

NEURAL NETWORK AND DEEP LEARNING ASSIGNMENT-2

GITHUB LINK: - <https://github.com/revathiatchi/NeuralAssignment2.git>

RECORDING LINK:

<https://github.com/revathiatchi/NeuralAssignment2/assets/156601745/0ca76e7f-042b-4efa-9e47-cd906a545b44>

1. Write a program that takes two strings from the user: first_name, last_name. Pass these variables to fullname function that should return the (full name).
For example: First_name = "your first name", last_name = "your last name"
Full_name = "your full name"

```
def fullname(first_name, last_name):  
    return first_name + " " + last_name  
    first_name = "Revathi"  
    last_name = "Atchi"  
print("Full Name:", fullname(first_name, last_name))
```

```
● ~ def fullname(first_name, last_name):  
    |     return first_name + " " + last_name  
    first_name = "Revathi"  
    last_name = "Atchi"  
    print("Full Name:", fullname(first_name, last_name))
```

Output: -

```
Full Name: Revathi Atchi
```

- A) Write function named "string_alternative" that returns every other char in the full_name string. Str = "Good evening"

Output: Go vnn

```
def str_alternative(Str):  
    return Str[::2]  
print(str_alternative("Good evening"))
```

```
def str_alternative(Str):  
    ... return Str[::2]  
print(str_alternative("Good evening"))
```

Output: -

2) Write a python program to find the wordcount in a file (input.txt) for each line and then print the output. Finally store the output in **output.txt** file.

Example:

Input: a file includes two lines:

Python Course

Deep Learning Course

Output :

Python Course

Deep Learning Course Word_Count:

Python: 1

Course: 2

Deep: 1

Learning: 1

```
text = open("input.txt", "r")
```

```
d = dict()
```

```
for line in text:
```

```
    line = line.strip()
```

```
    line = line.lower()
```

```
    words = line.split(" ")
```

```
    for word in words:
```

```
        if word in d:
```

```
            d[word] = d[word] + 1
```

```
        else:
```

```
            d[word] = 1
```

```
file1 = open('output.txt', 'w')
```

```
s=""
```

```
for key in list(d.keys()):
```

```
    s += key + ":" + str(d[key]) + "\n"
```

```
file1.write(s)
```

```
file1.close()
```

```

● text = open("input.txt", "r")
  d = dict()
  for line in text:
      line = line.strip()
      line = line.lower()
      words = line.split(" ")
      for word in words:
          if word in d:
              d[word] = d[word] + 1
          else:
              d[word] = 1
  file1 = open('output.txt', 'w')
  s=""
  for key in list(d.keys()):
      s += key+ ":" + str(d[key])+ "\n"
  file1.write(s)
  file1.close()

```

Output: -

input.txt	NNAssignment2.ipynb	output.txt
C: > Users > REVATHI > Desktop > MASTERS > Sprin g2024 > Revathi > input.txt	C: > Users > REVATHI > Desktop > MASTERS > Sprin g2024 > Revathi > output.txt	
1 Python Course	1 python:1	
2 Deep Learning Course	2 course:1	
	3 deep:1	
	4 learning:1	
	5 course1:1	
	6	

3) Write a program, which reads heights (inches.) of customers into a list and convert these heights to centimeters in a separate list using:

1) Nested Interactive loop.

```

heights = [150, 155, 145, 148]
centimeters = []
for height in heights:
    cm = height * 2.54
    centimeters.append(cm)
print("Heights in inches:", heights)
print("Heights in centimeters:", centimeters)

```

```

[12] ✓ 0.0s
heights = [150, 155, 145, 148]
centimeters = []
for height in heights:
    cm = height * 2.54
    centimeters.append(cm)
print("Heights in inches:", heights)
print("Heights in centimeters:", centimeters)

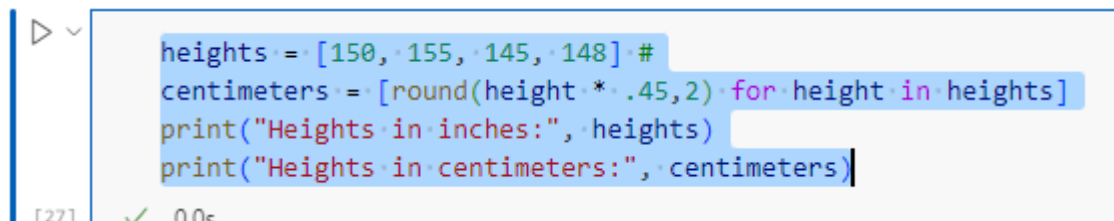
```

Output: -

```
... Heights in inches: [150, 155, 145, 148]
Heights in centimeters: [381.0, 393.7, 368.3, 375.92]
```

2) List comprehensions

```
heights = [150, 155, 145, 148] #
centimeters = [round(height * .45, 2) for height in heights]
print("Heights in inches:", heights)
print("Heights in centimeters:", centimeters)
```



```
heights = [150, 155, 145, 148] #
centimeters = [round(height * .45, 2) for height in heights]
print("Heights in inches:", heights)
print("Heights in centimeters:", centimeters)
```

Output: -

```
... Heights in inches: [150, 155, 145, 148]
Heights in centimeters: [67.5, 69.75, 65.25, 66.6]
```