**Name -> ROHAN NYATI**

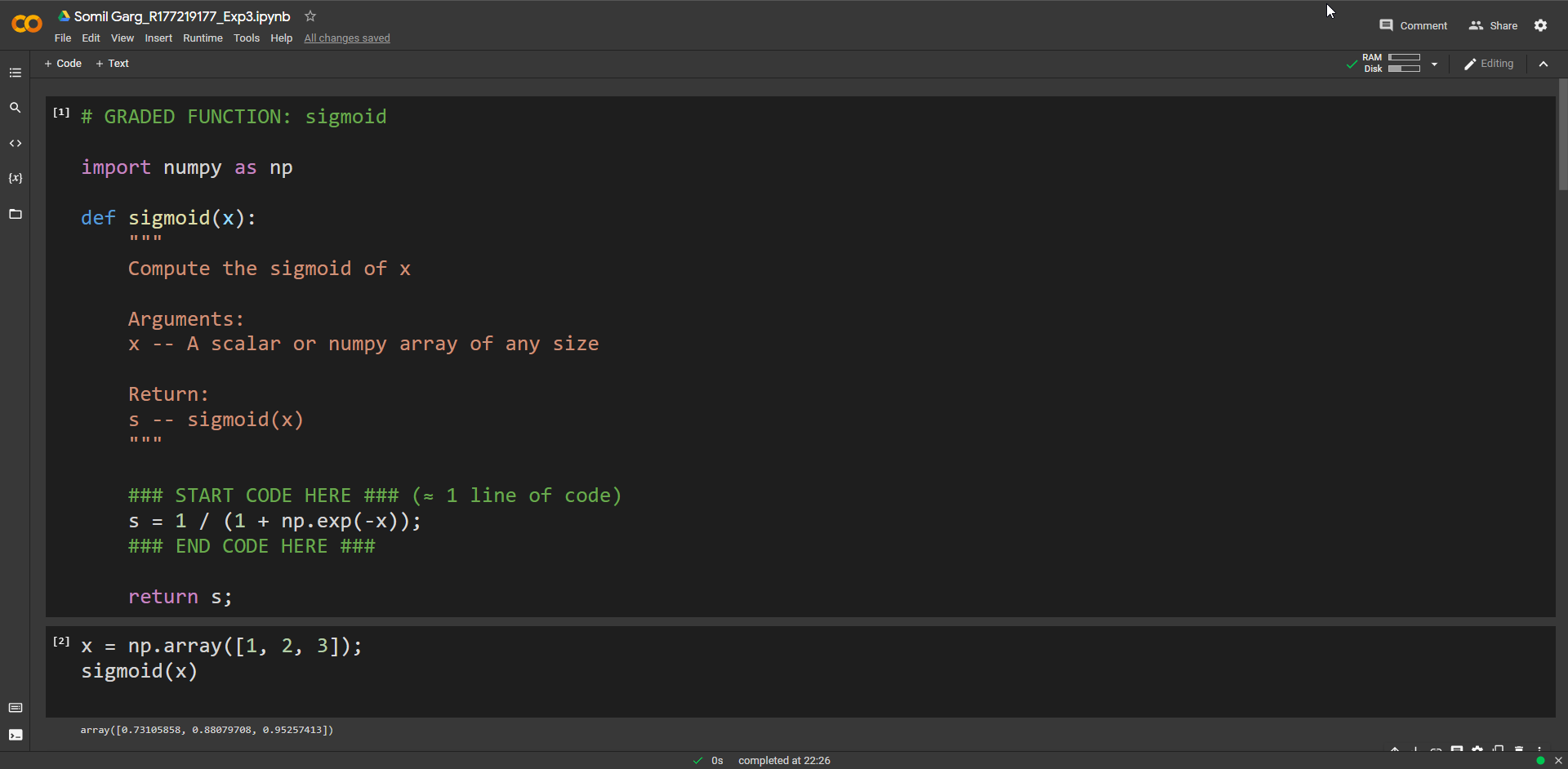
**Roll No -> R177219148**

**Sap Id -> 500075940**

**Course -> B-Tech CSE AI&ML B5**

**Experiment -> 3**

1. **Exercise: Implement the sigmoid function using numpy. Instructions: x could now be either a real number, a vector, or a matrix.**

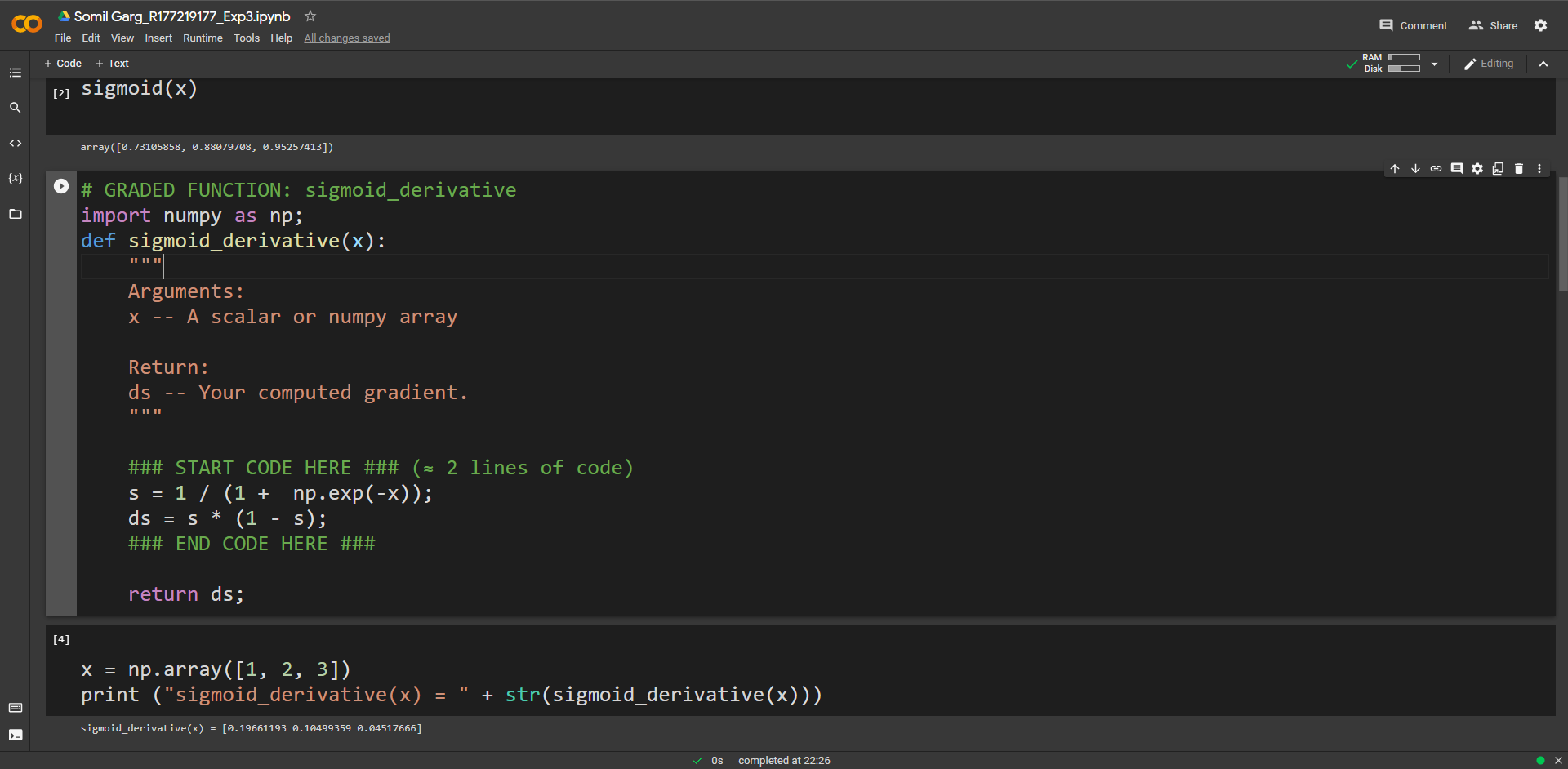


1. **Exercise: Implement the function sigmoid\_grad() to compute the gradient of the sigmoid function with respect to its input x.**

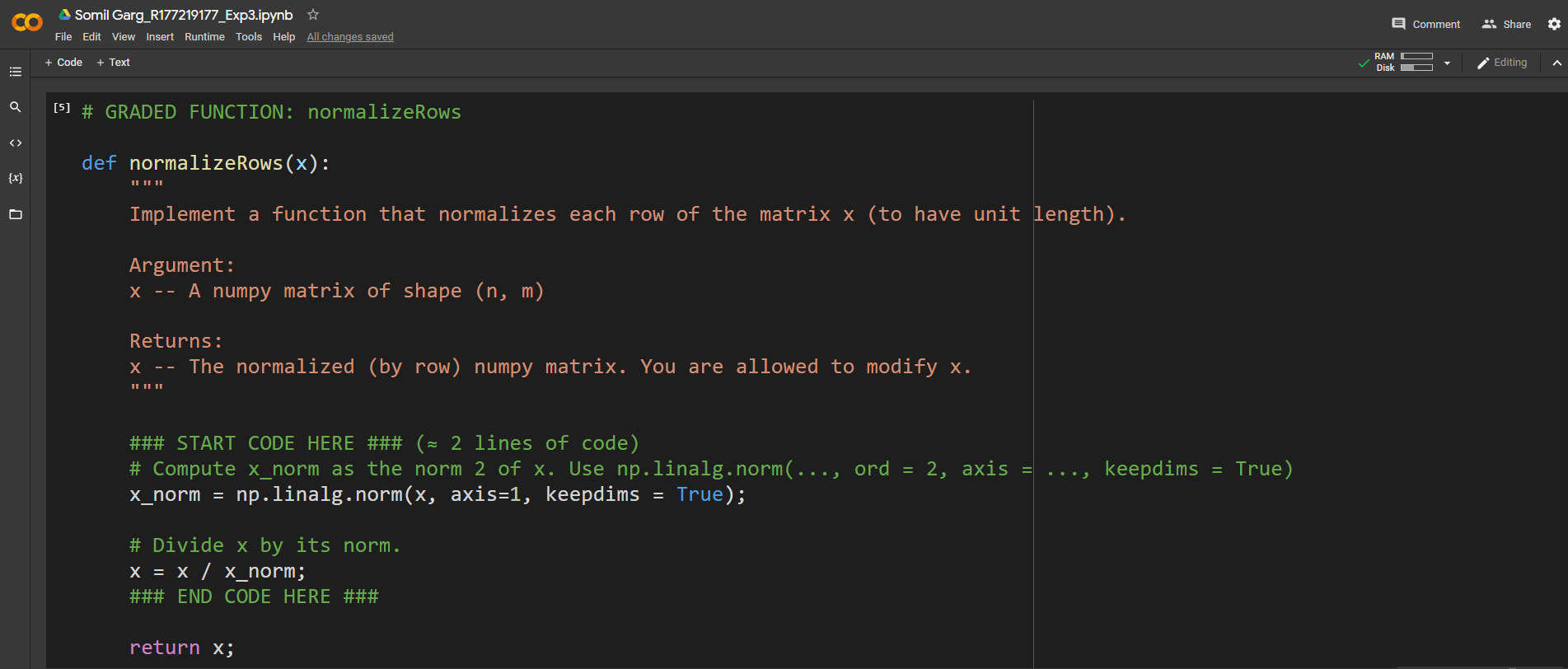
**The formula is: sigmoid\_derivative(x) = σ ′ (x) = σ(x)(1 − σ(x)) (1.1.2)**

**Code this function in two steps:**

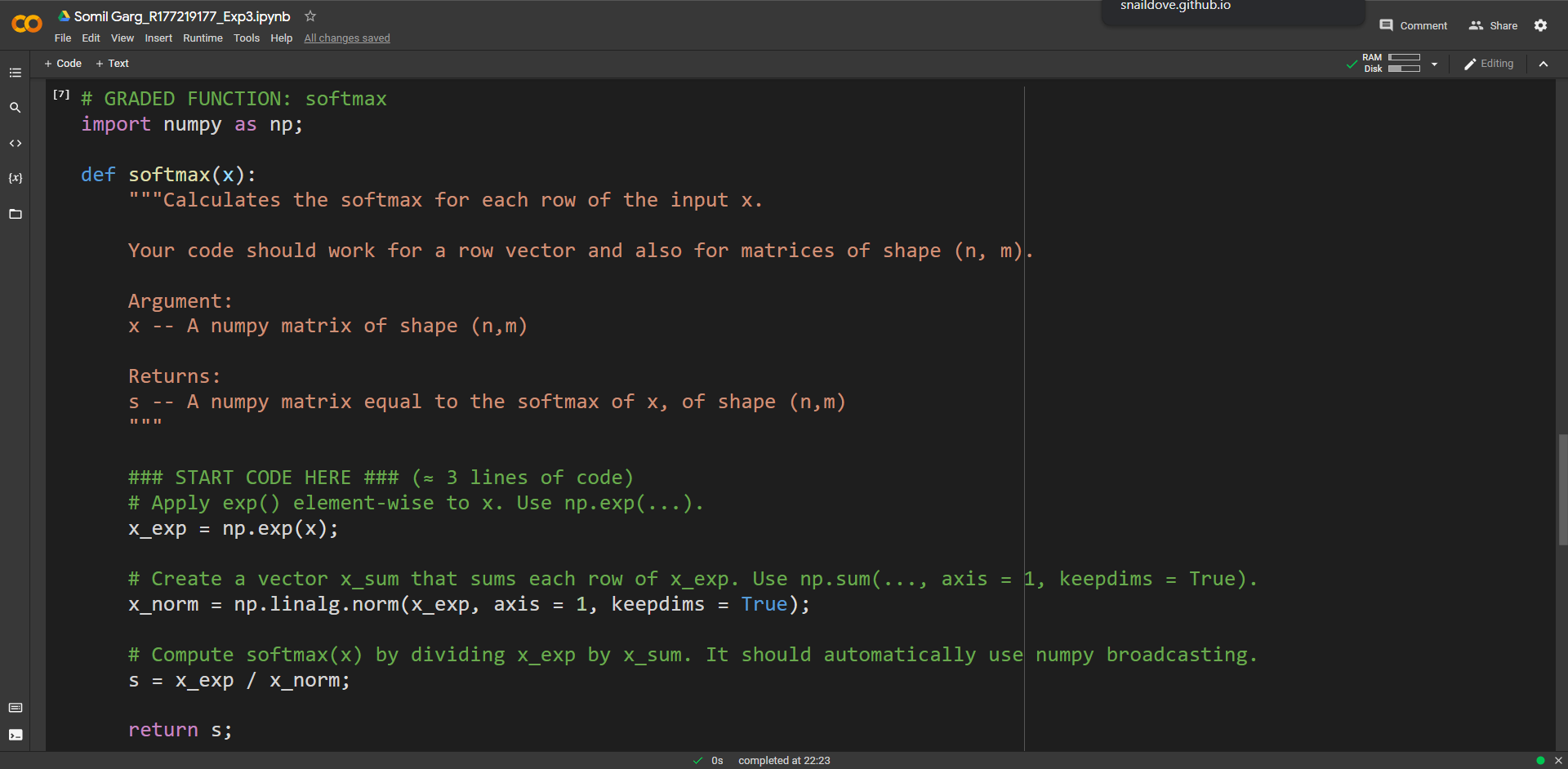
1. **Set s to be the sigmoid of x. You might find your sigmoid(x) function useful.**
2. **Compute σ ′ (x) = s(1 − s)**

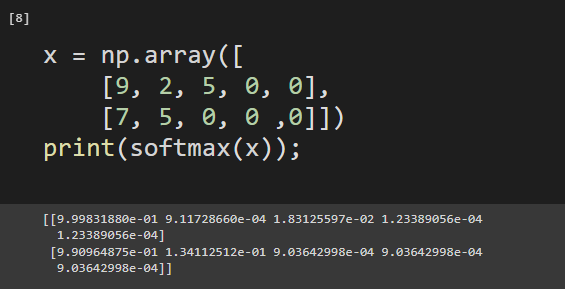


1. **Implement normalizeRows() to normalize the rows of a matrix. After applying this function to an input matrix x, each row of x should be a vector of unit length (meaning length 1).**



1. **Implement a softmax function using numpy. You can think of softmax as a normalizing function used when your algorithm needs to classify two or more classes.**

****

****

1. **Implement the numpy vectorized version of the L1 loss. You may find the function abs(x) (absolute value of x) useful.**

**Reminder:**

* **The loss is used to evaluate the performance of your model. The bigger your loss is, the more different your predictions (yˆ) are from the true values (y).**

**In deep learning, you use optimization algorithms like Gradient Descent to train your model and to minimize the cost.**

* **L1 loss is defined as: L1(ˆy, y) = ∑m i=0 |y (i) − yˆ (i) |**

