



---

# LAB EXAM

---

Data Science



JILSE JACOB

RES:- AJC20MCA2043

RMCA Batch -A

1. Create 2 1D array A,B with 16 elements and transform it into 4x4 2D array .Perform following operations on the matrix

a)Matrix Multiplication

b)Transpose of A

c)From B,Display the last 2 elements of 3<sup>rd</sup> and 4<sup>th</sup> row

**Program:**

```
import numpy as n
```

```
a = n.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
```

```
b = n.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
```

```
a = a.reshape(4,4)
```

```
b = b.reshape(4,4)
```

```
print("matrix A\n",a)
```

```
print("\nmatrix B\n",b)
```

```
f=n.multiply(a,b)
```

```
print("\n\n multiply of A and B :\n",f)
```

```
e=a.T
```

```
print("\n\nTranspose of A :\n",e)
```

```
g=n.multiply(e,b)
```

```
print("\n\nA Transpose multiply B\n",g)
```

```
print("\n\nlast 2 elements of 3rd row",g[2][2],g[2][3])
```

```
print("last 2 elements of 4rd row",g[3][2],g[3][3])
```

**Output :**

matrix A

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

matrix B

```
[[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

multiply of A and B :

```
[[ 1 4 9 16]
 [25 36 49 64]
 [ 81 100 121 144]
 [169 196 225 256]]
```

Transpose of A :

```
[[ 1 5 9 13]
 [ 2 6 10 14]
 [ 3 7 11 15]
 [ 4 8 12 16]]
```

A Transpose multiply B

```
[[ 1 10 27 52]
 [ 10 36 70 112]]
```

[ 27 70 121 180]

[ 52 112 180 256]]

last 2 elements of 3rd row 121 180

last 2 elements of 4rd row 180 256

Process finished with exit code 0

## 2. Program for natural language processing which perform speech tagging?

### Program:

```
import nltk

text = "since long ago all are not happy with there life ." \
      "Till the end of time everything" \
      " will be same there is nothing we could do." \
      "People all over the world are waithing for" \
      " the salvation promised by jesus"

tokeni=nltk.sent_tokenize(text)

new = []

for i in tokeni:

    newtoken = nltk.word_tokenize(i)

    print(newtoken)

    tagg = nltk.pos_tag(newtoken)

    print(tagg)

    new = new.append(tagg)

grammers = "NP : {<DT>?<JJ>*<NN>}"

chunked = nltk.RegexpParser(grammers)

chunk = chunked.parse(tagg)

print(chunk)
```

### Output:

```
['since', 'long', 'ago', 'all', 'are', 'not', 'happy', 'with', 'there', 'life', '.Till', 'the', 'end', 'of', 'time', 'everything', 'will', 'be', 'same', 'there', 'is', 'nothing', 'we', 'could', 'do.People', 'all', 'over', 'the', 'world', 'are', 'waithing', 'for', 'the', 'salvation', 'promised', 'by', 'jesus']
```

```
[('since', 'IN'), ('long', 'RB'), ('ago', 'IN'), ('all', 'DT'), ('are', 'VBP'), ('not', 'RB'), ('happy', 'JJ'), ('with', 'IN'), ('there', 'EX'), ('life', 'NN'), ('.Till', 'VBD'), ('the', 'DT'), ('end', 'NN'), ('of', 'IN'), ('time', 'NN'), ('everything', 'NN'), ('will', 'MD'), ('be', 'VB'), ('same', 'JJ'), ('there', 'EX'), ('is', 'VBZ'), ('nothing', 'NN'), ('we', 'PRP'), ('could', 'MD'), ('do.People', 'VB'), ('all', 'DT'), ('over', 'IN'), ('the', 'DT'), ('world', 'NN'), ('are', 'VBP'), ('waithing', 'VBG'), ('for', 'IN'), ('the', 'DT'), ('salvation', 'NN'), ('promised', 'VBN'), ('by', 'IN'), ('jesus', 'NN')]
```

(S

since/IN

long/RB

ago/IN

all/DT

are/VBP

not/RB

happy/JJ

with/IN

there/EX

(NP life/NN)

.Till/VBD

(NP the/DT end/NN)

of/IN

(NP time/NN)

(NP everything/NN)

will/MD

be/VB

same/JJ

there/EX

is/VBZ

(NP nothing/NN)

we/PRP

could/MD

do.People/VB

all/DT

over/IN

(NP the/DT world/NN)

are/VBP

waithing/VBG

for/IN

(NP the/DT salvation/NN)

promised/VBN

by/IN

(NP jesus/NN))

Process finished with exit code 0