**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **03-06-2020** | **Name:** | **Karthik J** |
| **Course:** | **DSDV** | **USN:** | **4AL16EC030** |
| **Topic:** | About EDA | **Semester & Section:** | **8TH A** |
| **GitHub Repository:** | Karthik-J |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| EDA Playground has a significant part to play in Doulos' commitment to invest in training the next generation of electronic system designers. EDA Playground is a free-to-access online learning resource where engineers can practice their coding skills and share code snippets with others in the community," says **Rob Hurley Doulos CEO**  As we look back proudly on 25 years of delivering engineering know-how to engineers worldwide, it's exciting that an EDA and semiconductor IP leader like Synopsys is supporting us in ensuring that aspiring designers can develop their skills by accessing such market-leading solutions.   “This collaboration will enable broader and deeper adoption of key subjects like SystemVerilog and UVM,“ says **Michael Sanie, Synopsys Senior Director of Verification Marketing.** “Synopsys is supportive of initiatives from partners such as Doulos that will assist more customers to become effective and productive in addressing their verification challenges.“  Doulos plans include innovations and extensions to EDA Playground in support of the roll-out of online training and blended learning solutions. “Today's engineers can access so much of the information they need online,“ says **John Aynsley, Doulos CTO**, “To some extent that goes for acquiring knowledge as well; but productive design requires engineers that are highly skilled; and skills are only sharpened through hands-on practise with the support of experts who know how to teach. The best of tomorrow's training solutions will blend all of the above. EDA Playground will play a key part in enabling this.“  Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL). It is a language used for describing a digital system like a network switch or a microprocessor or a memory or a flip−flop. It means, by using a HDL we can describe any digital hardware at any level. Designs, which are described in HDL are independent of technology, very easy for designing and debugging, and are normally more useful than schematics, particularly for large circuits.  Verilog supports a design at many levels of abstraction. The major three are −   * Behavioral level * Register-transfer level * Gate level  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](http://epwave.readthedocs.org) 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. * Quickly try something out   + Try out a language feature with a small example.   + Try out a library that you’re thinking of using.  Example Usecases  * **Quick prototyping** – try out syntax or a library/language feature. * When **asking questions on** [Stack Overflow](http://stackoverflow.com/) or other online forums, attach a link to the code and simulation results. * Use during **technical interviews** to test candidates’ SystemVerilog/Verilog coding and debug skills. * Try verifying using **different verification frameworks**: UVM, SVUnit, plain Verilog, or Python.  Tools & Simulators For settings and options documentation, see [Tools & Simulators Options](https://eda-playground.readthedocs.io/en/latest/settings.html#tools-simulators-options-label)  Available tools and simulators are below. EDA Playground can support many different tools. [Contact us](http://www.doulos.com) to add your EDA tool to EDA Playground.  Playground About EDA Playground EDA Playground gives engineers immediate hands-on access to simulating small amounts of System Verilog, Verilog, VHDL, C++, System C and other HDLs. All that's needed is a web browser. The goal is to accelerate learning of design and test-bench development with easier code sharing, and with simple access to EDA tools and libraries. EDA Playground is specifically designed for small prototypes and examples. It was developed by Victor Lyuboslavsky of Victor EDA, Inc and was acquired by Doulos in early 2015. Its growing user community is testimony to a great idea well executed. Tools currently supported include solutions from Aldec, Inc and Cadence, Inc. Synopsys VCS support will be available in the Q3 release.  EDA Playground Logo |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date:** | | **03-06-2020** | **Name:** | **Karthik J** |  |
| **Course:** | | [Programming with Python: Hands-on Introduction for Beginners](https://www.udemy.com/course/python-programming-beginners/) | **USN:** | **4AL16EC030** |  |
| **Topic:** | |  | **Semester & Section:** | **8th A** |  |
|  | **AFTERNOON SESSION DETAILS** | | | | |
|  | **Image of session** | | | | |
|  | Dictionary A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values. Example Create and print a dictionary:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } print(thisdict) Python - Tuples A tuple is an immutable sequence of Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.  Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also.  For example −  tup1 = ('physics', 'chemistry', 1997, 2000);  tup2 = (1, 2, 3, 4, 5 );  tup3 = "a", "b", "c", "d"; Accessing Values in Tuples To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index.  For example −  tup1 = ('physics', 'chemistry', 1997, 2000);  tup2 = (1, 2, 3, 4, 5, 6, 7 );  print "tup1[0]: ", tup1[0];  print "tup2[1:5]: ", tup2[1:5]; Updating Tuples Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples  Example  tup1 = (12, 34.56);  tup2 = ('abc', 'xyz');  # Following action is not valid for tuples  # tup1[0] = 100;  # So, let's create a new tuple as follows  tup3 = tup1 + tup2;  print tup3; Delete Tuple Elements Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.  To explicitly remove an entire tuple, just use the **del** statement.  example −  tup = ('physics', 'chemistry', 1997, 2000);  print tup;  del tup;  print "After deleting tup : ";  print tup;  This produces the following result. Note an exception raised, this is because after **del tup** tuple does not exist any-more − Basic Tuples Operations Tuples respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string. Indexing, Slicing, and Matrixes Because tuples are sequences, indexing and slicing work the same way for tuples as they do for strings. Assuming following input −  L = ('spam', 'Spam', 'SPAM!') | | | | |