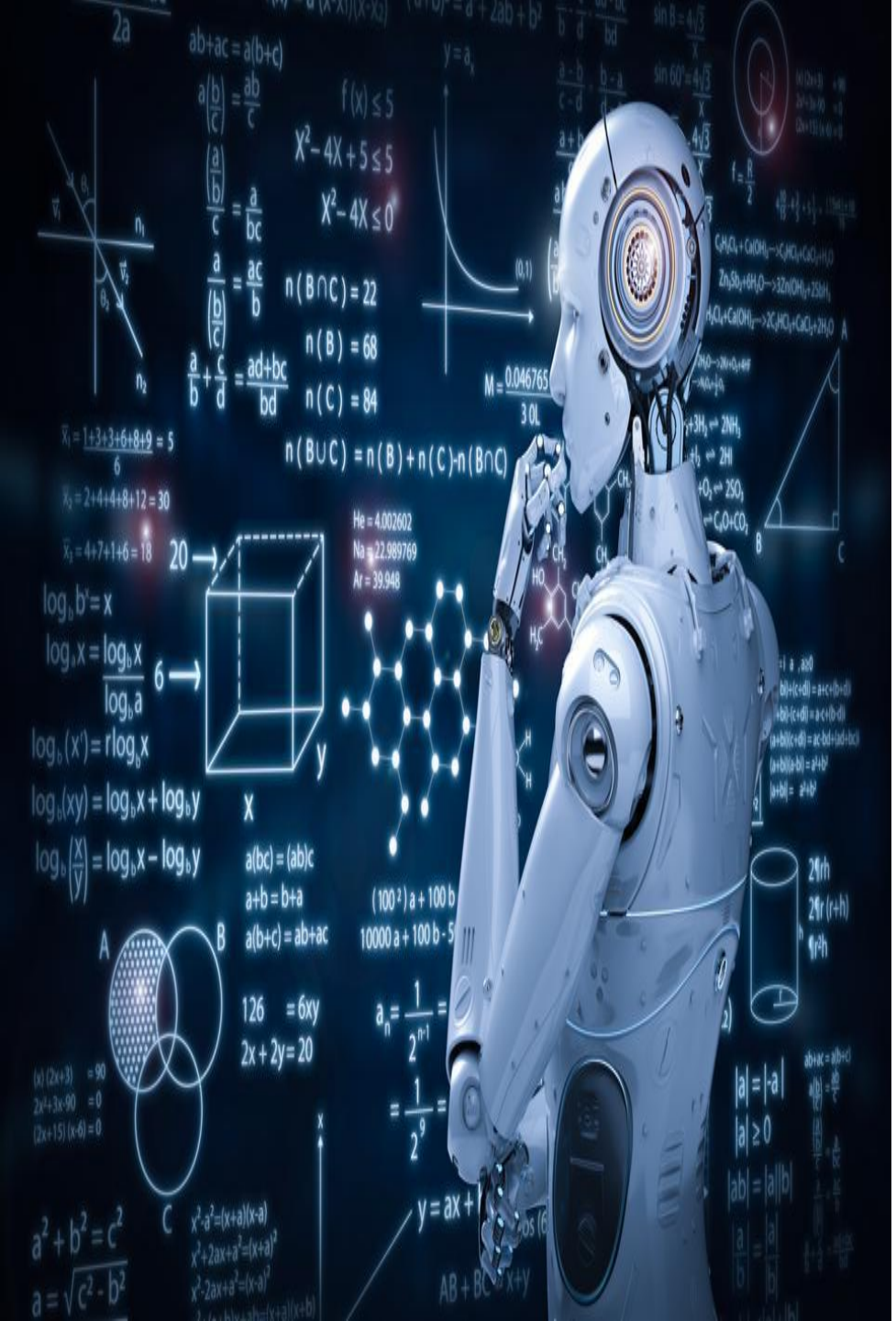


Practical Machine Learning

Day 2: Mar22 DBDA

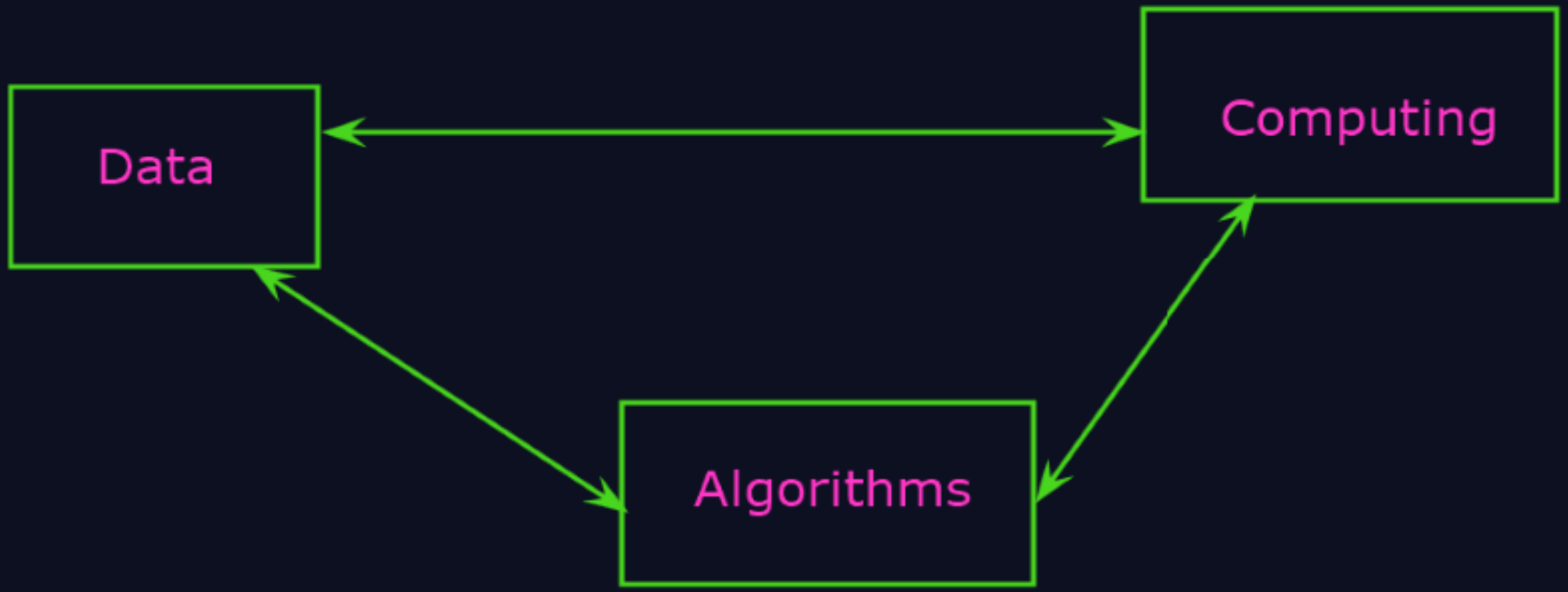
Kiran Waghmare

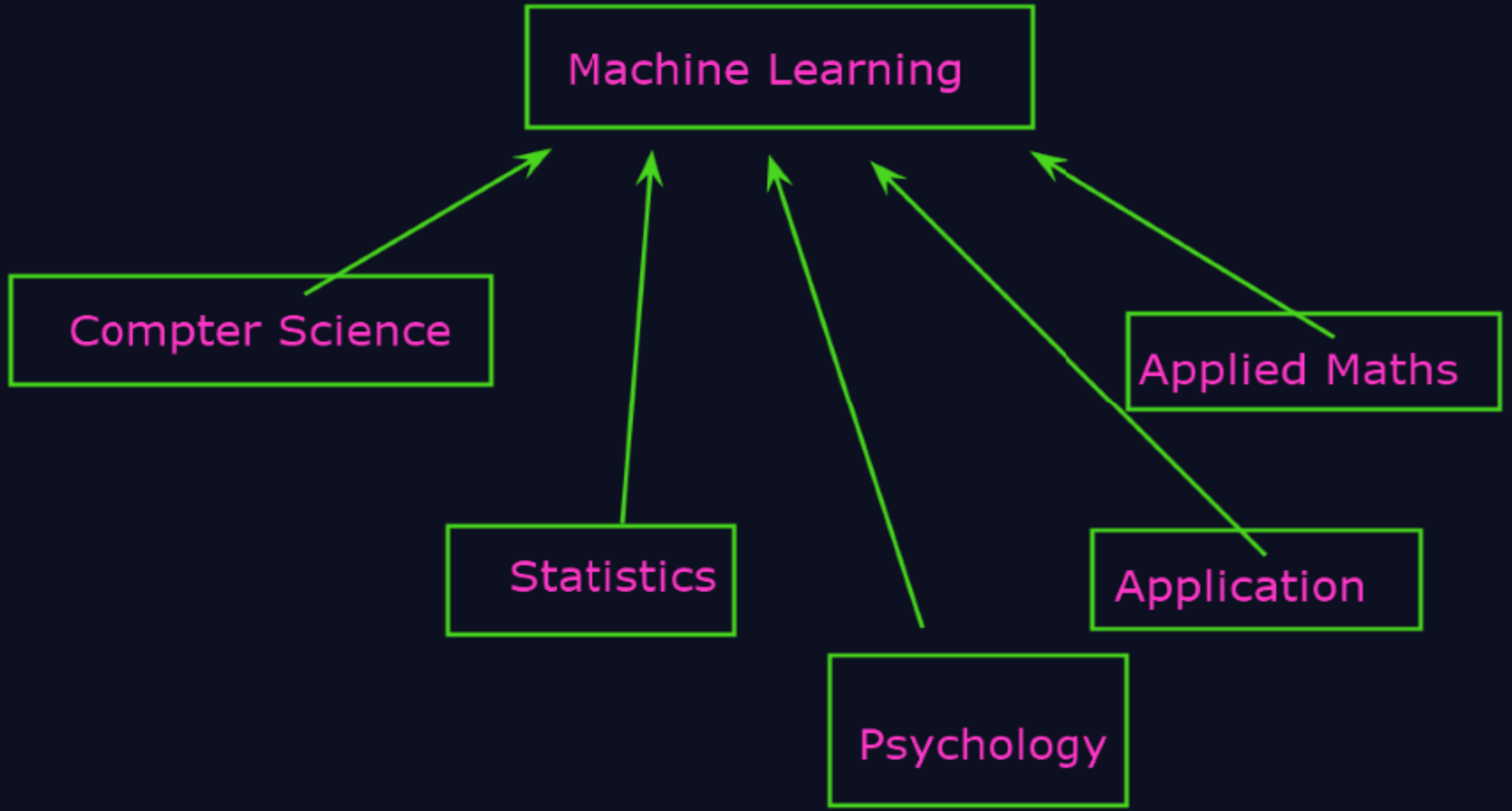
Agenda

- Evaluating ML techniques
- Python Libraries
- Introduction to Scikit Learn

Why Study Machine Learning?
The Time is Ripe

- Algorithms
  - Many basic effective and efficient algorithms available.
- Data
  - Large amounts of on-line data available.
- Computing
  - Large amounts of computational resources available.


Data
Computing
Algorithms


Machine Learning
Compter Science
Statistics
Psychology
Application
Applied Maths

Challenges and Limitations of Machine Learning

- The primary challenge of machine learning is the lack of data or the **diversity in the dataset.**
- A machine cannot learn if there **is no data available**.
- Besides, a dataset with a lack of diversity gives the machine a hard time.
- A machine **needs to have heterogeneity** to learn meaningful insight.
- It is rare that an algorithm can extract information when there are no or few variations.
- It is recommended to have at least 20 observations per group to help the machine learn.
- This constraint leads to poor evaluation and prediction.

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code Commit

In [49]:
```
tuple1=("Good","Morning","CDAC","Mumbai","Good","Good")
```

In [50]:
```
tuple1.count("Good")
```

Out[50]: 3

In [51]:
```
tuple1.index("Good")
```

Out[51]: 0

Numpy

In [52]:
```
import numpy as np
```

In [ ]:

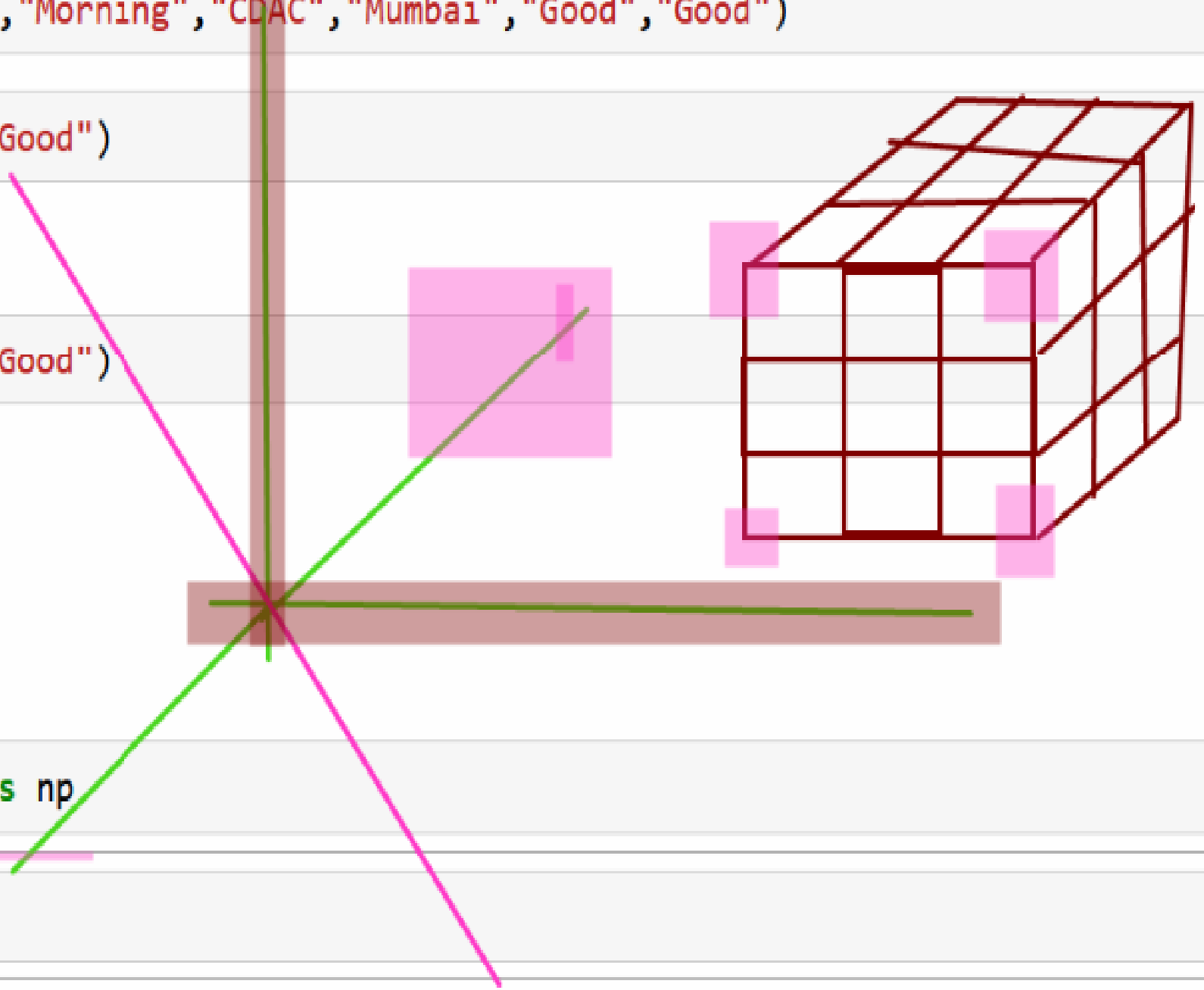
In [49]:
```
tuple1=("Good","Morning","CDAC","Mumbai","Good","Good")
```

In [50]:
```
tuple1.count("Good")
```

Out[50]: 3

In [51]:
```
tuple1.index("Good")
```

Out[51]: 0



Numpy

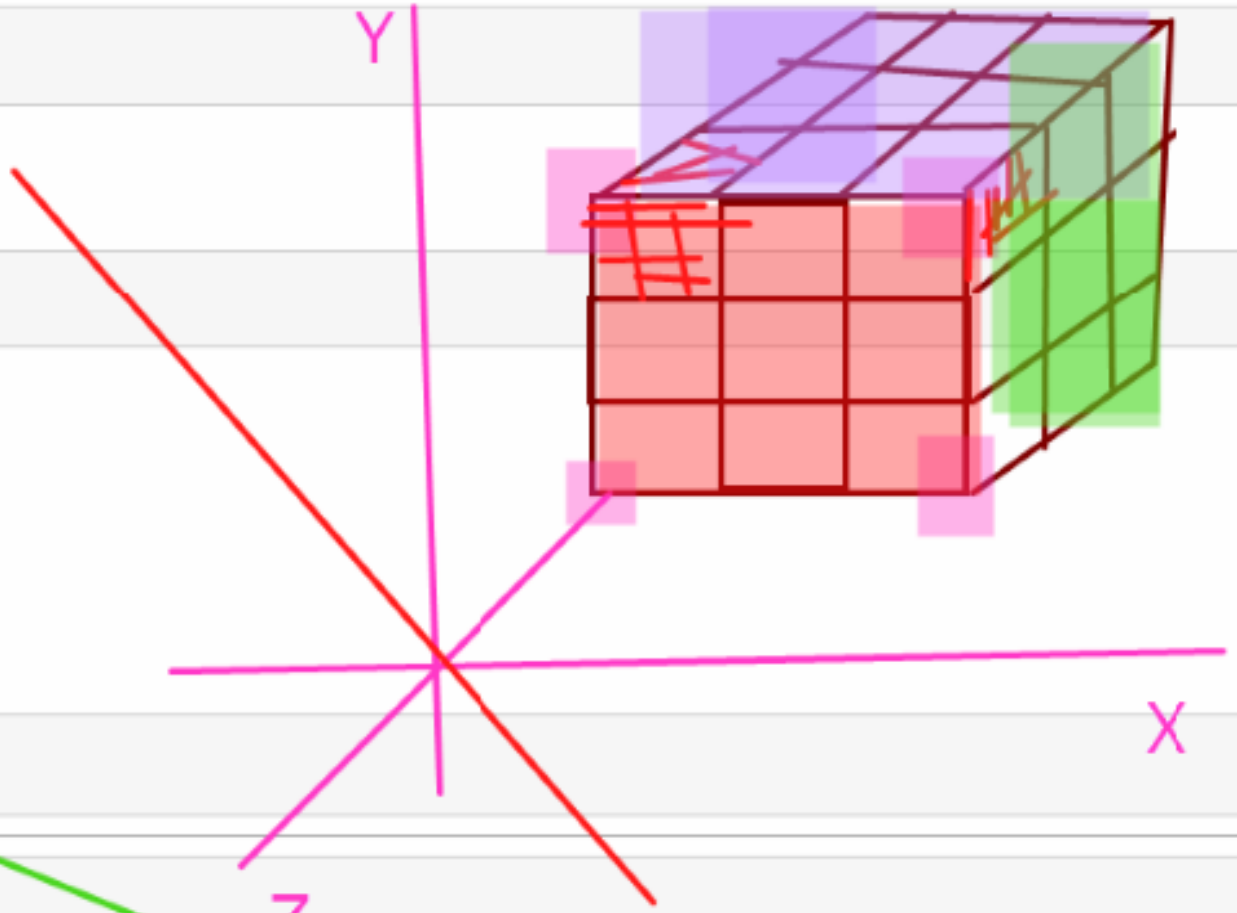
In [52]:
```
import numpy as np
```

In [ ]:

```
In [49]: tuple1=("Good","Morning","CDAC","Mumbai","Good","Good")

In [50]: tuple1.count("Good")
Out[50]: 3

In [51]: tuple1.index("Good")
Out[51]: 0
```



Numpy

```
In [52]: import numpy as np

In [ ]:
```

functionalities

High dimensional data

Trusted | Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

Run Code

Commit

In [58]:
```
lst1=[1,2,3,4,5]
lst2=[11,12,13,14,5]
lst3=[21,22,23,24,25]
```

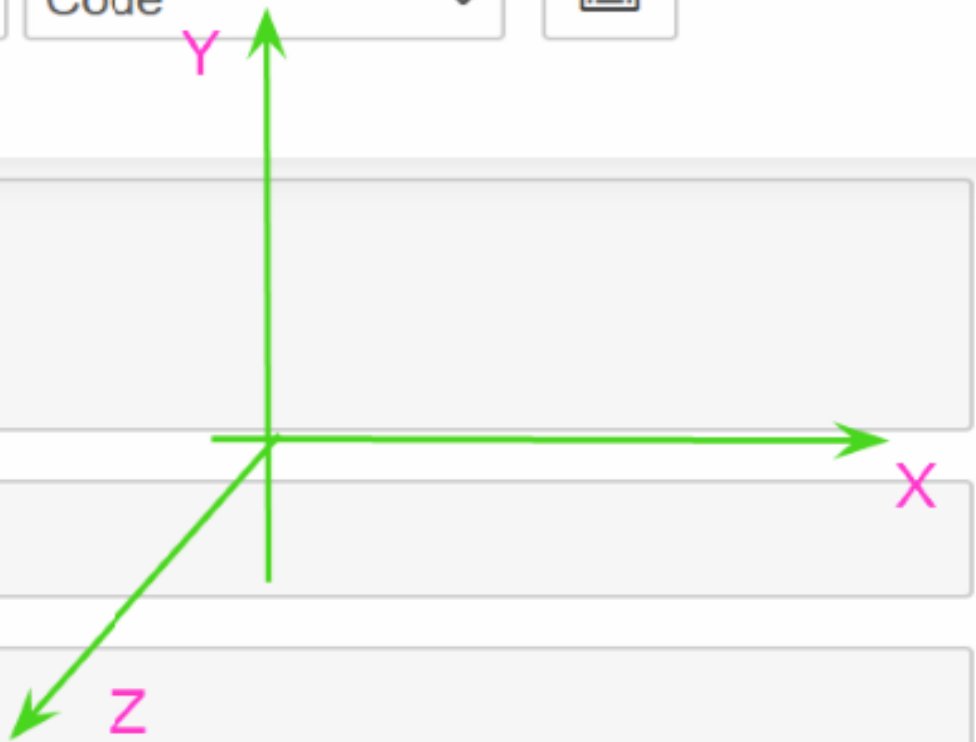
(Row,col)

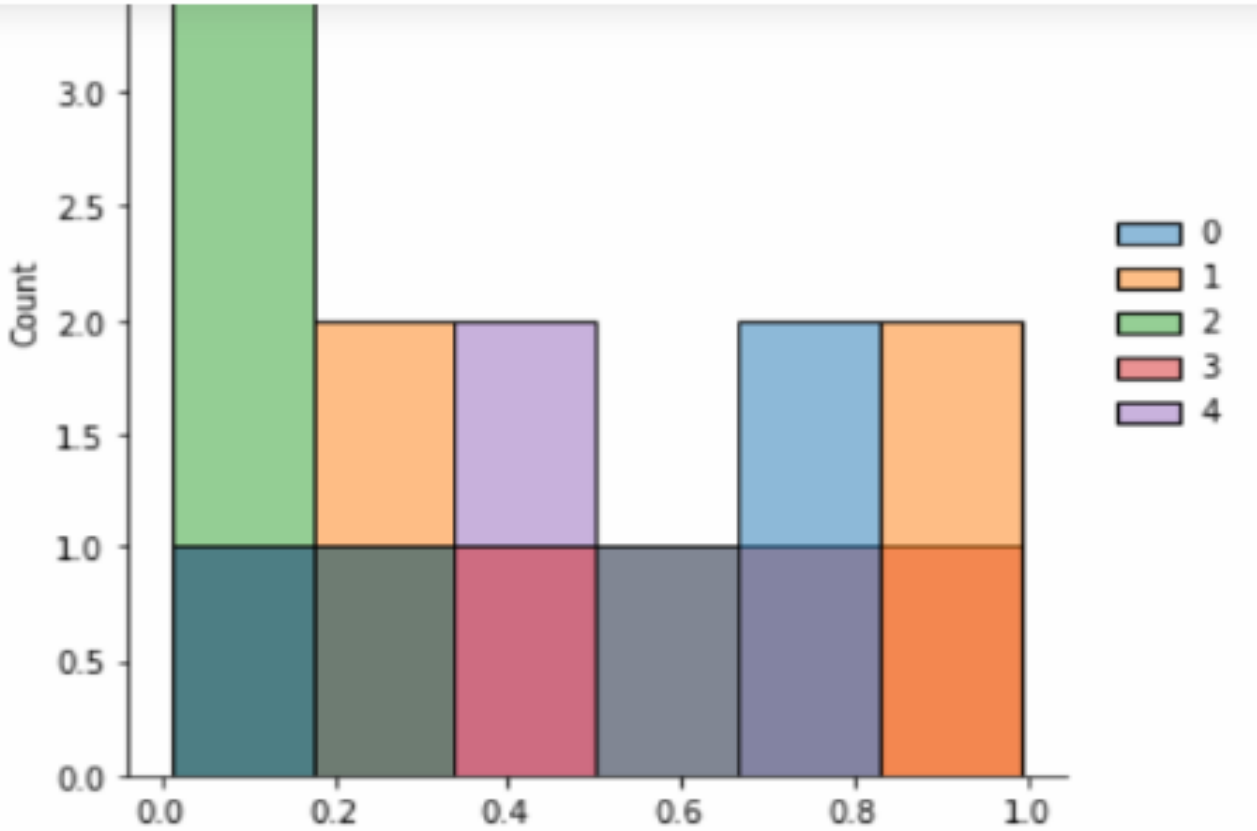
In [60]:
```
arr= np.array([lst1,lst2,lst3])
```

In [62]:
```
arr
```

Out[62]:
```
array([[ 1,  2,  3,  4,  5],
       [11, 12, 13, 14,  5],
       [21, 22, 23, 24, 25]])
```

In [61]:
```
type(arr)
```





Id	Name	%

Panda

```
In [ ]: Series: 1D : homogeneous data  ----> column we are considerig
        Dataframe: 2D hetrogeneous data ---> Tabular structure we are considering
```

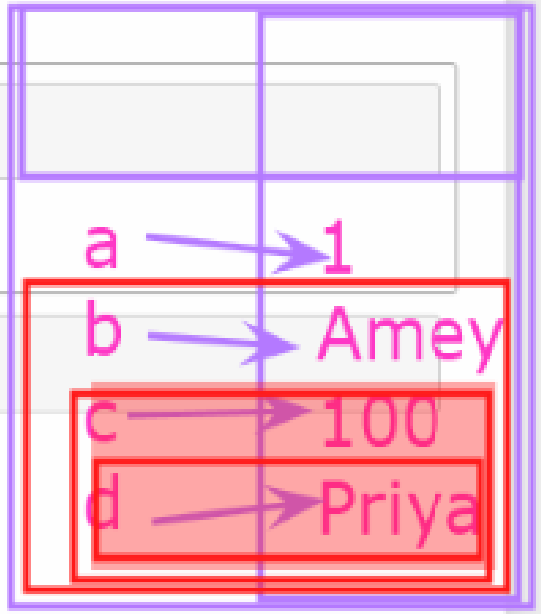
```
Out[118]: Ram      90
          Rahul    80
          Amit     85
          Tina     95
          dtype: int64
```

```
In [120]: per2=per1["Rahul"]
Out[120]: 80
```

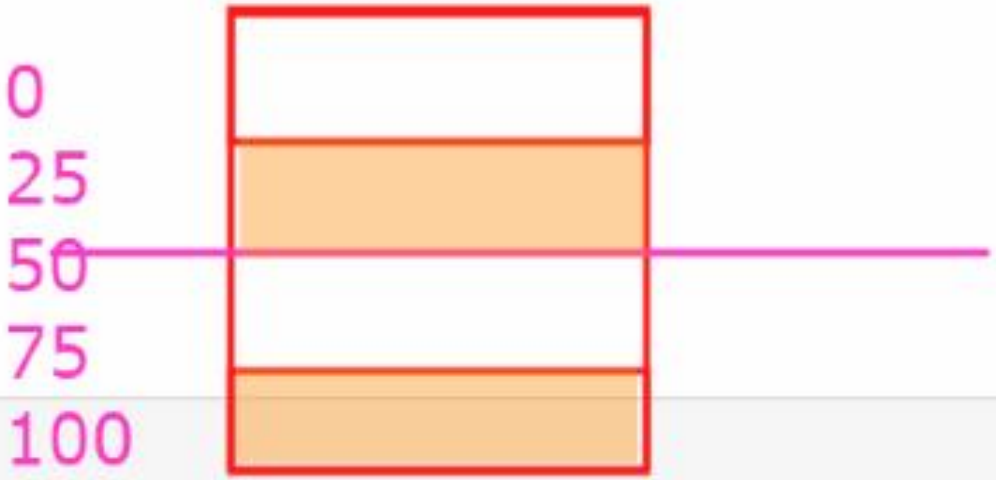
```
In [121]: per1[per1>=85]
Out[121]: Ram      90
          Amit     85
          Tina     95
          dtype: int64
```

```
In [123]: "Tina" in per1
Out[123]: True
```

```
In [ ]:
```



```
Genre           object
Publisher       object
NA_Sales       float64
EU_Sales       float64
JP_Sales       float64
Other_Sales    float64
Global_Sales   float64
dtype: object
```



In [136]:
```
games.describe()
```

Out[136]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Globa
count	16598.000000	16327.000000	16598.000000	16598.000000	16598.000000	16598.000000	16598
mean	8300.605254	2006.406443	0.264667	0.146652	0.077782	0.048063	0
std	4791.853933	5.828981	0.816683	0.505351	0.309291	0.188588	1
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0
25%	4151.250000	2003.000000	0.000000	0.000000	0.000000	0.000000	0
50%	8300.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	0
75%	12449.750000	2010.000000	0.240000	0.110000	0.040000	0.040000	0
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	82

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code Commit

```
Out[138]: Action          3316
          Sports          2346
          Misc            1739
          Role-Playing    1488
          Shooter         1310
          Adventure       1286
          Racing          1249
          Platform         886
          Simulation       867
          Fighting         848
          Strategy         681
          Puzzle           582
          Name: Genre, dtype: int64
```

```
In [139]: games.Genre.value_counts(normalize=True)
```

```
Out[139]: Action          0.199783
          Sports          0.141342
          Misc            0.104772
          Role-Playing    0.089649
          Shooter         0.078925
          Adventure       0.077479
          Racing          0.075250
          Platform        0.053380
          Simulation      0.052235
          Fighting        0.051090
          Strategy        0.041029
          Puzzle          0.035064
          Name: Genre, dtype: float64
```