

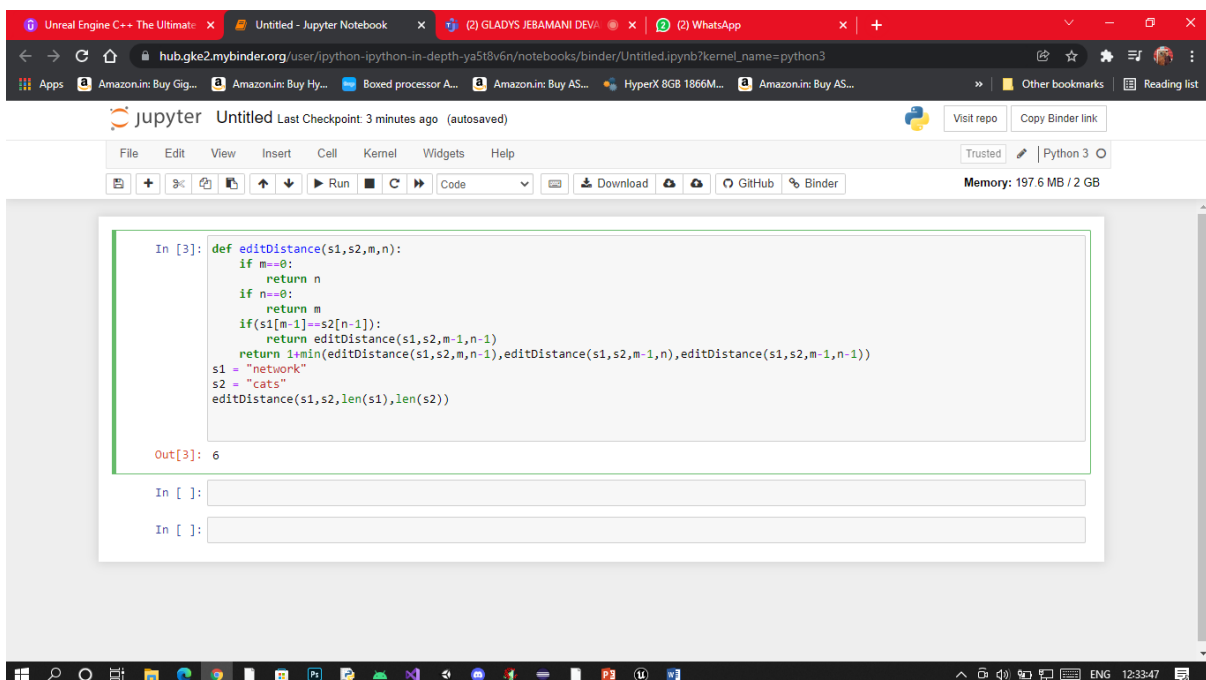
IR PRACTICAL 03

Aim: Implement Dynamic programming algorithm for computing the edit distance between strings s1 and s2. (Hint. Levenshtein Distance).

Code:

```
def editDistance(s1,s2,m,n):  
    if m==0:  
        return n  
    if n==0:  
        return m  
    if(s1[m-1]==s2[n-1]):  
        return editDistance(s1,s2,m-1,n-1)  
    return 1+min(editDistance(s1,s2,m,n-1),editDistance(s1,s2,m-1,n),editDistance(s1,s2,m-1,n-1))  
s1 = "network"  
s2 = "cats"  
editDistance(s1,s2,len(s1),len(s2))
```

Output:



The screenshot shows a Jupyter Notebook window with the following content:

```
In [3]: def editDistance(s1,s2,m,n):  
        if m==0:  
            return n  
        if n==0:  
            return m  
        if(s1[m-1]==s2[n-1]):  
            return editDistance(s1,s2,m-1,n-1)  
        return 1+min(editDistance(s1,s2,m,n-1),editDistance(s1,s2,m-1,n),editDistance(s1,s2,m-1,n-1))  
s1 = "network"  
s2 = "cats"  
editDistance(s1,s2,len(s1),len(s2))  
  
Out[3]: 6
```

The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar at the bottom showing the memory usage (197.6 MB / 2 GB) and the current kernel (Python 3).