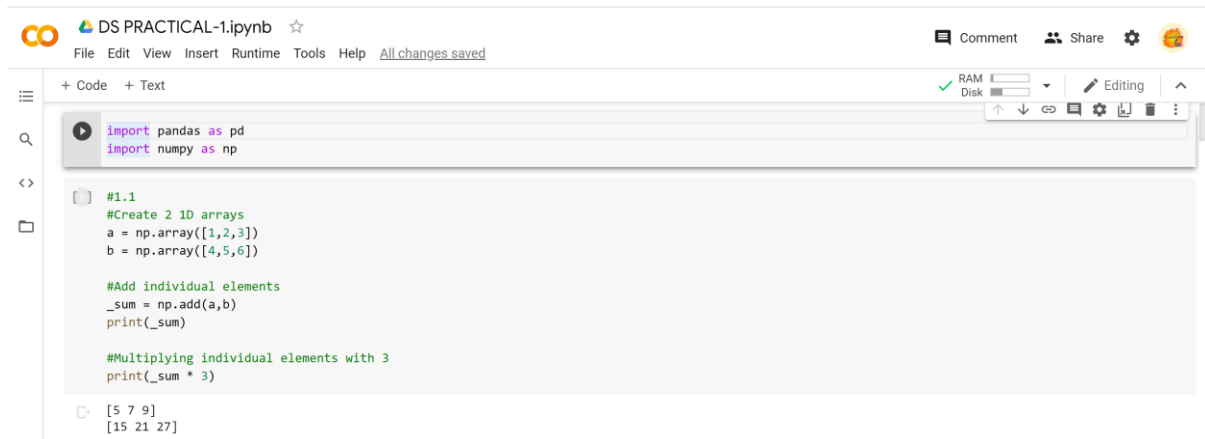


Data preparations in Data Science

I.

- (i) Create 2- One dimensional Arrays and print sum of individual element and multiply each with 3



A Jupyter Notebook interface titled 'DS PRACTICAL-1.ipynb'. The code cell contains the following Python code:

```
import pandas as pd
import numpy as np

#1.1
#Create 2 1D arrays
a = np.array([1,2,3])
b = np.array([4,5,6])

#Add individual elements
_sum = np.add(a,b)
print(_sum)

#Multiplying individual elements with 3
print(_sum * 3)
```

The output of the code is displayed below the cell:

```
[ 5  7  9]
[15 21 27]
```

- (ii) Manipulate Logical and, OR Not operations on Array



A Jupyter Notebook interface titled 'DS PRACTICAL-1.ipynb'. The code cell contains the following Python code:

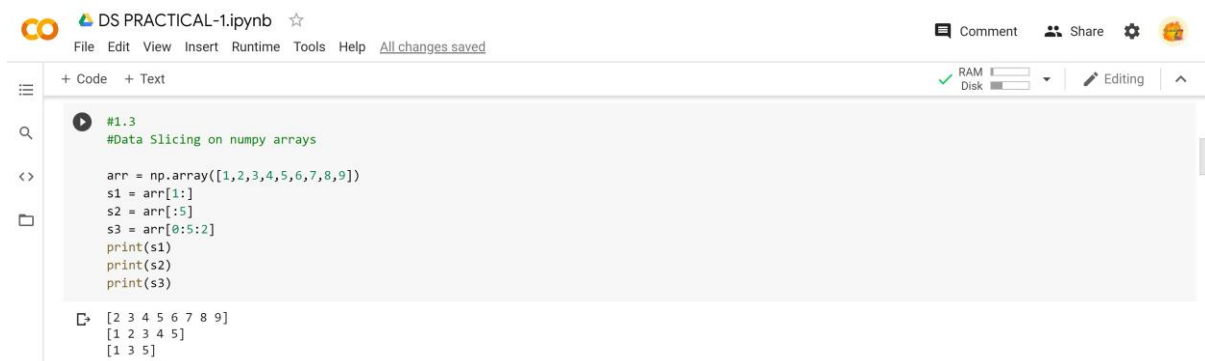
```
#1.2
#Logical operators on Numpy arrays
_or = np.logical_or(a>1, a<4)
_and = np.logical_and(a>1,a<4)
_not = np.logical_not(a<0)

print('Logical And: ', _and," , Logical OR: ", _or," , Logical Not: ",_not)
```

The output of the code is displayed below the cell:

```
Logical And: [False  True  True] , Logical OR: [ True  True  True] , Logical Not: [ True  True  True]
```

- (iii) Perform data Slicing Operations



A Jupyter Notebook interface titled 'DS PRACTICAL-1.ipynb'. The code cell contains the following Python code:

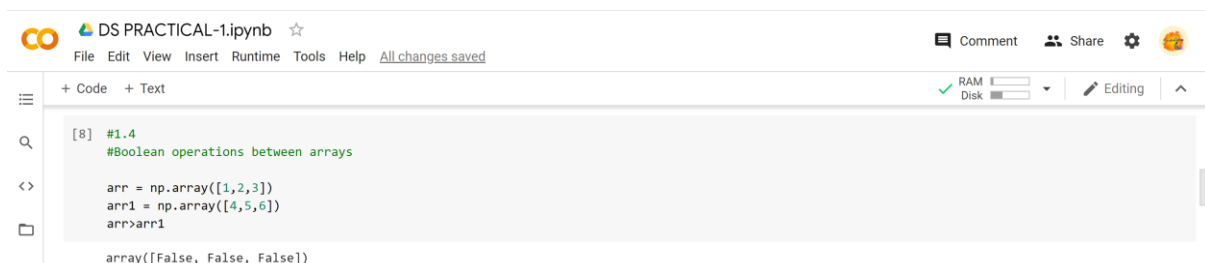
```
#1.3
#Data Slicing on numpy arrays

arr = np.array([1,2,3,4,5,6,7,8,9])
s1 = arr[1:]
s2 = arr[:5]
s3 = arr[0:5:2]
print(s1)
print(s2)
print(s3)
```

The output of the code is displayed below the cell:

```
[ 2  3  4  5  6  7  8  9]
[ 1  2  3  4  5]
[ 1  3  5]
```

- (iv) Perform Boolean operations between arrays



A Jupyter Notebook interface titled 'DS PRACTICAL-1.ipynb'. The code cell contains the following Python code:

```
#1.4
#Boolean operations between arrays

arr = np.array([1,2,3])
arr1 = np.array([4,5,6])
arr>arr1

array([False, False, False])
```

II. Create 2- Two dimensional Arrays and print sum of individual element and multiply each with 3



DS PRACTICAL-1.ipynb

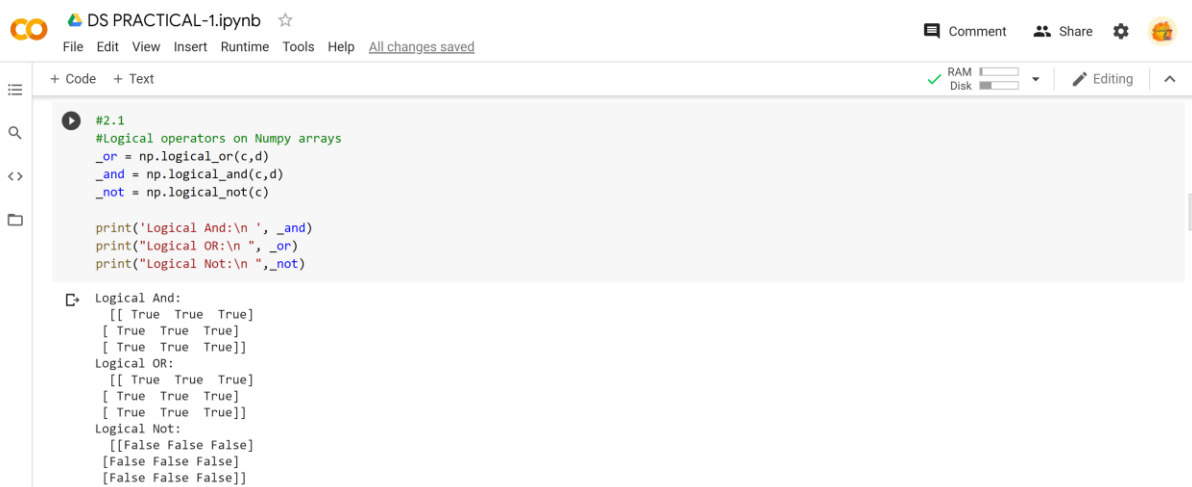
```
#2
#Create 2d arrays
c = np.array([[1,2,3],[4,5,6],[7,8,9]])
d = np.array([[11,12,13],[14,15,16],[17,18,19]])

#Add individual elements
_sum = np.add(a,b)
print(_sum)

#Multiplying individual elements with 3
print(_sum * 3)
```

[5 7 9]
[15 21 27]

(i) Manipulate Logical and, OR Not operations on Array



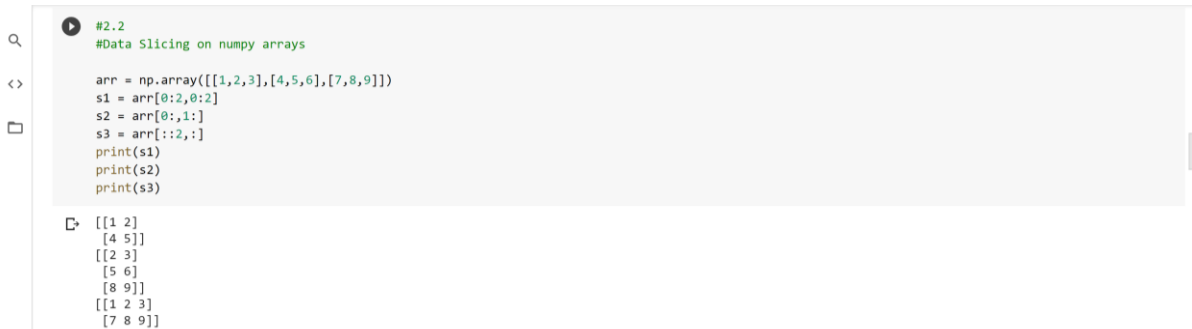
DS PRACTICAL-1.ipynb

```
#2.1
#Logical operators on Numpy arrays
_or = np.logical_or(c,d)
_and = np.logical_and(c,d)
_not = np.logical_not(c)

print('Logical And:\n ', _and)
print("Logical OR:\n ", _or)
print("Logical Not:\n ", _not)
```

Logical And:
[[True True True]
[True True True]
[True True True]]
Logical OR:
[[True True True]
[True True True]
[True True True]]
Logical Not:
[[False False False]
[False False False]
[False False False]]

(ii) Perform data Slicing Operations

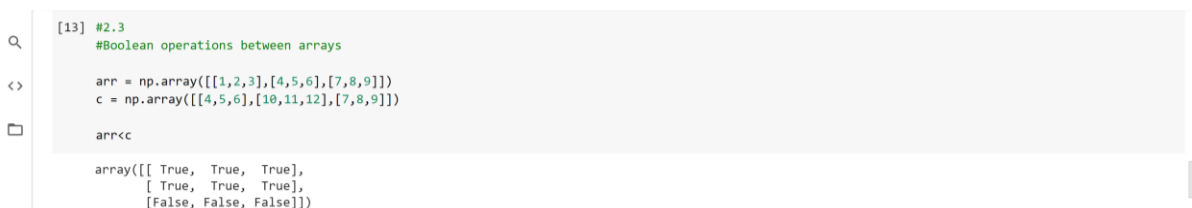


#2.2
#Data Slicing on numpy arrays

```
arr = np.array([[1,2,3],[4,5,6],[7,8,9]])
s1 = arr[0:2,0:2]
s2 = arr[0:,1:]
s3 = arr[:,2,:]
print(s1)
print(s2)
print(s3)
```

[[1 2]
[4 5]]
[[2 3]
[5 6]
[8 9]]
[[1 2 3]
[7 8 9]]

(iii) Perform Boolean operations between arrays



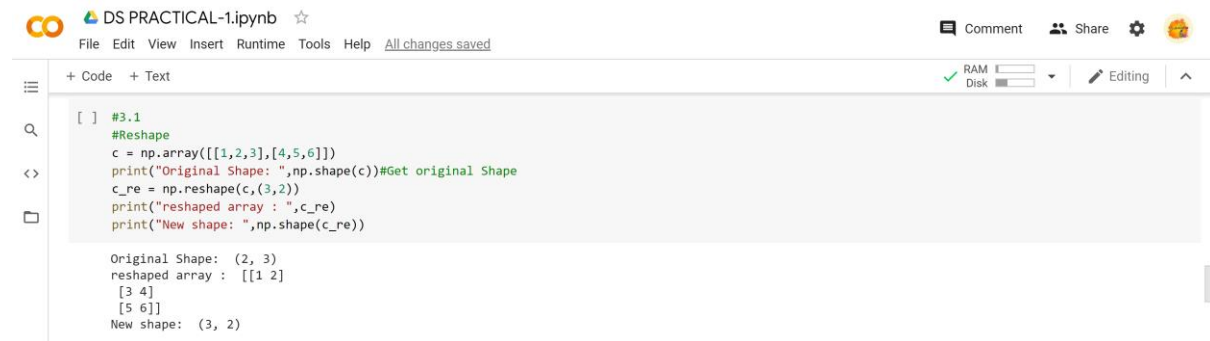
[13] #2.3
#Boolean operations between arrays

```
arr = np.array([[1,2,3],[4,5,6],[7,8,9]])
c = np.array([[4,5,6],[10,11,12],[7,8,9]])
arrcc
```

array([[True, True, True],
[True, True, True],
[False, False, False]])

III. reshape(), arrange(), resize(), hsplit(), Extract ones, Two's in arrays, Scalar()

reshape()



```
#3.1
#Reshape
c = np.array([[1,2,3],[4,5,6]])
print("Original Shape: ",np.shape(c))#Get original Shape
c_re = np.reshape(c,(3,2))
print("reshaped array : ",c_re)
print("New shape: ",np.shape(c_re))
```

Original Shape: (2, 3)
reshaped array : [[1 2]
[3 4]
[5 6]]
New shape: (3, 2)

arrange()

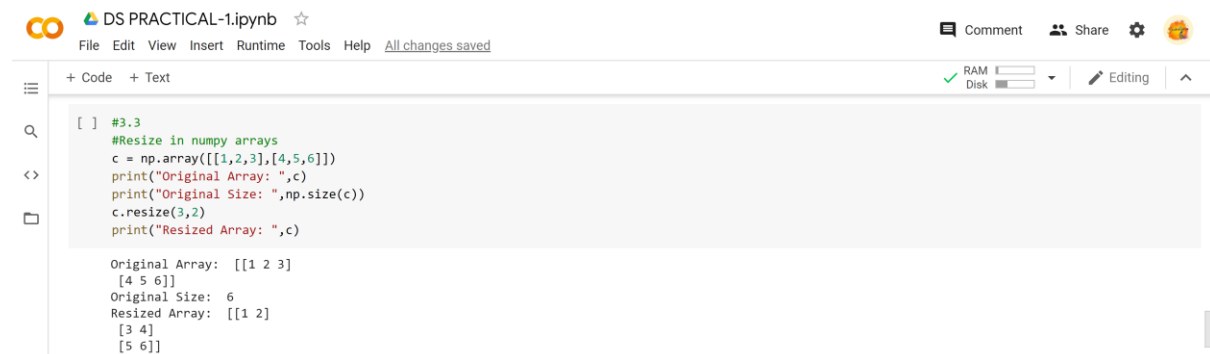


```
#3.2
#Arrange
arr = np.arange(0,9,2)
arr2 = np.arange(-10,-20,-1)
arr3 = np.arange(-10,20,1)
arr4 = np.arange(0,-50,2)

print(arr)
print(arr2)
print(arr3)
print(arr4)
```

[0 2 4 6 8]
[-10 -11 -12 -13 -14 -15 -16 -17 -18 -19]
[-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7]
[8 9 10 11 12 13 14 15 16 17 18 19]
[]

resize()



```
#3.3
#Resize in numpy arrays
c = np.array([[1,2,3],[4,5,6]])
print("Original Array: ",c)
print("Original Size: ",np.size(c))
c.resize(3,2)
print("Resized Array: ",c)
```

Original Array: [[1 2 3]
[4 5 6]]
Original Size: 6
Resized Array: [[1 2]
[3 4]
[5 6]]

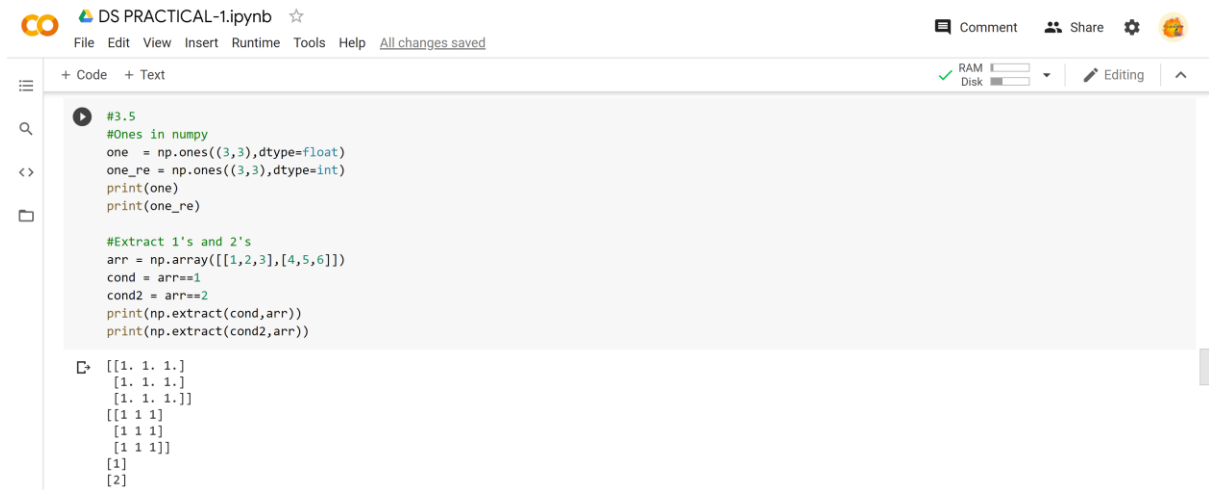
hsplit()



```
#3.4
#hsplit - splitting column wise
arr = np.arange(17).reshape(4,4)
arr_hsplit = np.hsplit(arr,2) #divides array column wise into 2 subarrays
arr_hsplit
```

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

Extract ones, Two's in arrays



The image shows a Jupyter Notebook interface with a code cell containing the following Python code:

```
#3.5
#Ones in numpy
one = np.ones((3,3),dtype=float)
one_re = np.ones((3,3),dtype=int)
print(one)
print(one_re)

#Extract 1's and 2's
arr = np.array([[1,2,3],[4,5,6]])
cond = arr==1
cond2 = arr==2
print(np.extract(cond,arr))
print(np.extract(cond2,arr))
```

The output of the code is displayed below the code cell:

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[[1 1 1]
 [1 1 1]
 [1 1 1]]
[1]
[2]
```

Scalar()



The image shows a Jupyter Notebook interface with a code cell containing the following Python code:

```
[6] #3.6
#Scalar()
np.asscalar(np.array([24]))
```

The output of the code is displayed below the code cell:

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead:
24
```

IV. Create Data Frame with a MultiIndex

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
#4
#Creating dataframe with multi index
arr = [[1,2,3,4,5],['red','green','blue','white','black']]
mi = pd.MultiIndex.from_arrays(arr, names=('number', 'color'))
print("Original Multiindex: ",mi)
print("Data Frame: ")
mi.to_frame(index=False)
```

Output:

```
Original Multiindex: MultiIndex([(1, 'red'),
                                (2, 'green'),
                                (3, 'blue'),
                                (4, 'white'),
                                (5, 'black')],
                                names=['number', 'color'])
```

	number	color
0	1	red
1	2	green
2	3	blue
3	4	white
4	5	black

V. import .CSV file do the following

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
#5
df = pd.read_csv('lab1.csv')
df
```

Output:

	gender	Nationality	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16	2
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20	3
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7	0
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25	5
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50	12
...
475	F	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	S	Father	5	4	5
476	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	F	Father	50	77	14
477	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	S	Father	55	74	25
478	F	Jordan	Jordan	MiddleSchool	G-08	A	History	F	Father	30	17	14
479	F	Jordan	Jordan	MiddleSchool	G-08	A	History	S	Father	35	14	23

480 rows × 13 columns

([https://www.kaggle.com/aljarah/xAPI-Edu-Data- Students' Academic Performance Dataset](https://www.kaggle.com/aljarah/xAPI-Edu-Data-Students' Academic Performance Dataset))

1.describe()

DS PRACTICAL-1.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

RAM Disk Editing

```
#5.1
#describe
df.describe()
```

	raisedhands	VisITedResources	AnnouncementsView	Discussion
count	480.000000	480.000000	480.000000	480.000000
mean	46.775000	54.797917	37.918750	43.283333
std	30.779223	33.080007	26.611244	27.637735
min	0.000000	0.000000	0.000000	1.000000
25%	15.750000	20.000000	14.000000	20.000000
50%	50.000000	65.000000	33.000000	39.000000
75%	75.000000	84.000000	58.000000	70.000000
max	100.000000	99.000000	98.000000	99.000000

2. Mean()

DS PRACTICAL-1.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

RAM Disk Editing

```
[ ] #5.2
#mean
df.mean()
```

raisedhands	46.775000
VisITedResources	54.797917
AnnouncementsView	37.918750
Discussion	43.283333
dtype:	float64

3. Median()

DS PRACTICAL-1.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

RAM Disk Editing

```
[ ] #5.3
#median
df.median()
```

raisedhands	50.0
VisITedResources	65.0
AnnouncementsView	33.0
Discussion	39.0
dtype:	float64

4. Slicing operations

DS PRACTICAL-1.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

RAM Disk Editing

```
[ ] #5.4
#Slicing
print(df.iloc[0:7,0:6])
```

	gender	NationalITY	PlaceofBirth	StageID	GradeID	SectionID
0	M	KW	KuwaIT	lowerlevel	G-04	A
1	M	KW	KuwaIT	lowerlevel	G-04	A
2	M	KW	KuwaIT	lowerlevel	G-04	A
3	M	KW	KuwaIT	lowerlevel	G-04	A
4	M	KW	KuwaIT	lowerlevel	G-04	A
5	F	KW	KuwaIT	lowerlevel	G-04	A
6	M	KW	KuwaIT	MiddleSchool	G-07	A