

Spring 2024 CS5720

Neural Networks & Deep Learning - ICP-3

Name: Sravya Reddy Pilli

Student Id: 700747154

Github link : <https://github.com/09sravyareddy/NNDL-ICP3>

Recording Link:

https://drive.google.com/file/d/1gNppHpu9QoXxUN3BtKQBlsY46_aP06hs/view?usp=drive_link

Code & Output:

1) Create a class Employee and then do the following • Create a data member to count the number of Employees • Create a constructor to initialize name, family, salary, department • Create a function to average salary • Create a Fulltime Employee class and it should inherit the properties of Employee class • Create the instances of Fulltime Employee class and Employee class and call their member functions.

+ Code + Text

```

class Emp:

    count = 0 # contains count of employees
    emps = [] # list of employees

    def __init__(self, name, family, salary, dept):
        self.name = name
        self.family = family
        self.salary = salary
        self.dept = dept

        Emp.count += 1 # count is incremented each time the employee instance is created
        Emp.emps.append(self) # here we are adding the employee details to the list

    # method to calculate the average salary of the employees
    def average_salary(self):
        return sum(emp.salary for emp in Emp.emps) / Emp.count

# Fulltime Employee class inheriting the Employee class
class Fulltime_Emp(Emp):
    pass

# creating instances for above classes
emp1 = Emp("harry potter", "potter", 80000, "II")
fulltime_emp1 = Fulltime_Emp("albus james", "james", 70000, "Developer")
emp2 = Emp("ron weasley", "weasley", 70000, "Marketing")

# Accessing the classes using instances

```

```

# creating instances for above classes
emp1 = Emp("harry potter", "potter", 80000, "II")
fulltime_emp1 = Fulltime_Emp("albus james", "james", 70000, "Developer")
emp2 = Emp("ron weasley", "weasley", 70000, "Marketing")

# Accessing the classes using instances
print(emp1.average_salary())
print(emp2.average_salary())
print(fulltime_emp1.average_salary())

```

```

73333.33333333333
73333.33333333333
73333.33333333333

```

2) Numpy Using NumPy create random vector of size 20 having only float in the range 1-20. Then reshape the array to 4 by 5 Then replace the max in each row by 0 (axis=1) (you can NOT implement it via for loop)

```

▶ import numpy as np # importing the numpy module

saml = np.random.uniform(low=1, high=20, size=20) # generating random float values between 1 and 20
reshape_arr = saml.reshape((4, 5)) # reshaping the vector to a 4x5 dimension

print(reshape_arr)

# replacing the max values with 0 in each row
reshape_arr[np.arange(reshape_arr.shape[0]), np.argmax(reshape_arr, axis=1)] = 0

print(reshape_arr)

```

```

⇒ [[ 5.66145619  6.28835676 13.52091631  1.13209547 12.53619985]
   [18.95256754 16.69490354  9.60670469  9.06305109 16.56817353]
   [14.65061786  9.6814238   4.13105532 19.71644748 10.59019248]
   [ 6.31073127  1.10178373 13.81122185 15.41171665 17.95861732]]
[[ 5.66145619  6.28835676  0.          1.13209547 12.53619985]
 [ 0.          16.69490354  9.60670469  9.06305109 16.56817353]
 [14.65061786  9.6814238   4.13105532  0.          10.59019248]
 [ 6.31073127  1.10178373 13.81122185 15.41171665  0.          ]]

```

✓
9s



```
from google.colab import files
```

```
uploaded = files.upload()
```

```
for filename in uploaded.keys():  
    print(f'File "{filename}" uploaded')
```

```
with open('input.txt', 'r') as f:  
    lines = f.readlines()
```

```
# Calculate word counts for the entire file
```

```
word_counts = {}
```

```
for line in lines:  
    words = line.split()  
    for word in words:  
        word_counts[word] = word_counts.get(word, 0) + 1
```

```
# Print word count to the console
```

```
print('Word_Count:')
```

```
for word, count in word_counts.items():  
    print(f'{word}: {count}')
```

```
with open('output.txt', 'w') as f:  
    for line in lines:  
        f.write(line.strip() + '\n')
```

```
f.write('\n\nWord_Count:\n')
```

```
for word, count in word_counts.items():  
    f.write(f'{word}: {count}\n')
```

✓ 22s completed at 11:18 PM



Q Search



files

↑

📁

📄

🔍

..

sample_data

input (1).txt

input (2).txt

input.txt

output.txt

+ Code + Text

✓ 9s

↑

↓

```
f.write(f'{word}: {count}\n')

output_text = ""

for filename in uploaded.keys():
    output_text += f'File "{filename}" uploaded\n'

# Print the output in the console
print(output_text)

# Append the output to the 'output.txt' file
with open('output.txt', 'a') as f:
    f.write(output_text)
```

📁 Choose Files input.txt

- input.txt(text/plain) - 35 bytes, last modified: 1/17/2024 - 100% done

Saving input.txt to input (2).txt

File "input (2).txt" uploaded

Word_Count:

Python: 1

Course: 2

Deep: 1

Learning: 1

disk 81.45 GB available

output.txt File:

output.txt X input.txt

```
1 Python Course
2 Deep Learning Course
3
4
5 Word_Count:
6 Python: 1
7 Course: 2
8 Deep: 1
9 Learning: 1
10 File "input (2).txt" uploaded
11
```

3)

+ Code + Text

✓
10s

```
inches = []
centimeters = []

while True:
    height_inches = input("Enter height in inches (or 'q' to quit): ")
    if height_inches.lower() == 'q':
        break
    height_inches = float(height_inches)
    inches.append(height_inches)
    centimeters.append(height_inches * 2.54)

print("Heights in inches:", inches)
print("Heights in centimeters:", centimeters)
```

```
→ Enter height in inches (or 'q' to quit): 7
Enter height in inches (or 'q' to quit): 9
Enter height in inches (or 'q' to quit): 155
Enter height in inches (or 'q' to quit): 987
Enter height in inches (or 'q' to quit): q
Heights in inches: [7.0, 9.0, 155.0, 987.0]
Heights in centimeters: [17.78, 22.86, 393.7, 2506.98]
```

[25] 987

```
heights_inches = []
while True:
    height_inches = input("Enter height in inches (or 'q' to quit): ")
    if height_inches.lower() == 'q':
        break
    heights_inches.append(float(height_inches))

heights_cm = [height * 2.54 for height in heights_inches]

print("Heights in inches:", heights_inches)
print("Heights in centimeters:", heights_cm)
```

```
Enter height in inches (or 'q' to quit): 7
Enter height in inches (or 'q' to quit): 9
Enter height in inches (or 'q' to quit): 155
Enter height in inches (or 'q' to quit): 987
Enter height in inches (or 'q' to quit): q
Heights in inches: [7.0, 9.0, 155.0, 987.0]
Heights in centimeters: [17.78, 22.86, 393.7, 2506.98]
```

✓ Connected to Python 3 Google Compute Engine