**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **03/06/20** | **Name:** | **Gaganashree P** |
| **Course:** | **Digital designing using hdl** | **USN:** | **4al15ec024** |
| **Topic:** | **About EDA** | **Semester & Section:** | **8th & a** |
| **Github Repository:** | **Gaganashree-P** |  |  |

|  |
| --- |
| **MORNING SESSION DETAILS** |
| **Image of session** |
| **Report –**  **EDA Playground Help**  Quick Start   * 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 your language from the Testbench + Design menu. * Select your simulator from the Tools & Simulators menu. Using certain simulators will require you to supply additional identifcation information. * Type in your code in the testbench and design windows. * Click Run. * Tutorial <http://eda-playground.readthedocs.io/en/latest/tutorial.html>   **What is EDA Playground?**   * EDA Playground gives engineers immediate hands-on exposure to simulating SystemVerilog, Verilog, VHDL, C++/SystemC, and other HDLs. All you need is a web browser. * The goal is to accelerate learning of design/testbench development with easier code sharing and simpler access to EDA tools and libraries. * With a simple click, run your code and see console output in real time. * View waves for your simulation using EPWave browser-based wave viewer.   Save your code snippets (“Playgrounds”).   * Share your code and simulation results with a web link. Perfect for web forum discussions or emails. Great for asking questions or sharing your knowledge.     **EDA Playground Tutorial Demo Video**    **How to Download And Install Xilinx Vivado Design Suite**  Vivado Design Suite - HLx Editions Update 1 - 2019.2  Important Information  This is a common updater. You do not need to re-run it for Vitis if you have already run it for Vivado and vice versa.  Vivado Design Suite 2019.2.1 is now available with support for:  Additional Zynq UltraScale+ RFSoCs devices enabled:- (XCZU46DR, XCZU47DR, XCZU48DR, XCZU49DR)  For customers using these devices, Xilinx recommends installing Vivado 2019.2.1. For other devices, please continue to use Vivado 2019.2.  Note: Download verification is only supported with Google Chrome and Microsoft Internet Explorer web browsers.    **Vivado Design Suite for implementation of HDL code**  Introduction  Synthesis is the process of transforming an RTL-specified design into a gate-level  representation. Vivado® synthesis is timing-driven and optimized for memory usage and  performance. Vivado synthesis supports a synthesizeable subset of:  • SystemVerilog: IEEE Standard for SystemVerilog-Unified Hardware Design,  Specification, and Verification Language (IEEE Std 1800-2012)  • Verilog: IEEE Standard for Verilog Hardware Description Language (IEEE Std 1364-2005)  • VHDL: IEEE Standard for VHDL Language (IEEE Std 1076-2002)  • VHDL 2008  • Mixed languages: Vivado supports a mix of VHDL, Verilog, and SystemVerilog.  In most instances, the Vivado tools also support Xilinx design constraints (XDC), which is  based on the industry-standard Synopsys design constraints (SDC).  Synthesis Methodology  The Vivado IDE includes a synthesis and implementation environment that facilitates a push button flow with synthesis and implementation runs. The tool manages the run data automatically, allowing repeated run attempts with varying Register Transfer Level (RTL) source versions, target devices, synthesis or implementation options, and physical or timing constraints.  Within the Vivado IDE, you can do the following:  • Create and save a strategy. A strategy is a configuration of command options that you can apply to design runs for synthesis or implementation. See Creating Run Strategies.  • Queue the synthesis and implementation runs to launch sequentially or simultaneously with multi-processor machines. See Running Synthesis.  • Monitor synthesis or implementation progress, view log reports, and cancel runs. See Monitoring the Synthesis Run.    **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 mux4to1(a,sel,out);  input [3:0] a;  input [1:0] sel;  output out;  wire mux[2:0];  mux2to1 m1 (a[3],a[2],sel[0],mux\_1),  m2 (a[1],a[4],sel[0],mux\_2),  m3 (mux\_1,mux\_2,sel[1],out);  endmodule. |

**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Date:** | **3-6-2020** | **Name:** | **Gaganashree P** | | **Course:** | **Python programming** | **USN:** | **4AL15EC024** | | **Topic:** | **1.Scrape real estate property data from the web** | **Semester & Section:** | **8th A** | | **Github Repository:** | **Gaganashree-P** |  |  | |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
| **Image of session** |
| The leading real estate sites of the world are a treasure trove of valuable data. The database of any of popular real estate site targeting the US might contain information on more than 100 million homes. These homes include the ones for sale, rent, or even ones not currently on the market. It provides rent and property estimates called “Zestimates” as well. It helps owners, as well as customers, plan better by trying to estimate the prices of properties in the next one, five or even ten years.  When it comes to buying or renting properties, we know that the first thing that comes to one’s mind is price comparison. These sites for housing provide price comparison with all listings in that area, as well as basic information like the type of house it is, number of rooms, the size, a short description, etc. You can even get new estimates for a property if a certain change has been recently made – for example, say a swimming pool has been added in the backyard, or the kitchen has been remodeled. These portals also provide multiple APIs for developer networks.  It partnered with Microsoft to provide a bird’s eye view of famous properties. These consist of pictures that are taken from airplanes and these pictures are far superior to the ones taken from satellites.  **Why scrape data from real estate sites?**  The large property listing companies target an entire nation and work on millions of properties. But in case you are a real estate agent, or if you are setting up shop and targeting a specific state or region, it is better that instead of trying to gather data yourself, you scrape it from a major real estate listing website.  You can also build Machine Learning models to predict the prices of properties and compare your predictions with Zillow™’s Zestimates™ and see which one is better or closer to real values.  **How to set things up?**  In case you have followed any of our previous “How to scrape” articles, you might already have the necessary setup ready on your computers. In case you have not, I recommend you to follow this [article](https://www.promptcloud.com/blog/tutorial-how-to-scrape-amazon-product-details-prices-using-python/) to set up Python, its packages and the text editor before you can get your hands dirty with the code.  **Where is the code?**  Without much ado, we decided to bring you the code for the scraper that will help you extract information from a property listing. It is written in Python and subsequently, I will show you how to run it and what you will get once you run it.  #!/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 Zillow House Listing 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')  property\_json = {}  property\_json['Details\_Broad'] = {}  property\_json['Address'] = {}  # Extract Title of the property listing  for title in soup.findAll('title'):  property\_json['Title'] = title.text.strip()  break  for meta in soup.findAll('meta', attrs={'name': 'description'}):  property\_json['Detail\_Short'] = meta['content'].strip()  for div in soup.findAll('div', attrs={'class': 'character-count-truncated'}):  property\_json['Details\_Broad']['Description'] = div.text.strip()  for (i, script) in enumerate(soup.findAll('script',  attrs={'type': 'application/ld+json'})):  if i == 0:  json\_data = json.loads(script.text)  property\_json['Details\_Broad']['Number of Rooms'] = json\_data['numberOfRooms']  property\_json['Details\_Broad']['Floor Size (in sqft)'] = json\_data['floorSize']['value']  property\_json['Address']['Street'] = json\_data['address']['streetAddress']  property\_json['Address']['Locality'] = json\_data['address']['addressLocality']  property\_json['Address']['Region'] = json\_data['address']['addressRegion']  property\_json['Address']['Postal Code'] = json\_data['address']['postalCode']  if i == 1:  json\_data = json.loads(script.text)  property\_json['Price in $'] = json\_data['offers']['price']  property\_json['Image'] = json\_data['image']  break  with open('data.json', 'w') as outfile:  json.dump(property\_json, outfile, indent=4)  with open('output\_file.html', 'wb') as file:  file.write(html)  print ('----------Extraction of data is complete. Check json file.----------')  To run the code given above, you need to save it in a file with the extension, such as propertyScraper.py. Once that is done, from the terminal, run the command –  python propertyScraper.py  When you run it, you will be prompted to enter the URL of a property listing. This is the webpage that will actually be scraped by the program. We have used two links and scraped the data of two properties. Here are the links –   1. https://www.zillow.com/homedetails/638-Grant-Ave-North-Baldwin-NY-11510/31220792\_zpid/ 2. https://www.zillow.com/homedetails/10-Walnut-St-Arlington-MA-02476/56401372\_zpid/   The JSON files obtained on running the code on the given in a later subtopic.  **Code explanation**  Before going into how the code runs and what it returns, it is important to understand the code itself. As usual, we first hit the URL given and capture the entire HTML which we convert into beautiful soup object. Once that is done, we extract specific divs, scripts, titles, and other tags with specific attributes. This way we are able to pinpoint specific information that we may want to extract from a page.  You can see that we have also extracted an image link for each property. This has been done deliberately since for something like real estate, images are just as much value as other information. While we have indeed extracted several fields from the real estate listing pages, it is to be noted that the HTML page does contain many more data points. Hence we are also saving the HTML content locally so that you can go through it and scrape more information.  Report – Report can be typed or hand written for up to two pages.  Request Headers  Note  Whenever I use this code in the next videos:  r = requests.get("http://www.pythonhow.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/")  please use this instead:  r = requests.get("http://www.pyclass.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/", headers={'User-agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:61.0) Gecko/20100101 Firefox/61.0'})  The rest of the code stays the same.  So, we're just changing the domain name from pythonhow to pyclass and we're adding a header argument. Some webpages don't like scripts sometimes, so adding a header allows the script to impersonate a web browser. |