**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **23/05/2020** | **Name:** | **DHAMINI C L** |
| **Course:** | **Python** | **USN:** | **4AL17EC025** |
| **Topic:** | **Loops programs** | **Semester & Section:** | **6TH & A** |
| **Github Repository:** | **DHAMINI-CL-Course** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report – Report can be typed or hand written for up to two pages.**  A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).  This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.  With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.  fruits = ["apple", "banana", "cherry"] for x in fruits:   print(x)  Looping Through a String  Even strings are iterable objects, they contain a sequence of characters:  for x in "banana":   print(x)  The break Statement  With the break statement we can stop the loop before it has looped through all the items:  fruits = ["apple", "banana", "cherry"] for x in fruits:   print(x)   if x == "banana":     break  fruits = ["apple", "banana", "cherry"] for x in fruits:   if x == "banana":     break   print(x)  The continue Statement  With the continue statement we can stop the current iteration of the loop, and continue with the next:  fruits = ["apple", "banana", "cherry"] for x in fruits:   if x == "banana":     continue   print(x) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:23/5/2020** |  | **Name: DHAMINI C L** |  | |
| **Course:PYTHON** |  | **USN:4AL17EC025** |  | |
| **Topic: List Comprehensions** |  | **Semester & Section:6TH A SEC** |  | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Report – Report can be typed or hand written for up to two pages.**  .  With the recap of the Python lists fresh in mind, you can easily see that defining and creating lists in Python can be a tiresome job: typing in all the values separately can take quite some time and you can easily make mistakes.  But how do you get to this formula-like way of building and using these constructs in Python? Let's dig a little bit deeper. List Comprehension in Python: The Mathematics Luckily, Python has the solution for you: it offers you a way to implement a mathematical notation to do this: list comprehension.  **Remember** in maths, the common ways to describe lists (or sets, or tuples, or vectors) are:  In other words, you'll find that the above definitions actually tell you the following:   * The sequence S is actually a sequence that contains values between 0 and 9 included that are raised to the power of two. * The sequence V, on the other hand, contains the value 2 that is raised to a certain power. For the first element in the sequence, this is 0, for the second this is 1, and so on, until you reach 12. * Lastly, the sequence M contains elements from the sequence S, but only the even ones.   If the above definitions give you a headache, take a look at the actual lists that these definitions would produce:  of each list and the operations that were described in them! | | | |