we require to extract for each property. To find the URL link we search the json by div class="searchListing” and then by div class="bd” and then find all ‘a’ tags. There are multiple occurrences of this combination but we need the second occurrence, so we take the second element of the list url[1] and the ‘href’ column which is a href="https://en.spitogatos.gr/property/1110437170"

<div class="searchListing ….

…

<div class="bd padding….

……..

<a class="img\_right greyscale opacity margin-right-small" href="https://en.spitogatos.gr/find-agents/BROKERHOUSE..

…….

<div class="searchListing\_details\_\_inner clearfix">

<div class="media search\_listing\_title">

<a href=<https://en.spitogatos.gr/find-agents/civilco/10596?profile=branded>….</a><!-- listingAgent -->

<div class="bd padding-right">

<h4 class="text color cyan larger semibold searchListing\_title">

<a href="https://en.spitogatos.gr/property/1110437170">Apartment 59 m<sup>2</sup>

<span class="hide-phone"> For Sale</span></a>

</h4>

def getPageHomes(pageUrl):

soup = BeautifulSoup(requests.get(pageUrl).content, features='html.parser')

**listings** = soup.find\_all('div', attrs={'class': 'searchListing'})

if listings == None:

return

for listing in listings:

urls = listing.find('div', attrs={'class': 'bd'}).find\_all('a')

if urls == None:

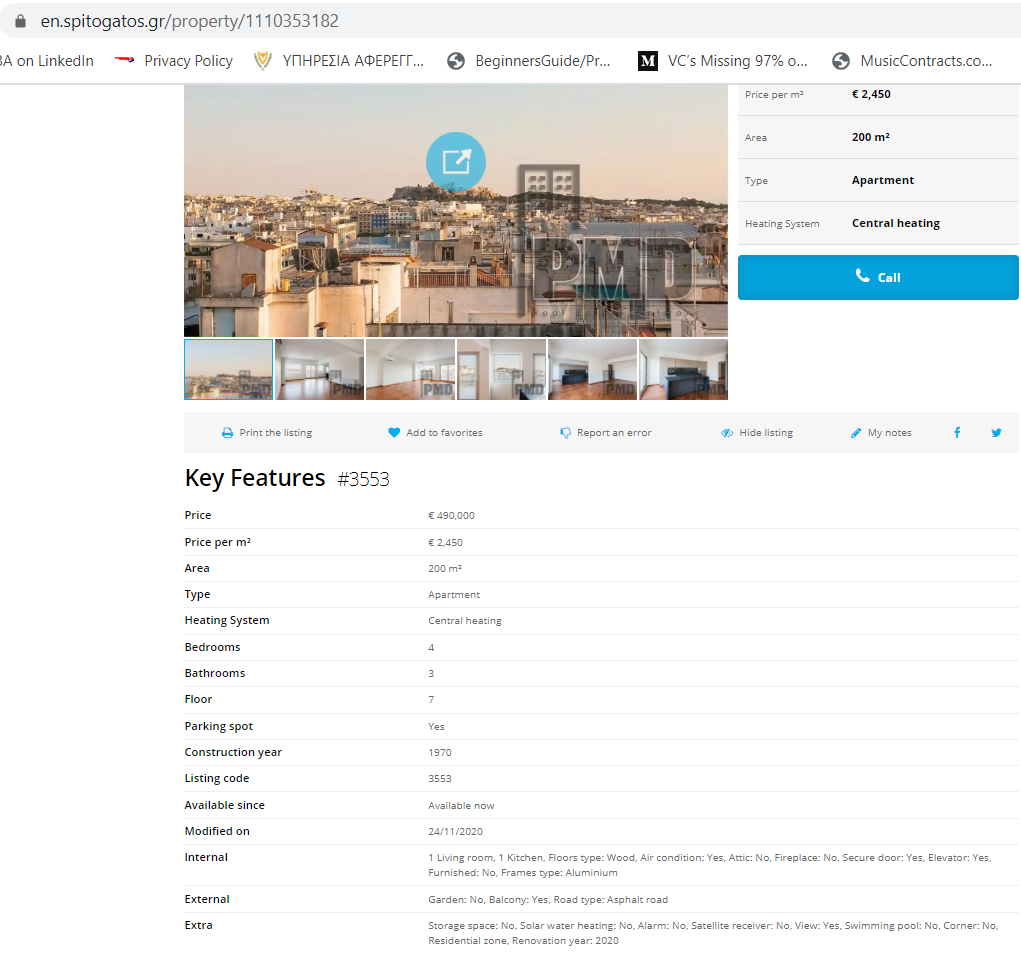
return

homeUrl = urls[1]['href']

getHomeInfo(homeUrl)

return soup // to check if another page exists

Once we are finally on a single real estate offering page, we extract the fields we require and clean up the number data from currency signs, commas, and periods.



Once we are at the single listing url things are a bit simpler. We need to look for all the elements that that fullfill the following criteria div id=“listingDetailsMainContent” and div class=”padding-phone-only” and div class= “padding-small-top”.

That produces a one dimensional list of key:value’s :

homeElements = [

<div class="padding-small-top padding-small-bottom bck highlight links cyan">

<div class="desktop-3-details columns nested text color black semibold">

<h6>Price</h6>

</div>

<div class="desktop-5-details columns nested">

<span class="padding-right">

€ 261,000

</span>

</div>

<div class="clear"></div>

</div>,

<div class="border-top padding-small-top padding-small-bottom bck highlight links cyan">

<div class="desktop-3-details columns nested text color black semibold">

<h6>Price per m²</h6>

</div>

<div class="desktop-5-details columns nested">€ 2,373</div>

<div class="clear"></div>

</div>,

<div class="border-top padding-small-top padding-small-bottom bck highlight links cyan">

<div>

…..

</div>,

,

,

,

</div>, <div class="border-top padding-small-top padding-small-bottom bck highlight links cyan">

<div class="border-top padding-small-top padding-small-bottom bck highlight links cyan">

<div class="desktop-3-details columns nested text color black semibold">

<h6>Internal</h6>

</div>

<div class="desktop-5-details columns nested">1 Living room, 1 Kitchen, 2 WC, Floors type: Wood, Air condition: Yes, Attic: No, Fireplace: No, Playroom: No, Secure door: Yes, Elevator: Yes, Furnished: Yes, Internal staircase: No, Frames type: Aluminium, Double glass: Yes, Window screens: No, Painted: Yes, UnderFloor Heating: No

</div>

<div class="clear"></div>

</div>,

<div class="border-top padding-small-top padding-small-bottom bck highlight links cyan">

<div class="desktop-3-details columns nested text color black semibold">

<h6>External</h6>

</div>

<div class="desktop-5-details columns nested">Garden: No, Lot size: 263 m², Balcony: Yes, Size of balconies: 13 m², Awning: No, Road type: Asphalt road, Orientation: East meridian

</div>

<div class="clear"></div>

</div>,

<div class="border-top padding-small-top padding-small-bottom bck highlight links cyan">

<div class="desktop-3-details columns nested text color black semibold">

<h6>Extra</h6>

</div>

<div class="desktop-5-details columns nested">New development: Yes, Luxurious, Storage space: Yes, Solar water heating: Yes, Alarm: No, Satellite receiver: No, Airy: Yes, Bright: Yes, Disabled access: No, View: Yes, Penthouse: No, Swimming pool: No, Average monthly shared expenses: 50 €, Pets welcome: Yes, Facade: Yes, Corner: No, Residential zone, Renovated: Yes, Renovation year: 2020, Requires renovation: No, Preserved: No, Investment: Yes, Neoclassical: No, Unfinished: No, Night power: Yes, Suitable for commercial use: Yes, Energy class: A, No agent fee for the buyer</div>

<div class="clear"></div>

</div>

]

Fir the key we search for div class="desktop-3-details and for the values we div class="desktop-5-details . In the case of key=’Price’, we need to be careful because sometimes price is in a further container span class="padding-right but sometimes it isn’t.

Finally, we also store the data from the side table next to the picture of the property by search for div id="carouselSideDetails and then by div class="tableRow" we create a list of key:value pairs of all the side elements which we access by finding all div class=”tableCell”. We only use the key = ‘Neighborhood” value from this search because we already have the other information.

<div class="desktop-3 tablet-3 columns nested">

<div id="carouselSideDetails" class="bck light\_grey hide-print">

<div class="table">

<div class="tableRow">

<div class="tableCell">

<h6 class="text small">**Neighborhood**</h6>

</div>

<div class="tableCell">

<h6 class="text black color bold">**Athens - Center Kipseli** </h6>

</div>

</div>

…

<div class="tableRow bck highlight">

<div class="tableCell">

<h6 class="text small">Heating System</h6>

</div>

<div class="tableCell">

<h6 class="text black color bold">Central heating (Petrol)</h6>

</div>

</div>

</div>

</div>

def getHomeInfo(homeUrl):

try:

html =requests.get(homeUrl).content

except:

return

if html == None:

return

soup = BeautifulSoup(html, features='html.parser')

**mainContent** = **soup**.find('div', attrs={'id': 'listingDetailsMainContent'})

if mainContent == None:

return

**info** = **mainContent**.find('div', attrs={'class': 'padding-phone-only'})

if info == None:

return

**homeInfo** = **HomeInfo**() // our data structure object

homeInfo.url = homeUrl // store individual listing url

**homeElements** = **info.**find\_all('div', attrs={'class': 'padding-small-top'})

if homeElements == None:

return

for homeElement in homeElements:

**key** = **homeElement**.find('div', attrs={'class': 'desktop-3-details'}).text.strip()

if (**key** == '**Price**'):

**priceContainer** = **homeElement**.find('div', attrs={'class': 'desktop-5-details'})

if priceContainer == None:

return

**value** = **priceContainer**.find('span', attrs={'class': 'padding-right'})

else:

**value** = **homeElement**.find('div', attrs={'class': 'desktop-5-details'})

value = value.text.strip().replace('\n', ' ').replace('"', '')

**fieldLabel** = **homeInfo**.fieldLabelsEn.get(key)

if **fieldLabel** != None:

setattr(homeInfo, fieldLabel, value) // store values in attributes of homeInfo object

**sideElements** = **soup**.find('div', attrs={'id': 'carouselSideDetails'}).find\_all('div', attrs={'class': 'tableRow'}

)

if sideElements != None:

for sideElement in sideElements:

**cells** = **sideElement**.find\_all('div', attrs={'class': 'tableCell'})

key = cells[0].text.strip()

value = cells[1].text.strip()

if (**key** == '**Neighborhood**'):

homeInfo.generalRegion = value

break

# Clean prices

homeInfo.price = cleanPrice(homeInfo.price)

homeInfo.pricePerSqm = cleanPrice(homeInfo.pricePerSqm)

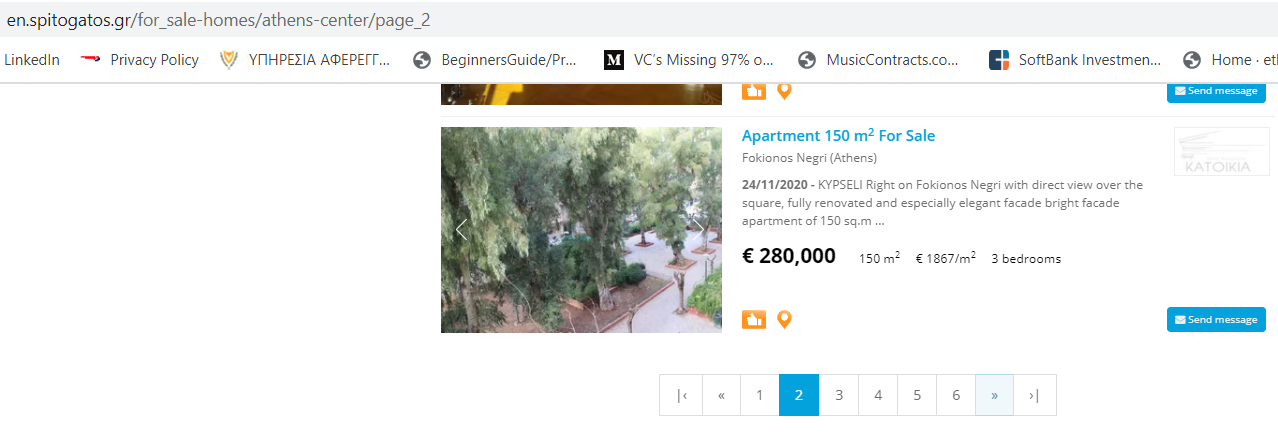
# Clean dates

homeInfo.availableDate = cleanDate(homeInfo.availableDate)

homeInfo.lastUpdate = cleanDate(homeInfo.lastUpdate)

f.write(str(homeInfo) + '\n')

To go to the next page for the next 10 listings, we look at the little next button at the bottom of this page



If we look at the html code, to find the next page we need to look for li class="next" href=”URL” element or if no more pages left li class="disable”:

<li class="next">

<a href="https://en.spitogatos.gr/for\_sale-homes/athens-center/page\_2">»</a>

</li>

<li class="last disabled">

<a class="disable">›|</a></li>

So we implement in Python, the loop that takes us through each page until no more pages exist, after downloading the data on the 10 homes from each page:

def getRegionHomes(regionUrl):

soup = BeautifulSoup(requests.get(regionUrl).content, features='html.parser')

getPageHomes(regionUrl)

**nextPageElement** = soup.find('li', attrs={'class': 'next'})

if nextPageElement == None:

return

while nextPageElement.find('a', attrs={'class': 'disable'}) == None:

# There is a next page; find its URL

**pageUrl** = **nextPageElement**.find('a')['href']

soup = getPageHomes(**pageUrl**)

**nextPageElement** = soup.find('li', attrs={'class': 'next'})

if nextPageElement == None:

return

**RESULTS:**

When I first ran the home\_scrape.py program my IP address was blocked from downloading the residential listings. To debug the code and test if it worked. I subscribed to proxycrawl.com for a key that allows 1000 successful requests so that I can test the python code on a real website with relevant data. I have included a file with 100 successful requests in file properties-residential-20201125.txt and there is a bit more capacity to check the code. I have included a new key in the python code that would allow you to run up to 700 more requests.

**At the end, I decided not to built the database for the project by downloading data from this site. For at least two reasons: 1) while all the data is already publicly available and it seems that scrapping websites may be legal in US, I was not sure if it was legal in Greece and 2) if the website has put in place an authentication barrier, then one should get the company’s permission to copy and use the data. So, I had to look elsewhere to get the data I needed for the project.**

I reached out to Arbitrage Real Estate IKE which publishes research on Greek Real Estate market ([www.arbitrage-re.com](http://www.arbitrage-re.com)) and asked for 60,000 listings to test my machine learning algorithm. They were more than happy to provide me with a database run of 100,000+ residential housing price offerings and they were very interested to see the results of my prediction model analysis.

But the code did work!

Two first lines from properties-residential-20201125.txt (delimited by ‘~’ because data included commas)

*additionalFeatures~address~availableDate~buildingCoefficient~code~externalFeatures~floor~generalRegion~heating~homeType~internalFeatures~lastUpdate~numBathrooms~numBedrooms~parking~price~pricePerSqm~runDate~specificRegion~sqm~url~year*

*New development: Yes, Luxurious, Storage space: Yes, Solar water heating: Yes, Alarm: No, Satellite receiver: No, Airy: Yes, Bright: Yes, Disabled access: No, View: Yes, Penthouse: No, Swimming pool: No, Average monthly shared expenses: 50 €, Pets welcome: Yes, Facade: Yes, Corner: No, Residential zone, Renovated: Yes, Renovation year: 2020, Requires renovation: No, Preserved: No, Investment: Yes, Neoclassical: No, Unfinished: No, Night power: Yes, Suitable for commercial use: Yes, Energy class: A, No agent fee for the buyer~~Available now~~1011~Garden: No, Lot size: 263 m², Balcony: Yes, Size of balconies: 13 m², Awning: No, Road type: Asphalt road, Orientation: East meridian~3~Athens - Center Patision - Acharnon~Central heating (Fan coil)~Apartment~1 Living room, 1 Kitchen, 2 WC, Floors type: Wood, Air condition: Yes, Attic: No, Fireplace: No, Playroom: No, Secure door: Yes, Elevator: Yes, Furnished: Yes, Internal staircase: No, Frames type: Aluminium, Double glass: Yes, Window screens: No, Painted: Yes, UnderFloor Heating: No~2020-11-25~2~3~No~261000~2373~2020-11-25~Ag. Meletiou - Viktorias Sq. - Marni, Patision - Acharnon~110 m²~https://en.spitogatos.gr/property/1110147481~2020~*

or:

|  |  |
| --- | --- |
| additionalFeatures | New development: No, Storage space: No, Solar water heating: No, Alarm: No, Satellite receiver: No, Airy: No, Bright: Yes, View: Yes, Penthouse: No, Swimming pool: No, Average monthly shared expenses: 15 â‚¬, Holiday rental: No, Pets welcome: Yes, Facade: No, Corner: Yes, Residential zone, Renovated: Yes, Requires renovation: No, Preserved: No, Investment: No, Neoclassical: No, Unfinished: No, Night power: Yes, Suitable for commercial use: No, Energy class: A |
| address | Anatoliou 39, 116 32, Athina |
| availableDate | Available now |
| buildingCoefficient | 0.5 |
| code | 106805 |
| externalFeatures | Garden: No, Balcony: Yes, Size of balconies: 5 mÂ², Awning: Yes, Road type: Asphalt road, Orientation: East meridian |
| floor | Ground floor |
| generalRegion | Athens - Center Pagkrati |
| heating | Central heating (Petrol) |
| homeType | Apartment |
| internalFeatures | 1 Living room, 1 Kitchen, Floors type: Wood and tile, Air condition: No, Attic: No, Fireplace: No, Playroom: No, Secure door: No, Elevator: Yes, Furnished: No, Internal staircase: No, Frames type: Aluminium, Double glass: No, Window screens: No, Painted: Yes, UnderFloor Heating: No |
| lastUpdate | 44159 |
| numBathrooms | 1 |
| numBedrooms | 1 |
| parking | No |
| price | 75000 |
| pricePerSqm | 1500 |
| runDate | 44160 |
| specificRegion | Pagkrati Center, Pagkrati |
| sqm | 50 mÂ² |
| url | https://en.spitogatos.gr/property/1110188139 |
| year | 1978 |