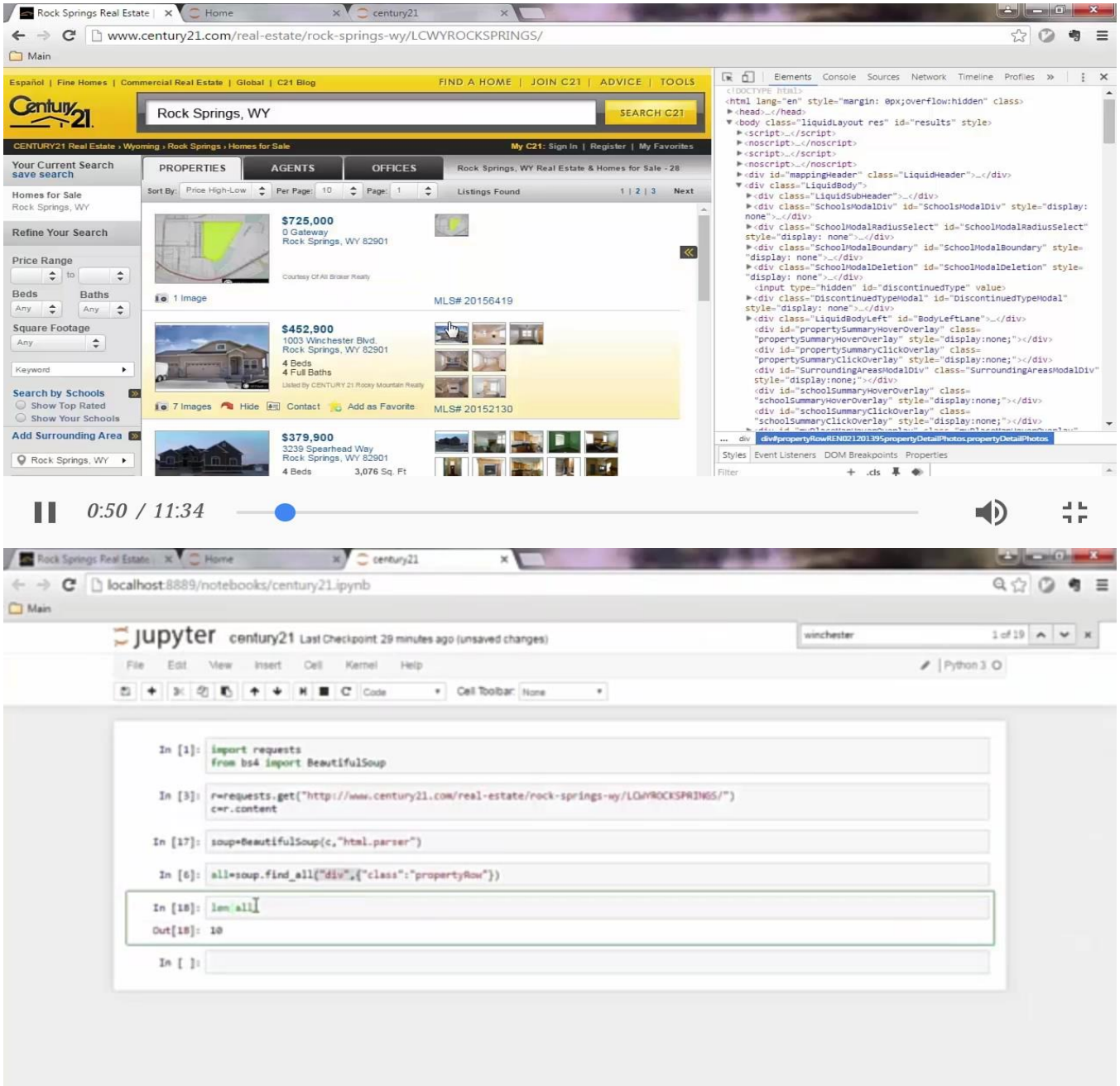
DAILY ASSESSMENT FORMAT

Date:	2nd June 2020	Name:	Soundarya NA
Course:	UDEMY	USN:	4AL16EC077
Topic:	PYTHON: Application 8: Scrape real estate property data from the web	Semester & Section:	8th - B

FORENOON SESSION DETAILS

Image of session



Report:

Introduction:

The real estate market is something that every person living in any country has to deal with, and as a result it makes a great topic about data analytics.

There are 2 ways to capture this data from any of the real estate sites like Zillow, Trulia, ForRent.com etc.

1- Via Rest APIs exposed by the websites but increasingly Rest APIs are either being blocked or tied with paid subscriptions. Also Rest APIs can provide only the data exposed by the website owners so there is limited data/info available.

2- Web-Scraping websites to extract the required information from raw htmls of the web pages. This approach is quite flexible as there is no limit on that data that you can extract. Limitation with this approach is if the website html tags are changed then the code also needs to be updated with that so it needs continuous integration.

Requirement:

- Gather data or text information about the real estate market in San Francisco Bay Area. The location covers but not limited to: San Francisco, San Jose, Pleasanton, Fremont, Hayward, Livermore, Berkeley, Sunnyvale etc.
- The data may include but not limited to: property data (type, size, bedroom/bathroom), price data (current listing price, and or historical selling price), address, city/state/zip code, safety/security (crime rate in neighborhood), agent information (name, company website, and so on), Reviews (review of the property, review of the agent).
- Collect information on at least 600 distinct properties or more from your search. The raw data types may include string, float, integer and so on.

Code:

```
import urllib.request
import urllib.parse
import urllib.error
import ssl
import re
```

```
import pandas as pd
import np
import json
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr
import seaborn as sns

def get_headers():
   #Headers
   headers={'accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,
*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
         'accept-language':'en-US,en;q=0.9',
         'cache-control':'max-age=0',
         'upgrade-insecure-requests':'1',
         'user-agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/80.0.3987.122 Safari/537.36'}
  return headers



ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
count=1 # for pagination
address=[]
rent=[]
sch_crime=[]
sugg_income=[]
add1=[]
area=[]
```

```
bed=[]
bath=[]
floor=[]
commute=[]
descp=[]
addr_link=[]
urls = ["https://www.trulia.com/for_rent/Oakland,CA/1p_beds/SINGLE-FAMILY_HOME_type/",
        "https://www.trulia.com/for_rent/San_Jose,CA/1p_beds/SINGLE-FAMILY_HOME_type/",
      "https://www.trulia.com/for_rent/San_Francisco,CA/1p_beds/SINGLE-FAMILY_HOME_type/",
"https://www.trulia.com/for_rent/Sunnyvale,CA/1p_beds/SINGLE-FAMILY_HOME_type/",
      "https://www.trulia.com/for_rent/Berkeley,CA/1p_beds/SINGLE-FAMILY_HOME_type/",
      "https://www.trulia.com/for_rent/Fremont,CA/1p_beds/SINGLE-FAMILY_HOME_type/",
      "https://www.trulia.com/for_rent/Pleasanton,CA/1p_beds/SINGLE-FAMILY_HOME_type/",
      "https://www.trulia.com/for_rent/Livermore,CA/SINGLE-FAMILY_HOME_type/"]



for x in urls:
   count=1
   y=x
   while(count < 5):  # will go till 4 pages
      print(x)
      req = Request(x, headers=get_headers())  #req all headers
      htmlfile = urlopen(req)
      htmltext = htmlfile.read()
      #print (htmltext)
      soup = BS(htmltext,'html.parser')
      #print (soup.prettify())

      for tag in soup.findAll('div',attrs={'data-testid':'property-price'}): #gets rent
            row = tag.get_text()
```

```
            if not row:
                row="NA"
            print(row)
            rent.append(row)
#for tag in soup.findAll('div',attrs={'class':'Text__TextBase-sc-1i9uasc-0-div Text__TextContainerBase-sc-1i9uasc-1 lcNNgu'}): #gets add
            #row = tag.get_text()
            #print(row)
            #address.append(row)

        for tag in soup.findAll('div',attrs={'data-testid':'property-region'}): #add1
            row = tag.get_text()
            if not row:
                row="NA"
            print(row)
            add1.append(row)

        for tag in soup.findAll('div',attrs={'data-testid':'property-street'}): #area code
            row = tag.get_text()
            if not row:
                row="NA"
            print(row)
            area.append(row)

        for tag in soup.findAll('div',attrs={'data-testid':'property-beds'}): #bed
            row = tag.get_text()
            if not row:
                row="NA"
            print(row)
            bed.append(row)
```

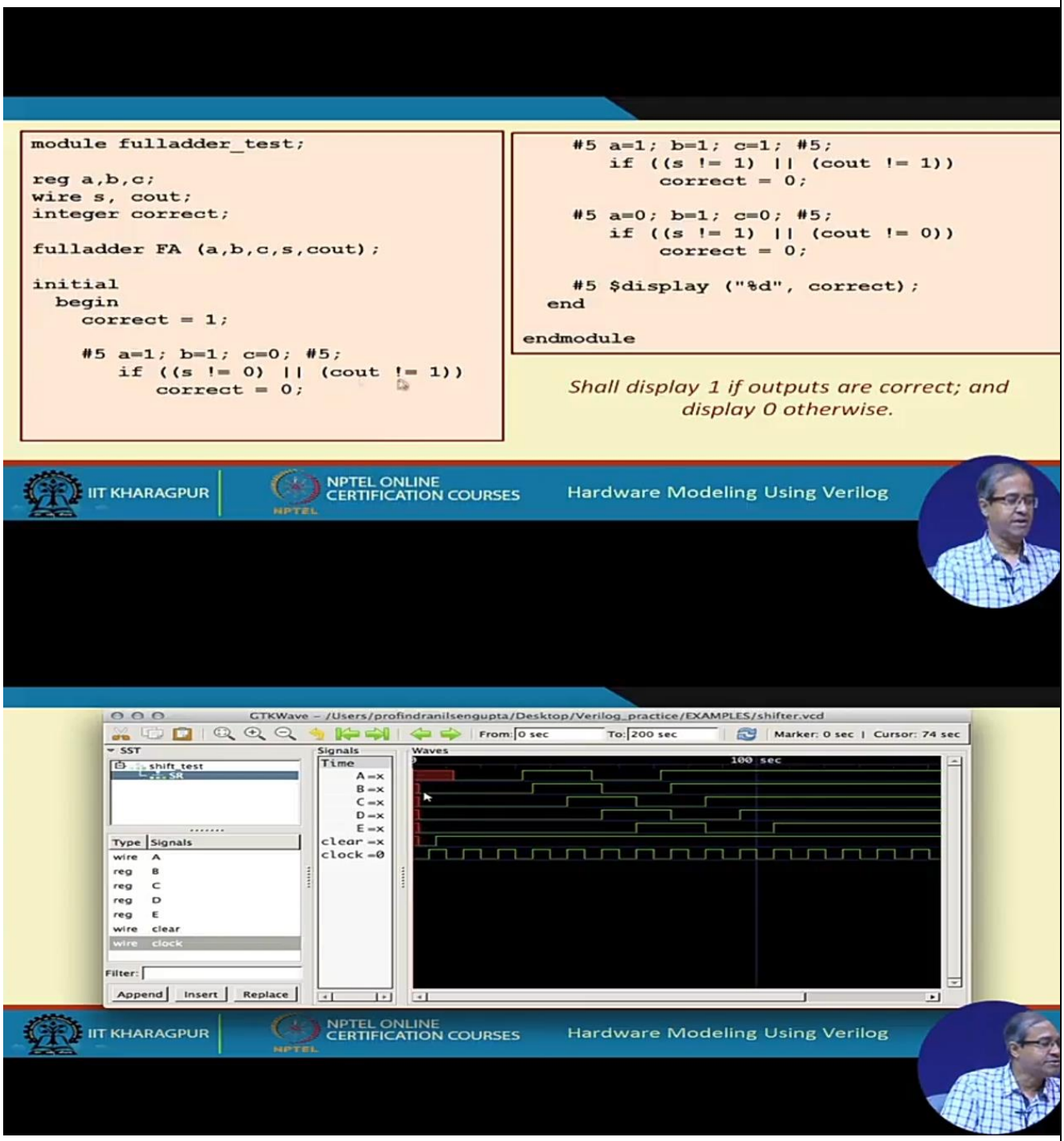
```
for tag in soup.findAll('div',attrs={'data-testid':'property-baths'}): #bath
            row = tag.get_text()
            if not row:
               row="NA"
            print(row)
            bath.append(row)

      for tag in soup.findAll('div',attrs={'data-testid':'property-floorSpace'}):  #floorsize
            row = tag.get_text()
            if not row:
               row="NA"
            print(row)
            floor.append(row)



      links=[]
      for cards in soup.findAll('div',attrs={'class':'Box-sc-8ox7qa-0 jIGxjA
PropertyCard__PropertyCardContainer-sc-1ush98q-2 gKJaNz'}):

         for link in cards.findAll('a', attrs={'href': re.compile("^/")}):
            links.append("https://www.trulia.com"+link.get('href')) #appends all links in the page

      #print(links) # picking up each link and reading inside it
      for link in links:
         addr_link.append(link)
         req = Request(link, headers=get_headers())
htmlfile = urlopen(req)
         htmltext = htmlfile.read()
         #print (htmltext)
         soup = BS(htmltext,'html.parser')  # Reads inside links
```

```
        #print("hello")


        for tag in soup.findAll('div',attrs={'aria-label':'Crime'}):  # crime
            row = tag.get_text()
            if not row:
                row="NA"
            print(row)
            sch_crime.append(row)


        for tag in soup.findAll('span',attrs={'class':'Text__TextBase-sc-1i9uasc-0 fOuqJu'}): # finds suggested income
            row = tag.get_text()
            if not row:
                row="NA"
            print(row)
            sugg_income.append(row)


        for tag in soup.findAll('div',attrs={'data-testid':'explore-the-area-commuteTab'}): #commute
            row = tag.get_text()
            if not row:
                row="NA"
            print(row)
            commute.append(row) #commute
for tag in soup.findAll('div',attrs={'data-testid':'seo-description-paragraph'}): #descp
            row = tag.get_text()


            print(row)
            descp.append(row) #commute


        # add more code here
```

```
        count=count+1
        page=str(count)+"_p"  # changes page,will go till page 4,total 120 links per city
        x=y+page
data_frame =
pd.DataFrame(list(zip(add1,area,rent,bed,bath,floor,descp,commute,sch_crime,sugg_income,
addr_link)),columns=["Address","Location","Rent","Bed","Bath","Size",
"Description","Commute","Crime","Income","URL"])
data_frame
```

Date:	2nd June 2020	**Name:**	Soundarya NA
Course:	Digital Design using HDL	**USN:**	4AL16EC077
Topic:	HDL	**Semester & Section:**	8th - B

Image:



Report:

Linear TestBench is the simplest, fastest and easiest way of writing testbenchs. This became novice verification engineer choice. It is also slowest way to execute stimulus. Typically, linear testbenchs are written in the VHDL or Verilog. In this TestBench, simple linear sequence of test vectors is mentioned. Stimulus code like this is easy to generate translating a vector file with a Perl script, for example. Small models like simple state machine or adder can be verified with this approach. The following code snippet shows linear Testbench. The code snippet shows some input combination only. This is also bad for simulator performance as the simulator must evaluate and schedule a very large number of events. This reduces simulation performance in proportion to the size of the stimulus process.

Typically, linear testbenchs perform the following tasks:

1. Instantiate the design under test (DUT)
2. Stimulate the DUT by applying test vectors.
3. Output results waveform window or to a terminal for visual inspection manually.

Example:

```
module adder(a,b,c);
input [15:0] a;
input [15:0] b;
output [16:0] c;
assign c = a + b;
endmodule
module top();
reg [15:0] a;
reg [15:0] b;
wire [16:0] c;
adder DUT(a,b,c);
initial
```

```
begin
a = 16'h45;
b = 16'h12;
#10 $display("a=%0d,b=%0d,c=%0d",a,b,c);
end
endmodule //TestBench code end
```

To test all possible scenarios which are known to us, it is not an easy task. Development time increases exponentially as the number of scenarios increases and maintain them is nightmare. Instead of listing out all the possible scenarios, pickup some randomly and check the DUT.

Implement a 4:1 mux and write the test bench code to verify the module:

Gate level modelling:

```
module m41(out, a, b, c, d, s0, s1);
output out;
input a, b, c, d, s0, s1;
wire sobar, s1bar, T1, T2, T3, T4;
not (s0bar, s0), (s1bar, s1);
and (T1, a, s0bar, s1bar), (T2, b, s0bar, s1),(T3, c, s0, s1bar), (T4, d, s0, s1);
or(out, T1, T2, T3, T4);
endmodule
```

Data flow modelling:

```
module m41 ( input a,
input b,
input c,
input d,
input s0, s1,
output out);
 assign out = s1 ? (s0 ? d : c) : (s0 ? b : a);
endmodule
```

Behavioral modelling:

```
module m41 ( a, b, c, d, s0, s1, out);
input wire a, b, c, d;
input wire s0, s1;
output reg out;
always @ (a or b or c or d or s0, s1)
begin
case (s0 | s1)
2'b00 : out <= a;
```

```
2'b01 : out <= b;
2'b10 : out <= c;
2'b11 : out <= d;
endcase
end
endmodule
```

Structural modelling:

```
module and_gate(output a, input b, c, d);
assign a = b & c & d;
endmodule
module not_gate(output f, input e);
assign e = ~ f;
endmodule
module or_gate(output l, input m, n, o, p);
assign l = m | n | o | p;
endmodule
module m41(out, a, b, c, d, s0, s1);
output out;
input a, b, c, d, s0, s1;
wire s0bar, s1bar, T1, T2, T3;
not_gate u1(s1bar, s1);
not_gate u2(s0bar, s0);
and_gate u3(T1, a, s0bar, s1bar);
and_gate u4(T2, b, s0, s1bar);
and_gate u5(T3, c, s0bar, s1);
and_gate u6(T4, d, s0, s1);
or_gate u7(out, T1, T2, T3, T4);
endmodule
```

Testbench:

```
module top;
wire  out;
reg  a;
reg  b;
reg  c;
reg  d;
reg s0, s1;
m41 name(.out(out), .a(a), .b(b), .c(c), .d(d), .s0(s0), .s1(s1));
 initial
 begin
 a=1'b0; b=1'b0; c=1'b0; d=1'b0;
 s0=1'b0; s1=1'b0;
 #500 $finish;
end
always #40 a=~a;
always #20 b=~b;
always #10 c=~c;
always #5 d=~d;
always #80 s0=~s0;
always #160 s1=~s1;
always@(a or b or c or d or s0 or s1)
$monitor("At time = %t, Output = %d", $time, out);
endmodule;
```

Simulation waveform: