**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Date:** | **18-5-2020** | **Name:** | **Russell D’souza** | | **Course:** | **Python programming** | **USN:** | **4AL15EC023** | | **Topic:** | **List comprehension,Functions,imported modules** | **Semester & Section:** | **8th A** | | **Github Repository:** | **Russell1005** |  |  | |
|  |
|  |
| **AFTERNOON SESSION DETAILS** | |
| **Image of session** | |
| **Report – Report can be typed or hand written for up to two pages.**  In my session session I have studied about the basics of anaconda in Python and difference between the different versions of Python.  Python 3.0:  Python 2 made code development process easier than earlier versions. It implemented technical details of Python Enhancement Proposal (PEP). Python 2.7 (last version in 2.x ) is no longer under development and in 2020 will be discontinued.  On December 2008, Python released version 3.0. This version was mainly released to fix problems which exist in Python 2. The nature of these change is such that Python 3 was incompatible with Python 2. It is backward incompatible Some features of Python 3 have been backported to Python 2.x versions to make the migration process easy in Python 3.  As a result, for any organization who was using Python 2.x version, migrating their project to 3.x needed lots of changes. These changes not only relate to projects and applications but also all the libraries that form part of the Python ecosystem.  prime reasons for using Python 3.x versions:   * Python 3 supports modern techniques like AI, machine learning, and data science * Python 3 is supported by a large Python developer's community. Getting support is easy. * Its easier to learn Python language compared to earlier versions. * Offers Powerful toolkit and libraries * Mixable with other languages   When it comes to Python version 2 vs. 3 today, Python 3 is the outright winner. That's because Python 2 won't be available after 2020. Mass Python 3 adoption is the clear direction of the future.  After considering declining support for Python 2 programming language and added benefits from upgrades to Python 3, it is always advisable for a new developer to select Python version 3. However, if a job demands Python 2 capabilities, that would be an only compelling reason to use this version.  KEY DIFFERENCE:   * Python 3 syntax is simpler and easily understandable whereas Python 2 syntax is comparatively difficult to understand. * Python 3 default storing of strings is Unicode whereas Python 2 stores need to define Unicode string value with "u." * Python 3 value of variables never changes whereas in Python 2 value of the global variable will be changed while using it inside for-loop.   **Benefits of Using List Comprehensions**  List comprehensions are often described as being more [Pythonic](https://realpython.com/courses/idiomatic-python-101/) than loops or map(). But rather than blindly accepting that assessment, it’s worth it to understand the benefits of using a list comprehension in Python when compared to the alternatives. Later on, you’ll learn about a few scenarios where the alternatives are a better choice.  One main benefit of using a list comprehension in Python is that it’s a single tool that you can use in many different situations. In addition to standard [list creation](https://realpython.com/python-lists-tuples/), list comprehensions can also be used for mapping and filtering. You don’t have to use a different approach for each scenario.  This is the main reason why list comprehensions are considered Pythonic, as Python embraces simple, powerful tools that you can use in a wide variety of situations. As an added side benefit, whenever you use a list comprehension in Python, you won’t need to remember the proper order of arguments like you would when you call map().  List comprehensions are also more declarative than loops, which means they’re easier to read and understand. Loops require you to focus on how the list is created. You have to manually create an empty list, loop over the elements, and add each of them to the end of the list. With a list comprehension in Python, you can instead focus on *what* you want to go in the list and trust that Python will take care of *how* the list construction takes place.  **FUNCTIONS IN PYTHON**  In Python, a function is a group of related statements that performs a specific task.  Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.  Furthermore, it avoids repetition and makes the code reusable.  **Syntax of Function**  def function\_name(parameters):  """docstring"""  statement(s)  Above shown is a function definition that consists of the following components.   1. Keyword def that marks the start of the function header. 2. A function name to uniquely identify the function. Function naming follows the same [rules of writing identifiers in Python](https://www.programiz.com/python-programming/keywords-identifier#rules). 3. Parameters (arguments) through which we pass values to a function. They are optional. 4. A colon (:) to mark the end of the function header. 5. Optional documentation string (docstring) to describe what the function does. 6. One or more valid python statements that make up the function body. Statements must have the same indentation level (usually 4 spaces). 7. An optional return statement to return a value from the function. | |