CSL 303 : Artificial Intelligence

TUTORIAL ASSIGNMENT 3

Pre-requisites : Python

Date Assigned: 26h August, 2021 **Date Submitted :** 1st September, 2021

Submitted by:

Name: S. Bhavyesh

Roll Number: 191210045

Branch: Computer Science and Engineering

Semester: 5th

Submitted to: Dr. Chandra Prakash

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

A-7, Institutional Area, near Satyawadi Raja Harish Chandra Hospital, New Delhi, Delhi 110040

Assignment 1 : To complete assignment one, first you have to add your name and roll no in the Google Colab Instructions section below and print it. Perform the 10 task given in the assignment and submit it over Microsoft Team.

Your source file will most likely end in .pynb if you are using a Jupyter notebook; however, it might also end in .py if you are using a Python script.

```
#The following code ensures that Google CoLab is running the correct version of TensorFlow.

try:
  from google.colab import drive
  %tensorflow_version 2.x
  COLAB = True
  print("Hello World")
  print("Note: using Google CoLab")
except:
  print("Hello NITD")
  print("Note: not using Google CoLab")
  COLAB = False
```

```
Hello World
Note: using Google CoLab
```

```
# Print your name and Roll No.
print("S. Bhavyesh 191210045")


# Print the curent time
from datetime import datetime
print("Current Time =", datetime.now().strftime("%H:%M:%S"), "GMT")
```

```
S. Bhavyesh 191210045
Current Time = 08:01:30 GMT
```

Exercise 1. Create following list:

data = [2,34,23,16,56,45,34,26,78,56,1,16]

Write a function to count number of elements in a list . Now delete the duplicate values and print the list in ascending order.

```
# Exercise 1

def length(data):
    print("the number of elements in the list:",len(data))

data=[2,34,23,16,56,45,34,26,78,56,1,16]
length(data)
data_unique=list(set(data))
print("list with unique elements:",data_unique)
data_unique.sort()
print("list with sorted elements:",data_unique)
```

```
the number of elements in the list: 12
list with unique elements: [1, 2, 34, 45, 78, 16, 23, 56, 26]
list with sorted elements: [1, 2, 16, 23, 26, 34, 45, 56, 78]
```

OBSERVATION/COMMENTS

1. *set()* is used to convert the list to a set, meaning that it contains only unique elements.

2. Duplicate removal + sorting can be done by a single NumPy function: *np.unique()*

Exercise 2: Take five numbers as input from the user and save into a list. Find the maximum of the list and sort the data in descending order.

```
# Exercise 2

arr=[]
print("Enter 5 values:")
for i in range(5):
    temp=int(input())
    arr.append(temp)

print("max value in list:",max(arr))
arr.sort(reverse=True)
print("reverse-sorted array:",arr)
```

```
Enter 5 values:
1
7
2
9
4
max value in list: 9
reverse-sorted array: [9, 7, 4, 2, 1]
```

OBSERVATION/COMMENTS

Input in Python is by default *string* type. It has to be type-casted to *int* for our purpose.

Exercise 4:
(i) Generate two arrays A1 and A2 of size 5 X 4 and 3 X 4 respectively using np.random()
(ii) Join them and make an array A3 of 8 X 4. Now append random numbers ranging between from 0 to 5 to make the fourth array A4 of size 10 X 10.
(iii) Print all the arrays and their transpose (Transpose of ‘A’ can be obtained by ‘A.T’)

```
# Exercise 4

import numpy as np

a1=np.random.rand(5,4)
a2=np.random.rand(3,4)

a3=np.concatenate((a1,a2))

a4=np.random.uniform(0,5,10*10).reshape(10,10)
a4[:a3.shape[0],:a3.shape[1]]=a3

np.set_printoptions( linewidth=120)

print("matrices and their transposes: \n")
print("A1\n",a1,"\n\n","A1'\n",a1.T,"\n")
print("A2\n",a2,"\n\n","A2'\n",a2.T,"\n")
print("A3\n",a3,"\n\n","A3'\n",a3.T,"\n")
print("A4\n",a4,"\n\n","A4'\n",a4.T,"\n")
```

```
matrices and their transposes:

A1
 [[0.91457103 0.1254533  0.29975789 0.35603328]
 [0.54109095 0.39188982 0.72766911 0.16164315]
 [0.73675584 0.63719715 0.52022048 0.32763824]
 [0.48601461 0.48461641 0.91627873 0.78445968]
 [0.2834102  0.74031714 0.8204245  0.18025606]]

 A1'
 [[0.91457103 0.54109095 0.73675584 0.48601461 0.2834102 ]
 [0.1254533  0.39188982 0.63719715 0.48461641 0.74031714]
 [0.29975789 0.72766911 0.52022048 0.91627873 0.8204245 ]
 [0.35603328 0.16164315 0.32763824 0.78445968 0.18025606]]

A2
 [[0.33556109 0.30113735 0.64685735 0.97498506]
 [0.59465062 0.32809221 0.01743884 0.17112279]
 [0.87674175 0.21320626 0.45450218 0.58870321]]

 A2'
 [[0.33556109 0.59465062 0.87674175]
 [0.30113735 0.32809221 0.21320626]
 [0.64685735 0.01743884 0.45450218]
 [0.97498506 0.17112279 0.58870321]]
```

```
A3
 [[0.91457103 0.1254533  0.29975789 0.35603328]
 [0.54109095 0.39188982 0.72766911 0.16164315]
 [0.73675584 0.63719715 0.52022048 0.32763824]
 [0.48601461 0.48461641 0.91627873 0.78445968]
 [0.2834102  0.74031714 0.8204245  0.18025606]
 [0.33556109 0.30113735 0.64685735 0.97498506]
 [0.59465062 0.32809221 0.01743884 0.17112279]
 [0.87674175 0.21320626 0.45450218 0.58870321]]

 A3'
 [[0.91457103 0.54109095 0.73675584 0.48601461 0.2834102  0.33556109 0.59465062 0.87674175]
 [0.1254533  0.39188982 0.63719715 0.48461641 0.74031714 0.30113735 0.32809221 0.21320626]
 [0.29975789 0.72766911 0.52022048 0.91627873 0.8204245  0.64685735 0.01743884 0.45450218]
 [0.35603328 0.16164315 0.32763824 0.78445968 0.18025606 0.97498506 0.17112279 0.58870321]]
```

```
A4
 [[0.91457103 0.1254533  0.29975789 0.35603328 3.32315498 1.3457379  1.44186228 2.79700624 0.33916205 3.35218058]
 [0.54109095 0.39188982 0.72766911 0.16164315 2.67130134 3.13710136 1.1309794  1.36953949 3.95814044 2.38722103]
 [0.73675584 0.63719715 0.52022048 0.32763824 0.51787417 1.24451744 1.57086943 3.5489721  3.70264607 3.47850136]
 [0.48601461 0.48461641 0.91627873 0.78445968 0.78176236 2.01929137 4.55515553 0.82962036 2.55781608 4.67955588]
 [0.2834102  0.74031714 0.8204245  0.18025606 1.79087494 0.51938796 0.18661013 2.61869807 0.6022112  0.30510927]
 [0.33556109 0.30113735 0.64685735 0.97498506 1.39987158 3.41869668 4.23806024 3.55455621 0.43422482 2.79793747]
 [0.59465062 0.32809221 0.01743884 0.17112279 4.14875883 3.78829704 2.01625429 3.69624949 1.99462668 0.82704468]
 [0.87674175 0.21320626 0.45450218 0.58870321 0.44334635 4.53448108 3.31892575 4.68258814 2.49783053 4.05834927]
 [4.31845369 1.98373138 0.05163568 2.00419727 1.0032731  1.04821816 1.90811763 0.62674203 0.43946283 4.91086784]
 [4.85678247 4.47792534 1.84956317 2.78820271 0.83487548 4.58645094 3.22457277 1.86673737 4.6149407  1.83873381]]

 A4'
 [[0.91457103 0.54109095 0.73675584 0.48601461 0.2834102  0.33556109 0.59465062 0.87674175 4.31845369 4.85678247]
 [0.1254533  0.39188982 0.63719715 0.48461641 0.74031714 0.30113735 0.32809221 0.21320626 1.98373138 4.47792534]
 [0.29975789 0.72766911 0.52022048 0.91627873 0.8204245  0.64685735 0.01743884 0.45450218 0.05163568 1.84956317]
 [0.35603328 0.16164315 0.32763824 0.78445968 0.18025606 0.97498506 0.17112279 0.58870321 2.00419727 2.78820271]
 [3.32315498 2.67130134 0.51787417 0.78176236 1.79087494 1.39987158 4.14875883 0.44334635 1.0032731  0.83487548]
 [1.3457379  3.13710136 1.24451744 2.01929137 0.51938796 3.41869668 3.78829704 4.53448108 1.04821816 4.58645094]
 [1.44186228 1.1309794  1.57086943 4.55515553 0.18661013 4.23806024 2.01625429 3.31892575 1.90811763 3.22457277]
 [2.79700624 1.36953949 3.5489721  0.82962036 2.61869807 3.55455621 3.69624949 4.68258814 0.62674203 1.86673737]
 [0.33916205 3.95814044 3.70264607 2.55781608 0.6022112  0.43422482 1.99462668 2.49783053 0.43946283 4.6149407 ]
 [3.35218058 2.38722103 3.47850136 4.67955588 0.30510927 2.79793747 0.82704468 4.05834927 4.91086784 1.83873381]]
```

OBSERVATION/COMMENTS

1. *np.concatenate* has another postional argument, *axis* which is by default, 0. *axis* of 0 means stack arrays on top of another, while 1 signifies side-by-side.

2. *np.random.uniform* returns flat arrays, which have to be reshaped by the *reshape()* function.

3. *np.setprintoptions* disables the wrapping of output, so that elements of a row appear in one line, instead of:

```
A4
 [[4.67447155e-01 8.16815468e-01 1.70816983e-01 4.15098040e-01 8.03185046e-01 4.70332764e+00 3.35655865e+00
  1.76036353e+00 4.37606790e+00 5.13874899e-01]
 [2.65511465e-01 9.88953141e-01 4.27303770e-01 8.39486072e-01 4.58505965e+00 1.28710997e+00 7.76506068e-01
  2.88499906e+00 3.32401103e+00 3.12368753e+00]
 [2.14148589e-01 9.64947521e-01 9.27418252e-01 2.46030769e-01 2.72736312e+00 4.22489300e+00 4.97149933e+00
  4.29015759e+00 2.31162121e+00 3.87159901e+00]
 [1.84244479e-01 6.44291800e-01 5.93209335e-01 3.56868211e-01 4.85770284e+00 4.12845565e+00 1.91279495e+00
  2.89175205e+00 2.77353902e+00 3.54071484e+00]
 [9.12285362e-01 6.37859281e-01 2.21520836e-01 6.66011183e-01 2.40481554e+00 2.55047645e+00 1.10776745e+00
  1.22107205e+00 1.21778998e+00 3.10639509e-01]
 [6.48946751e-01 1.96853369e-03 1.64556372e-01 9.84090652e-03 7.17493275e-01 1.75852293e+00 1.18733099e+00
  5.89873380e-01 2.04437773e+00 2.15971134e+00]
 [2.33691446e-03 5.70100430e-01 6.10940302e-01 5.27252906e-01 3.64115279e+00 2.55170473e+00 1.96506322e+00
  3.43514730e+00 3.43505101e+00 4.66902469e+00]
 [2.44868872e-01 6.10213674e-01 7.41955684e-01 5.41212783e-01 4.59957139e-01 1.06829860e+00 1.89954798e+00
  4.10461682e+00 4.45118780e+00 2.82801750e+00]
 [2.78455534e+00 2.80193893e+00 1.41216745e+00 1.53768482e+00 4.50914424e-01 3.46531180e+00 4.40578073e+00
  1.58963934e+00 3.11178276e+00 2.84012200e+00]
 [1.59675771e+00 5.86479212e-01 3.85554421e-01 5.95680182e-01 2.40468447e+00 5.41061891e-01 8.08939863e-01
  3.73851963e+00 3.40572321e+00 3.09105318e+00]]
```

Exercise 5: Create two dictionaries.
The first dictionary ‘name’ will contain first name(key) of a person and its hash value(value). The Second will contain hash value(key) and mobile no(value).

i) Add 5 entries.
ii) Delete two entries by taking the input from user as the first name.
iii) Add two entries by taking the input as the first name and mobile no.

Hint: You can use remainder (%) to obtain hash value.

```python
# Exercise 5

name_dict={}
mob_dict={}

for i in range(5):
    name, mob= input("Enter name and mobile number:").split()
    name_dict[name]=int(mob)%31
    mob_dict[int(mob)%31]=mob

print("\nName Dictionary",name_dict)
print("Mob. Dictionary",mob_dict)

name = input("\n Enter name whose data is to be deleted:")
mob_dict.pop(name_dict[name])
name_dict.pop(name)

print(name_dict)
print(mob_dict)

name, mob= input("\n Enter name and mobile number:").split()
name_dict[name]=int(mob)%31
mob_dict[int(mob)%31]=mob

print(name_dict)
print(mob_dict)
```

```
Enter name and mobile number:Alpha 9876543210
Enter name and mobile number:Beta 9898989898
Enter name and mobile number:Gamma 9871234560
Enter name and mobile number:Delta 9182736450
Enter name and mobile number:Lambda 9999987654

Name Dictionary {'Alpha': 2, 'Beta': 24, 'Gamma': 9, 'Delta': 26, 'Lambda': 28}
Mob. Dictionary {2: '9876543210', 24: '9898989898', 9: '9871234560', 26: '9182736450', 28: '9999987654'}

 Enter name whose data is to be deleted:Gamma
{'Alpha': 2, 'Beta': 24, 'Delta': 26, 'Lambda': 28}
{2: '9876543210', 24: '9898989898', 26: '9182736450', 28: '9999987654'}

 Enter name and mobile number:Epsilon 9876512340
{'Alpha': 2, 'Beta': 24, 'Delta': 26, 'Lambda': 28, 'Epsilon': 8}
{2: '9876543210', 24: '9898989898', 26: '9182736450', 28: '9999987654', 8: '9876512340'}
```

OBSERVATION/COMMENTS

1. The hash function used here is *mob. number % 31.* A prime number was chosen for efficient hashing.

2. Removal of entries can also be done by *del*, but it does not do error handling properly, and *pop()* has the additional advantage of returning the popped value.

Exercise 6: Write a NumPy program to create an element-wise comparison (greater, greater_equal, less and less_equal) of two given arrays

```
# Exercise 6

import numpy as np

a=np.array([1,4,5,2,1])
b=np.array([7,3,5,6,7])

print("greater than:",np.greater(a,b))
print("greater than or equal to:",np.greater_equal(a,b))
print("less than:",np.less(a,b))
print("less than or equal to",np.less_equal(a,b))
```

```
greater than: [False  True False False False]
greater than or equal to: [False  True  True False False]
less than: [ True False False  True  True]
less than or equal to [ True False  True  True  True]
```

OBSERVATION/COMMENTS

Instead of the NumPy functions, we can use shorthand operators >, >=, <, <= respectively. For example, a>b is equivalent to *np.greater(a,b).* Both give same results.

Exercise 7: Write a NumPy program to create an array with the values 100, 71, 113, 1050 and determine the size of the memory occupied by the array.

```
# Exercise 7

import numpy as np

a=np.array([100,71,113,1050])
print(a.nbytes)
print(a.__sizeof__())
```

```
32
128
```

OBSERVATION/COMMENTS

a.nbytes gives the total memory space occupied by the *elements(64-bit floats = 8-byte floats, 4 of them means 32 bytes). a.__sizeof__()* also includes the overhead data of the NumPy array.

Exercise 8: Write a NumPy program to get the powers (x ^ 3) of an array values element-wise. Expected Output: Original array [1 2 3 4 5] Output array: [1 8 27 64 125]

```
# Exercise 8

import numpy as np

a=np.array([1,2,3,4,5])
print(a,a**3)
```

```
[1 2 3 4 5] [  1   8  27  64 125]
```

OBSERVATION/COMMENTS

The shorthand ** can be replaced by *np.power(a,3)*.

Exercise 9: Write a NumPy program to get the floor, ceiling and truncated values of the elements of a numpy array. Sample Output: Original array: [-1.3, -1.15, -0.1, 0.12, 1.7, 0.9, 1.1]

```
# Exercise 9

import numpy as np

a=[-1.3, -1.15, -0.1, 0.12, 1.7, 0.9, 1.1]

print("floor:",np.floor(a))
print("ceil.:",np.ceil(a))
print("trunc:",np.trunc(a))
```

```
floor: [-2. -2. -1.  0.  1.  0.  1.]
ceil.: [-1. -1. -0.  1.  2.  1.  2.]
trunc: [-1. -1. -0.  0.  1.  0.  1.]
```

OBSERVATION/COMMENTS

1. *np.floor()* returns an array of floored values.
2. *np.ceil()* returns an array of ceiling values.
1. *np.trunc()* removes digits beyond the decimal point.

Exercise 10: Write a program in python to display prime numbers from x to y (here x and y are user given values).

```
# Exercise 10

import math

x,y = input("Enter range for prime numbers:").split()
x,y=int(x),int(y)

for temp in range(x,y+1):
    max_val=int(math.sqrt(temp))+1
    for i in range(2,max_val):
        if(temp%i==0):
            break
    else:
        print(temp)
```

```
Enter range for prime numbers:10 50
11
13
17
19
23
29
31
37
41
43
47
```

OBSERVATION/COMMENTS

Notice that the *for-loop* has an *else* condition associated with it. This is a feature rarely found in other languages. This *else* acts as a check for normal execution of the loop. If the loop does not break out(looping through all values of the range, in this context all potential factors of the number), then the control goes to *else* part. Otherwise the *else* bit is also skipped.