

DAILY ASSESSMENT FORMAT

Date:	24th July 2020	Name:	K B KUSHI
Course:	workshop	USN:	4AL17EC107
Topic:	<ul style="list-style-type: none"> How to develop Pythonic coding rather than Python coding Certificate 	Semester & Section:	6 & B
GitHub Repository:	KUSHI-COURSES		

SESSION DETAILS

Session images

The screenshot displays a Google Meet interface during a session titled "Pythonic workshop Day 4 Session 1.ipynb". The presenter, Badhusha Mohideen, is sharing a Jupyter Notebook. The notebook content shows a code snippet for joining a list of strings without using a for loop, adhering to PEP 8 rules. The code is as follows:

```
result_list = ['True', 'False', 'File not found']
result_string = ''.join(result_list)
```

A notification bubble indicates that many people are in the session. The Meet interface shows 156 participants, with a list of names including Persis, Ritika kulkarni_4A, Jagadeesha Hegde, anusha k, Sneha G, and Madhu Basavaraj. The bottom of the screen shows the Windows taskbar with the search bar and various application icons.

The screenshot shows a Google Meet interface. At the top, the browser tabs include 'Invitation: Day 4 Online worksho...', 'Meet - Day 4 Online worksho...', '1. Mathematical and Physical Mo...', and 'alvas-education-foundation/Sus...'. The Meet header shows 'Badhusha Mohideen is presenting' and a participant list including 'SHRADDHA AC... and 155 more'. The main content area displays a Google Colab notebook titled 'Pythonic workshop Day 4 Session 1.ipynb'. The notebook code is as follows:

```
def myFunc(e):
    return len(e)
myfunc=lambda x : len(x)
cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']
cars.sort(key=myFunc)
print(cars)
```

The notebook interface includes tabs for '+ Code' and '+ Text', and a status bar at the bottom of the notebook showing 'RAM' and 'Disk' usage. The Meet sidebar on the right lists participants: Persis, Ritika kulkarni_4A, Jagadeesha Hegde, RASHMITHA POO., Karthik S, and Sinchana K N 4a1. The bottom of the screen shows the Windows taskbar with the search bar and various application icons.

Report:

A common neologism in the Python community is **pythonic**, which can have a wide range of meanings related to program style. To say that code is **pythonic** is to say that it uses Python idioms well, that it is natural or shows fluency in the language. Likewise, to say of an interface or language feature that it is **pythonic** is to say that it works well with Python idioms, that its use meshes well with the rest of the language.

Python scripts can put the system into different states, set configurations, and test all sorts of real-world use cases. Python can also be used to receive embedded system data that can be stored for analysis. Programmers can then use Python to develop parameters and other methods of analyzing that data.

There are certain things you can do with all sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership. In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements.

One of the special concepts in Python is the idea of writing idiomatic code that is most aligned with the language features and ideals. In Python, we call this idiomatic code **Pythonic**. While this idea is easy to understand, it turns out to be fairly hard to make concrete.

This course will take you on a tour of over 50 of the more popular and useful code examples demonstrating examples of **Pythonic** code. In the examples, you'll first see non-Pythonic code and then the more natural **Pythonic** version. Topics covered include the expansive use of dictionaries, hacking Python's memory usage via slots, using generators, comprehensions, and generator expressions, creating subsets of collections via slices (all the way to the database) and more.

Several of these are Python 3 features so you'll have even more reason to adopt Python 3 for your next project.