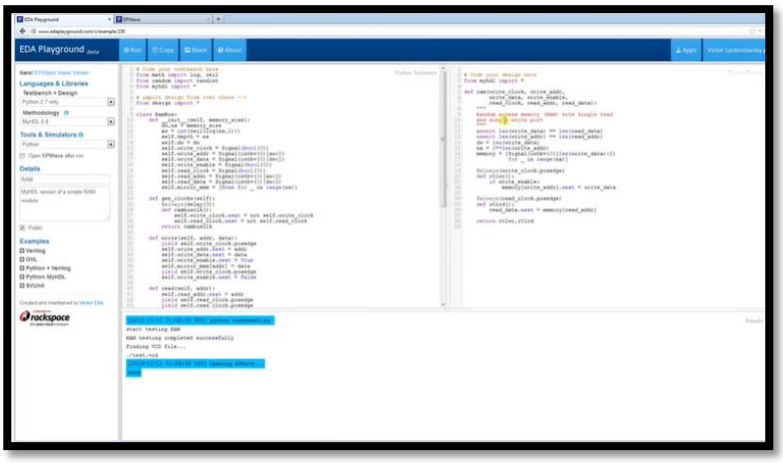
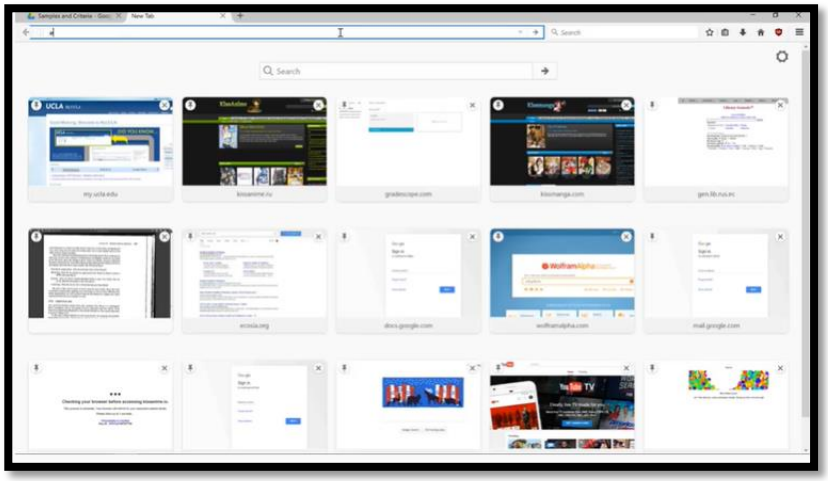
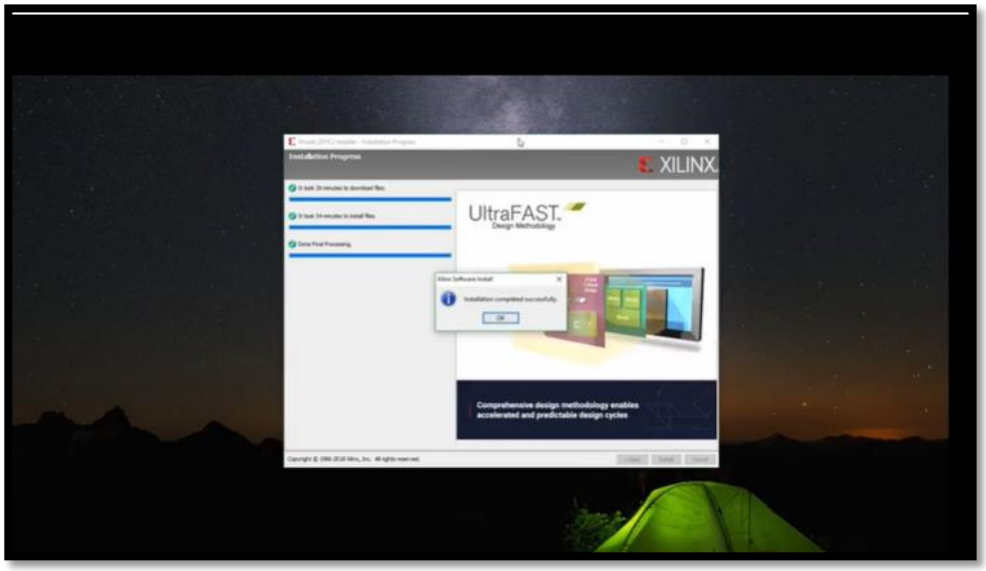
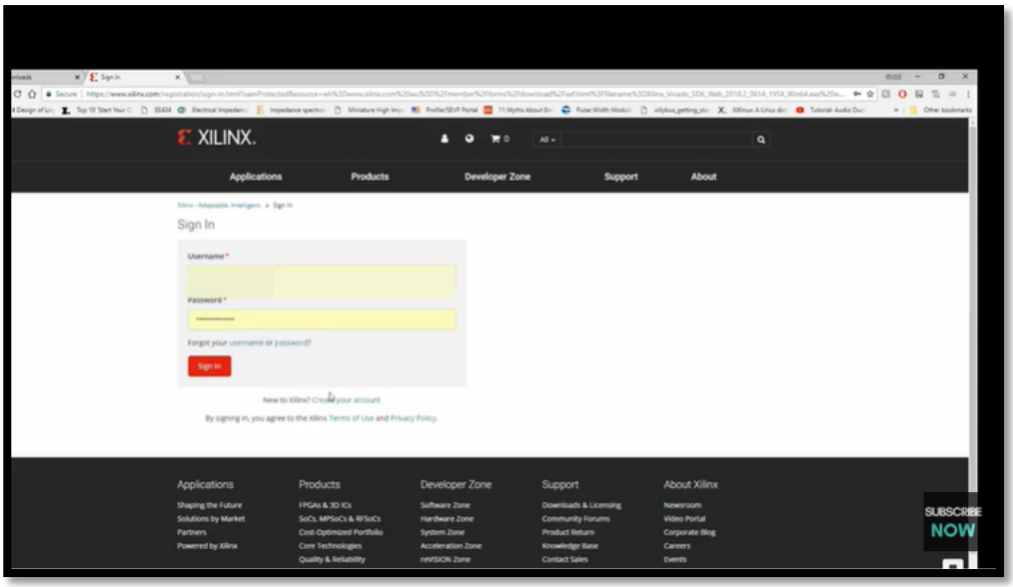
DAILY ASSESSMENT FORMAT

Date:	**03/06/2020**	**Name:**	**Davis S. Patel**
Course:	**DIGITAL DESIGN USING HDL**	**USN:**	**4AL16EC045**
Topic:	**EDA Playground Online complier EDA Playground Tutorial Demo Video How to Download And Install Xilinx Vivado Design Suite Vivado Design Suite for implementation of HDL code**	**Semester & Section:**	**8th - A**
GitHub Repository:	**Davis**		

FORENOON SESSION DETAILS
Image of session    


XILINX.
UltraFAST.
Design Methodology
Comprehensive design methodology enables accelerated and predictable design cycles


XILINX.
Applications
Products
Developer Zone
Support
About
Sign In
Username *
Password *
Forgot your username or password?
Sign In
By signing in, you agree to the Xilinx Terms of Use and Privacy Policy.
Applications
Products
Developer Zone
Support
About Xilinx
SUBSCRIBE
NOW

REPORT –

EDA Playground

In a separate web browser window, log in to EDA Playground at: **http://www.edaplayground.com**

Log in. Click the Log in button (top right) Then either click on Google or Facebook or register by clicking on 'Register for a full account' (which enables all the simulators on EDA Playground)

Select 'Aldec Riviera Pro' from the Tools & Simulators menu. This selects the Aldec Riviera Pro simulator, which can be used however you logged in. Using certain other simulators will require you to have registered for a full account.

In either the Design or Testbench window pane, type in the following code:

```
module test;
  initial
  $display("Hello World!");
endmodule
```

Click Run (top left)

Yes, running a simulation is this

In the bottom pane, we see real-time results as our code is being compiled and then run. A run typically takes 1-5 seconds, depending on network traffic and simulator. Near the bottom of result output, we see:

```
Hello World!
```

Now, Click the Share tab near in the bottom pane and then type in a name and description. Then click Save.

The browser page will reload and the browser address bar will change. This is a persistent link to saved code. We can send the link by email, post it on a web page, post it on Stack Overflow forums, etc. Here is what the link looks like for one user's Hello World! playground: http://www.edaplayground.com/s/3/12

Now, let's try modifying existing code. Load the following example: RAM

On the left editor pane, before the end of initial block, add the following:

```
write_enable = 1;
data_write = 8'h2C;
toggle_clk_write;
toggle_clk_read;
$display ("data[%0h]: %0h",address_read, data_read);
```

Run the sim. In the results you should see this new message:

data[1b]: 2c

Click *Copy* to save a personal version of the modified *RAM* code, including the simulation results.

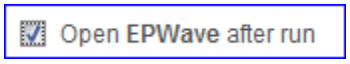
Loading Waves from EDA Playground

You can run a simulation on EDA Playground and load the resulting waves in EPWave.

Loading Waves for System Verilog and Verilog Simulations

- Go to your code on EDA Playground. For example: RAM Design and Test
- Make sure your code contains appropriate function calls to create a *.vcd file. For example:

```
initial begin
 $dumpfile("dump.vcd");
 $dumpvars(1);
End
```

- Select a simulator and check the **Open EPWave after run** checkbox. (Not all simulators may have this run option.)
  - 

- Click **Run**. After the run completes, the resulting waves will load in a new EPWave window. (Pop-ups must be enabled.)

Loading Waves for VHDL Simulation

- Check the **Open EPWave after run** checkbox.
- Specify the **Top entity** to simulate.
- Click **Run**. After the run completes, the resulting waves will load in a new EPWave window. (Pop-ups must be enabled.)
  - The waves for all signals in the specified **Top entity** and any of its components will be dumped.
  - In EPWave window, click **Get Signals** to select the signals to view.

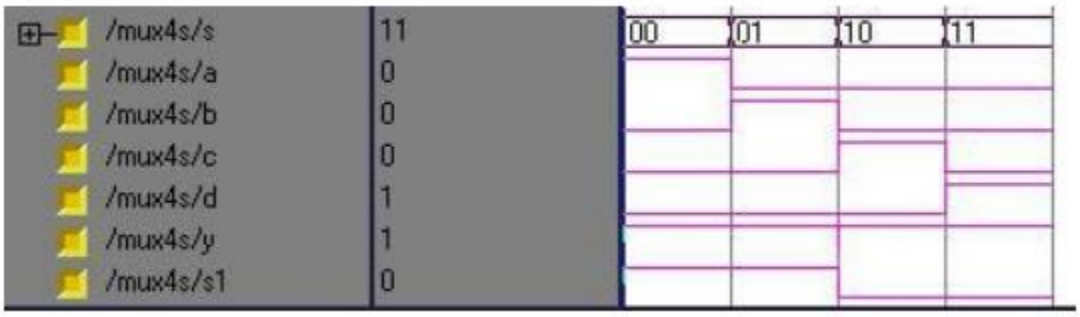
The Vivado Design Suite supports the following established industry design standards:

- Tcl
- AXI4, IP-XACT
- Synopsys design constraints (SDC)
- Verilog, VHDL, VHDL-2008, SystemVerilog
- SystemC, C, C++

The Vivado Design Suite solution is native Tcl based with support for SDC and Xilinx design constraints (XDC) formats. Extensive Verilog, VHDL, and SystemVerilog support for synthesis enables easier FPGA adoption. Vivado High-Level Synthesis (HLS) enables the use of native C, C++, or SystemC languages to define logic. Using standard IP interconnect protocol, such as AXI4 and IP-XACT, enables faster and easier system-level design integration. Support for these industry standards also enables the electronic design automation (EDA) ecosystem to better support the Vivado Design Suite. In addition, many new third-party tools are integrated with the Vivado Design Suite.

Task 3 - Implement 4 to 1 MUX using two 2 to 1 MUX using structural modelling style and test the module in online/offline compiler.

```
module
mux41str(i0,i1,i2,i3,s0,s1,y);
input i0,i1,i2,i3,s0,s1;

wire a,b,c,d;

output
y;

and g1(a,i0,s0,s1);

and g2(b,i1,(~s0),s1);

and g3(c,i2,s0,(~s1));

and g4(d,i3,(~s0),(~s1));

or(y,a,b,c,d);
```

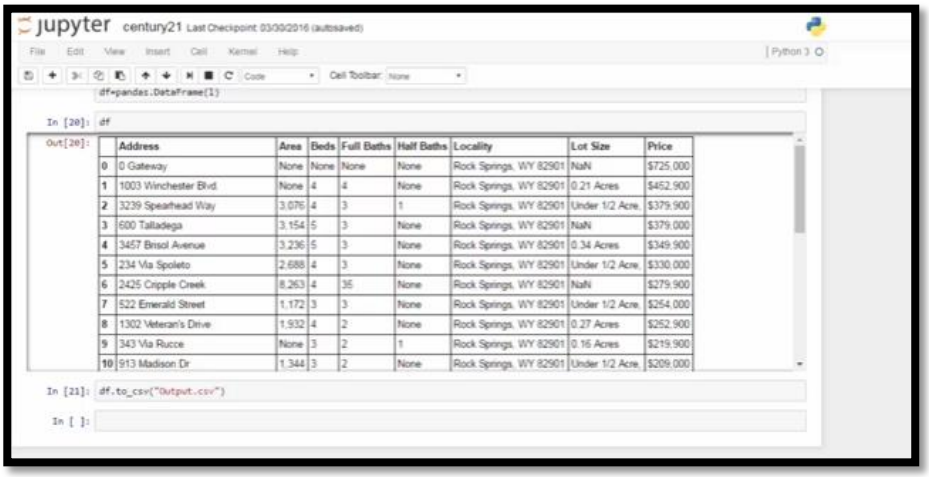


DAILY ASSESSMENT FORMAT

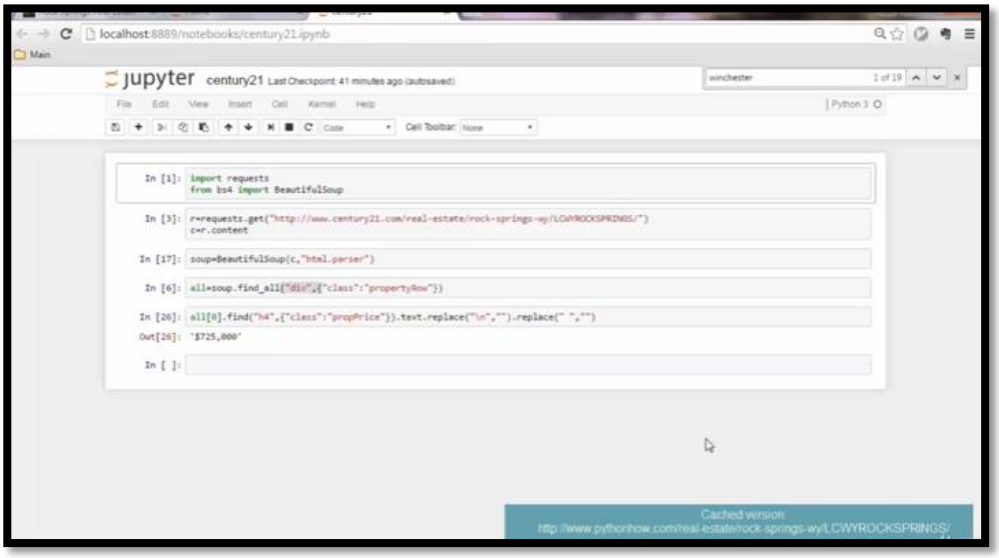
Date:	03/06/2020	Name:	Davis S. Patel
Course:	Python Course	USN:	4AL16EC045
Topic:	Application 7: Scrape Real Estate Property Data from the Web	Semester & Section:	8th - A
GitHub Repository:	Davis		

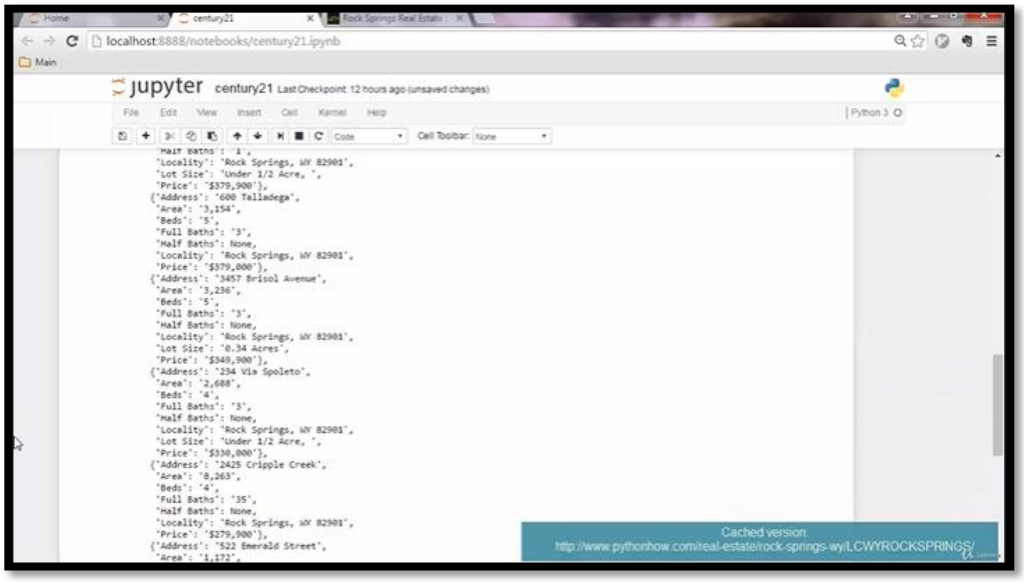
AFTERNOON SESSION DETAILS

Image of Session



	Address	Area	Beds	Full Baths	Half Baths	Locality	Lot Size	Price
0	0 Gateway	None	None	None	None	Rock Springs, WY 82901	NaN	$725,000
1	1003 Winchester Blvd	None	4	4	None	Rock Springs, WY 82901	0.21 Acres	$452,900
2	3239 Spearhead Way	3,076	4	3	1	Rock Springs, WY 82901	Under 1/2 Acre	$379,900
3	600 Talladega	3,154	5	3	None	Rock Springs, WY 82901	NaN	$379,000
4	3457 Brisol Avenue	3,236	5	3	None	Rock Springs, WY 82901	0.34 Acres	$349,900
5	234 Via Spoleto	2,688	4	3	None	Rock Springs, WY 82901	Under 1/2 Acre	$330,000
6	2425 Cripple Creek	8,263	4	3E	None	Rock Springs, WY 82901	NaN	$279,900
7	522 Emerald Street	1,172	3	3	None	Rock Springs, WY 82901	Under 1/2 Acre	$254,000
8	1302 Veteran's Drive	1,932	4	2	None	Rock Springs, WY 82901	0.27 Acres	$252,900
9	343 Via Rucce	None	3	2	1	Rock Springs, WY 82901	0.16 Acres	$219,900
10	913 Madison Dr	1,344	3	2	None	Rock Springs, WY 82901	Under 1/2 Acre	$209,000


localhost:8889/notebooks/century21.ipynb
jupyter century21 Last Checkpoint: 41 minutes ago (autosaved)
In [1]: import requests
from bs4 import BeautifulSoup
In [3]: r=requests.get("http://www.century21.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/")
c=r.content
In [17]: soup=BeautifulSoup(c,"html.parser")
In [6]: all=soup.find_all("div",{"class":"propertyRow"})
In [26]: all[0].find("h4",{"class":"propPrice"}).text.replace("\n","").replace(" ","")
Out[26]: '$725,000'
In [ ]:
Cached version
http://www.pythonhow.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/


localhost:8888/notebooks/century21.ipynb
jupyter century21 Last Checkpoint: 12 hours ago (unsaved changes)
'Locality': 'Rock Springs, WY 82901',
'Lot Size': 'Under 1/2 Acre, ',
'Price': '$379,900'},
{'Address': '600 Talladega',
'Area': '3,154',
'Beds': '5',
'Full Baths': '3',
'Half Baths': None,
'Locality': 'Rock Springs, WY 82901',
'Price': '$379,000'},
{'Address': '3457 Brisol Avenue',
'Area': '3,236',
'Beds': '5',
'Full Baths': '3',
'Half Baths': None,
'Locality': 'Rock Springs, WY 82901',
'Lot Size': '0.34 Acres',
'Price': '$349,900'},
{'Address': '234 Via Spoleto',
'Area': '2,688',
'Beds': '4',
'Full Baths': '3',
'Half Baths': None,
'Locality': 'Rock Springs, WY 82901',
'Lot Size': 'Under 1/2 Acre, ',
'Price': '$338,000'},
{'Address': '2425 Cripple Creek',
'Area': '8,263',
'Beds': '4',
'Full Baths': '35',
'Half Baths': None,
'Locality': 'Rock Springs, WY 82901',
'Price': '$279,900'},
{'Address': '521 Emerald Street',
'Area': '1,172',
Cached version
http://www.pythonhow.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/

REPORT –

Scrape Real Estate Property Data from the Web

The power of data cannot be stressed enough. Anything that is today, and there will be in the future all started somewhere at some granular level with the insights drawn upon data.

With just data, there's nothing much you can do. Essentially, data with quality is out of which remarkable insights are born.

Digital innovations born out of this data has been disrupting every industry. The ways and means of doing business have been transformed incredibly in the past decade or so.

A good, credible, and informative real estate website is one that has a huge database of real estate listings covering wide data points like – property details, buyer and seller information, and agent information. It is the presence of such huge amount of data that helps smarter decision-making absolute ease. A large pool of information that is authentic and credible will help buyers make a more informed decision. To acquire this kind of data from across the internet, real estate data extraction will help in getting all the information that is essential for successful real estate business.

When it comes to large volumes of data that is lying around the web in different formats and different sources, there's no other best solution like scraping that brings all the data hidden almost anywhere. Particularly for real estate data scraping, people search for various aspects – real estate listings, agent information, the price of the property, plot information, seller profiles and a lot more.

To provide the best real estate services, you need to have a repository of data that covers vast data point spread. Also, constantly refreshing this information will make you more reliable. This data could be stuck in websites, classifieds or any other digital source. Scraping this information will help you own the most exhaustive and authentic information that your clients can trust in terms of quality and in making informed decisions.

Some valuable data points to scrape:

- Agent information
- Property data
- Price data
- Address
- Reviews
- Property size
- City/State/Zip code
- Rent price
- Images

Scraping real estate listings would mean setting up web crawlers to scrape the desired data points held in real estate websites and other sources like digital classifieds. The bots would fetch this data and the information will be transformed into a structured format that enables analytics.

This data can be integrated using different formats or through any preferred database options. It could be integrated through FTP/AWS in CSV, XML, and text files.

WEB-SCRAPING USING BEAUTIFUL SOUP

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

– Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application

– Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application

– Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

CODE –

```
#!/usr/bin/python

# -*- coding: utf-8 -*-


import urllib.request

import urllib.parse

import urllib.error

from bs4 import BeautifulSoup

import ssl

import json

import ast

import os

from urllib.request import Request, urlopen


# For ignoring SSL certificate errors


ctx = ssl.create_default_context()

ctx.check_hostname = False

ctx.verify_mode = ssl.CERT_NONE


# Input from user
```

```
url = input('Enter Trulia Product Url- ')

# Making the website believe that you are accessing it using a mozilla browser

req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
webpage = urlopen(req).read()

# Creating a BeautifulSoup object of the html page for easy extraction of data.

soup = BeautifulSoup(webpage, 'html.parser')
html = soup.prettify('utf-8')
product_json = {}

# This code block will get you a one liner description of the listed property

for meta in soup.findAll('meta', attrs={'name': 'description'}):
    try:
        product_json['description'] = meta['content']
        break
    except:
        pass
```

```
# This code block will get you the link of the listed property


for link in soup.findAll('link', attrs={'rel': 'canonical'}):
    try:
        product_json['link'] = link['href']
        break
    except:
        pass


# This code block will get you the price and the currency of the listed property


for scripts in soup.findAll('script',
                            attrs={'type': 'application/ld+json'}):
    details_json = ast.literal_eval(scripts.text.strip())
product_json['price'] = {}
product_json['price']['amount'] = details_json['offers']['price']
product_json['price']['currency'] = details_json['offers'
        ]['priceCurrency']


# This code block will get you the detailed description of the the listed property


for paragraph in soup.findAll('p', attrs={'id': 'propertyDescription'}):
```

```
    product_json['broad-description'] = paragraph.text.strip()
product_json['overview'] = []

# This code block will get you the important points regarding the listed property

for divs in soup.findAll('div',
                         attrs={'data-auto-test-id': 'home-details-overview'
                         }):
    for divs_second in divs.findAll('div'):
        for uls in divs_second.findAll('ul'):
            for lis in uls.findAll('li', text=True, recursive=False):
                product_json['overview'].append(lis.text.strip())

# Creates a json file with all the information that you extracted

with open('house_details.json', 'w') as outfile:
    json.dump(product_json, outfile, indent=4)

# Creates an html file in your local with the html content of the page you parsed.
with open('output_file.html', 'wb') as file:
    file.write(html)
print ('----------Extraction of data is complete. Check json file.----------')
```