# 1. Write a Python Program to compute the multiplication of two matrices and then print it.

## Steps

1. Taking Entries for  number of rows and columns of the matrices

```
r1=int(input("enter the no. of rows in matrix 1:"))
c1= int(input("columns no.:"))

r2= int(input("no. rows in matrix 2:"))
c2= int(input("no.of columns in matrix 2: "))
```

```
enter the no. of rows in matrix 1:3
columns no.:2
no. rows in matrix 2:2
no.of columns in matrix 2: 3
```

2. For multiplying to a matrix we know that the column of the 1st matrix should be equal to the row of the 2nd matrix. For that we will apply if the statement to check c1 is equal to r2. Else  it will show the statement " multiplication of two matrix is not possible".
3. If c1=r2 then : we will take inputs of matrix 1 and 2 from user using for loop and then we will perform matrix multiplication of two matrix and will display the results

```
if c1==r2:
    print("Enter the value in matrix 1 ")
    mat1=[]

    for j in range (0,r1):
        col=[]

        for i in range(0,c1):
            a=int(input())
            col.append(a)

        mat1.append(col)
    print(mat1)
```

```
print("enter values in matrix 2  ")
mat2=[]

for l in range (0,r2):
    el=[]
    for o in range (0,c2):
        b=int(input())
        el.append(b)

    mat2.append(el)

print(mat2)
```

```
result=[]

for w in range (0,r1):
    re=[]
    for u in range (0,c2):
        c=0
        re.append(c)

    result.append(re)
```

```
for i in range(0,len(result)):
    for j in range(0,len(result[0])):
        for k in range(0,len(mat2)):
            result[i][j] +=mat1[i][k] * mat2[k][j]

print("final result after multiplycation is: ")

for row in result:
    print(row)
```

output:

```
Enter the value in matrix 1
2
0
2
1
2
3
[[2, 0], [2, 1], [2, 3]]
enter values in matrix 2
2
1
2
0
2
3
[[2, 1, 2], [0, 2, 3]]
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
[4, 2, 4]
[4, 4, 7]
[4, 8, 13]
```

---

# 2. Write a Python program to find yesterday, today and tomorrow's date.

For printing dates we will be using **np.datetime64('today', 'D')** and **np.timedelta64(1, 'D')** Methods from the numpy library.

- If we just print **np.datetime64('today', 'D')** it will print today's date.
- If we add the above two methods it will print tomorrow's date and
- if we subtract them it will print yesterday's date

```
import numpy as np
yesterday = np.datetime64('today', 'D') - np.timedelta64(1, 'D')
print("Yestraday date was: ",yesterday)
today     = np.datetime64('today', 'D')
print("Today date is: ",today)
tomorrow  = np.datetime64('today', 'D') + np.timedelta64(1, 'D')
print("Tomorrow date will be: ",tomorrow)
```

```
Yestraday date was:  2021-05-18
Today date is:  2021-05-19
Tomorrow date will be:  2021-05-20
```

# 3. Write a Python program to find the Nth largest number in a list.

Steps:

1. Taking input from the user about the total number of elements in the list for that we use **for loop** to store data in the list.

```
a=int(input("Total number of element in list :" ))
list1=[]
for i in range(0,a):
    b= int(input("enter the number : "))
    list1.append(b)

print("your list is:",list1)
```

2. Then we will arrange/sort the list using **.sort()** method

```
list1.sort()
```

3. Now we take input for nth highest number in the list

```
f=int(input("nth large number you want? "))
```

4. To make sure nth highest number should not be greater than the number of elements in the list we will use an if-else **statement.** If the nth highest number is less than the number of elements in the list we will print the nth number from the list in reverse

order. Else it will show invalid entry.

```
if f<a:
    print("your nth largest number is: ",list1[-f])
else:
    print("invalid entry")
```

5. output:

```
Total number of element in list :4
enter the number : 23
enter the number : 34
enter the number : 32
enter the number : 12
your list is: [23, 34, 32, 12]
nth large number you want? 3
your nth largest number is:  23
```

# 4. Explain the NLTK module in python and execute each step involved. Document the complete process.

NLTK- an amazing library to play with Natural Language
From the above title itself you can understand NLTK is used in Natural language processing. So before studying NLTK , let's first understand what NLP is.

**Natural language processing (NLP) :**

"Natural language processing is a field concerned with the ability of a computer to understand, analyze, manipulate, and potentially generate human language ."

It is a field of AI that deals with how computers and humans interact and how to program computers to process and analyze huge amounts of natural language data. This faces some challenges like speech recognition, natural language understanding, and natural language generation.

In layman terms, NLP is a way for computers to analyze human language and derive useful insights from it. It lets you organize and structure knowledge to let you perform the following tasks-

- 
- Automatic Summarization
- 
- Translation
- 
- Named Entity Recognition
- 
- Relationship Extraction
- 
- Sentiment Analysis
- 
- Speech Recognition
- 
- Topic Segmentation

**Natural Language Toolkit (NLTK):**

NLTK stands for Natural language toolkit  consisting of the most common algorithms such as tokenizing, part-of- speech tagging, stemming, sentiment analysis, topic segmentation and named entity identification.

Here we will try to perform some simple task that we can do with NLTK:

1. **NLTK Tokenize text**: It is the process of splitting text in small parts, from paragraphs ,to sentences and from sentences to words. We have two kinds of tokenizers- for sentences and for words.

NLTK Sentence Tokenizer: In this tokenizer paragraph is divided into sentences, lets try some examples

```
>>> text="hi my name is Anand. I take time to open up with people. But once I know you, I'm fine. I'm a shy person."
>>> from nltk.tokenize import sent_tokenize
>>> sent_tokenize(text)
output:

['hi my name is Anand.', 'I take time to open up with people.',
 "But once I know you, I'm fine.",
 "I'm a shy person."]
```

Now , lets try to tokenize  language other than English:

>>> sent_tokenize("बसंती इन कुत्तों के सामने मत नाचना. यह हाथ मुझे दे दे ठाकुर! रिश्ते तो हम तुम्हारे बाप लगते हैं, नाम है शहंशाह!")
output:

['बसंती इन कुत्तों के सामने मत नाचना.',
 'यह हाथ मुझे दे दे ठाकुर!',
 'रिश्ते तो हम तुम्हारे बाप लगते हैं, नाम है शहंशाह!']
NLTK Word Tokenizer: In this tokenizer every word is tokenize .

>>> nltk.word_tokenize(text)
output:

['hi',
 'my',
 'name',
 'is',
 'Anand',
 '.',
 'I',
 'take',
 'time',
 'to',
 'open',
 'up',
 'with',
 'people',
 '.',
 'But',
 'once',
 'I',
 'know',
 'you',
 ',',
 'I',
 "'m",
 'fine',
 '.',
 'I',
 "'m",
 'a',
 'shy',
 'person',
 '.']

## 2. Find Synonyms From NLTK WordNet: WordNet is an NLP database with synonyms, antonyms, and brief definitions. We downloaded this with the NLTK downloader.

```
>>> from nltk.corpus import wordnet
>>> syn=wordnet.synsets('love')
>>> syn
```

output:

[Synset(‘love.n.01’), Synset(‘love.n.02’), Synset(‘beloved.n.01’), Synset(‘love.n.04’), Synset(‘love.n.05’), Synset(‘sexual_love.n.02’), Synset(‘love.v.01’), Synset(‘love.v.02’), Synset(‘love.v.03’), Synset(‘sleep_together.v.01’)]
we can also get their definition and example:

```
>>> syn[0].definition()
```

output:

‘a strong positive emotion of regard and affection’

```
>>> syn[0].examples()
```

output:

[‘his love for his work’, ‘children need a lot of love’]

## 3. Find Antonyms From NLTK WordNet:To get the list of antonyms, we first need to check the lemmas- are there antonyms?

```
from nltk.corpus import wordnet
antonyms=[]
for syn in wordnet.synsets('depressed'):
        for l in syn.lemmas():
                if l.antonyms():
                        antonyms.append(l.antonyms()[0].name())
antonyms
```

output:

[‘elate’]

## 4. Stemming and Lemmatization NLTK: We have talked of stemming before this. Check Stemming and Lemmatization with Python. Well, stemming involves removing affixes from words and returning the root. Search engines like Google use this to efficiently index pages. The most common algorithm for stemming is the PorterStemmer. Let's take an example.

```
>>> stemmer.stem('believes')
output:
'believ'
```

If you noticed, the output that stemming gave us wasn't an actual word you could look up in the dictionary. So we come to lemmatizing- this will return real words. Let's do this too.

```
>>> from nltk.stem import WordNetLemmatizer
>>> lemmatizer=WordNetLemmatizer()
>>> lemmatizer.lemmatize('believes')
output:
'belief'
```

Since lemmatizing gives us better results within context, it is often slower than stemming.

## 5. NLTK Stop Words: We can filter NLTK stop words from text before processing it.

```
>>> from nltk.corpus import stopwords
>>> stopwords.words('english')
output:
```

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

```
>>> from nltk.corpus import stopwords
>>> text="Today is a great day. It is even better than yesterday. And yesterday was the best day ever!"
>>> stopwords=set(stopwords.words('english'))
>>> from nltk.tokenize import word_tokenize
>>> words=word_tokenize(text)
>>> wordsFiltered=[]
>>> for w in words:
        if w not in stopwords:
                wordsFiltered.append(w)
>>> wordsFiltered
```

output:
[‘Today’, ‘great’, ‘day’, ‘.’, ‘It’, ‘even’, ‘better’, ‘yesterday’, ‘.’, ‘And’, ‘yesterday’, ‘best’, ‘day’, ‘ever’, ‘!’]

## 6. Speech Tagging : NLTK can classify words as verbs, nouns, adjectives, and more into one of the following classes:

```
>>> import nltk
>>> from nltk.tokenize import PunktSentenceTokenizer
>>> text='I am a human being, capable of doing terrible things'
>>> sentences=nltk.sent_tokenize(text)
>>> for sent in sentences:
        print(nltk.pos_tag(nltk.word_tokenize(sent)))
```

output:

[(‘I’, ‘PRP’), (‘am’, ‘VBP’), (‘a’, ‘DT’), (‘human’, ‘JJ’), (‘being’, ‘VBG’), (‘,’, ‘,’), (‘capable’, ‘JJ’), (‘of’, ‘IN’), (‘doing’, ‘VBG’), (‘terrible’, ‘JJ’), (‘things’, ‘NNS’)]
Match the arguments at position 1 with the table to figure out the output.

Conclusion:

Hence,, we learned the basics of Natural Language Processing with Python using NLTK. Moreover, we performed tasks on tokenizing, stemming, lemmatization, finding synonyms and antonyms, speech tagging, and filtering out stop words.

# 5. What is a Convolution Neural Network? Illustrate using python and document the Process.

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars.

A Convolutional Neural Network (CNN) is composed of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. Let's develop better intuition for how Convolutional Neural Networks (CNN) work. We'll examine how humans classify images, and then see how CNNs use similar approaches.

Let’s say we wanted to classify the following image of a dog as a Golden Retriever:

As humans, how do we do this?

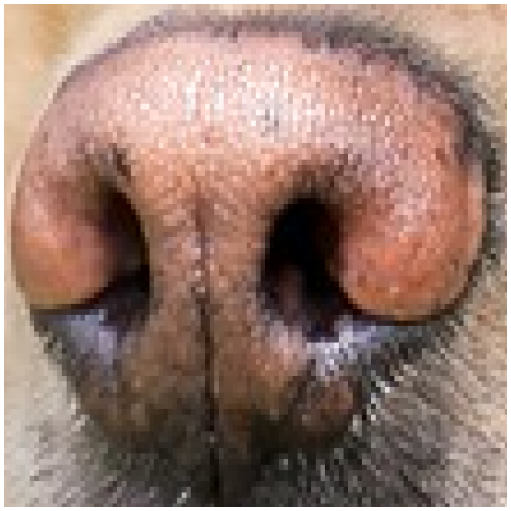
One thing we do is that we identify certain parts of the dog, such as the nose, the eyes, and the fur. We essentially break up the image into smaller pieces, recognize the smaller pieces, and then combine those pieces to get an idea of the overall dog.

In this case, we might break down the image into a combination of the following:

- A nose
- Two eyes
- Golden fur

These pieces can be seen below:

The eye of the dog.



The nose of the dog.

The fur of the dog.

## Going One Step Further

But let’s take this one step further. How do we determine what exactly a nose is? A Golden Retriever nose can be seen as an oval with two black holes inside it. Thus, one way of classifying a Retriever’s nose is to to break it up into smaller pieces and look for black holes (nostrils) and curves that define an oval as shown below:

A curve that we can use to determine a nose

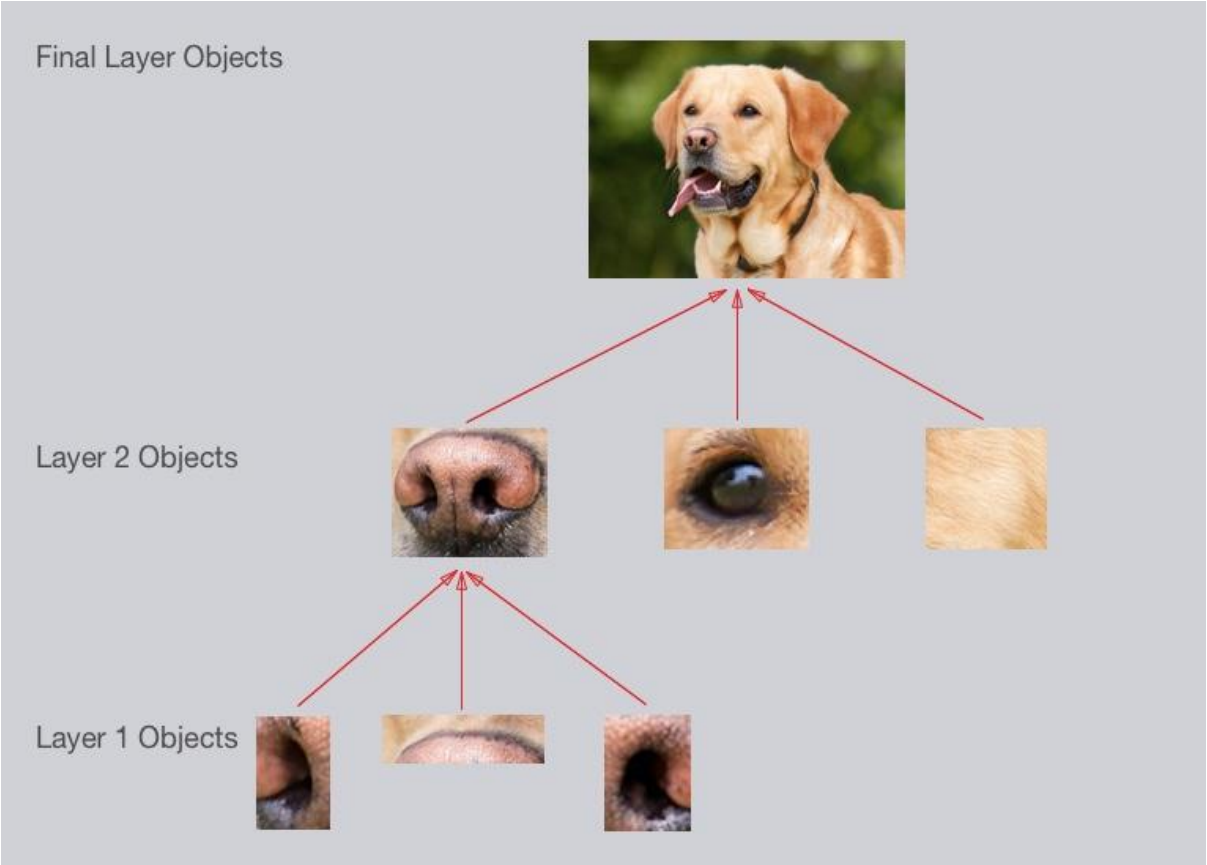
A nostril that we can use to classify the nose of the dog
Broadly speaking, this is what a CNN learns to do. It learns to recognize basic lines and curves, then shapes and blobs, and then increasingly complex objects within the image. Finally, the CNN classifies the image by combining the larger, more complex objects.

In our case, the levels in the hierarchy are:

- Simple shapes, like ovals and dark circles
- Complex objects (combinations of simple shapes), like eyes, nose, and fur
- The dog as a whole (a combination of complex objects)

With deep learning, we don't actually program the CNN to recognize these specific features. Rather, the CNN learns on its own to recognize such objects through forward propagation and backpropagation!

It's amazing how well a CNN can learn to classify images, even though we never program the CNN with information about specific features to look for.




An example of what each layer in a CNN might recognize when classifying a picture of a dog
A CNN might have several layers, and each layer might capture a different level in the hierarchy of objects. The first layer is the lowest level in the hierarchy, where the CNN generally classifies small parts of the image into simple shapes like horizontal and vertical lines and simple blobs of colors. The subsequent layers tend to be higher levels in the hierarchy and generally classify more complex ideas like shapes (combinations of lines), and eventually full objects like dogs.

Once again, CNN **learns all of this on its own**. We don't ever have to tell CNN to go looking for lines or curves or noses or fur. The CNN just learns from the training set and discovers which characteristics of a Golden Retriever are worth looking for.

Terms used in CNN:

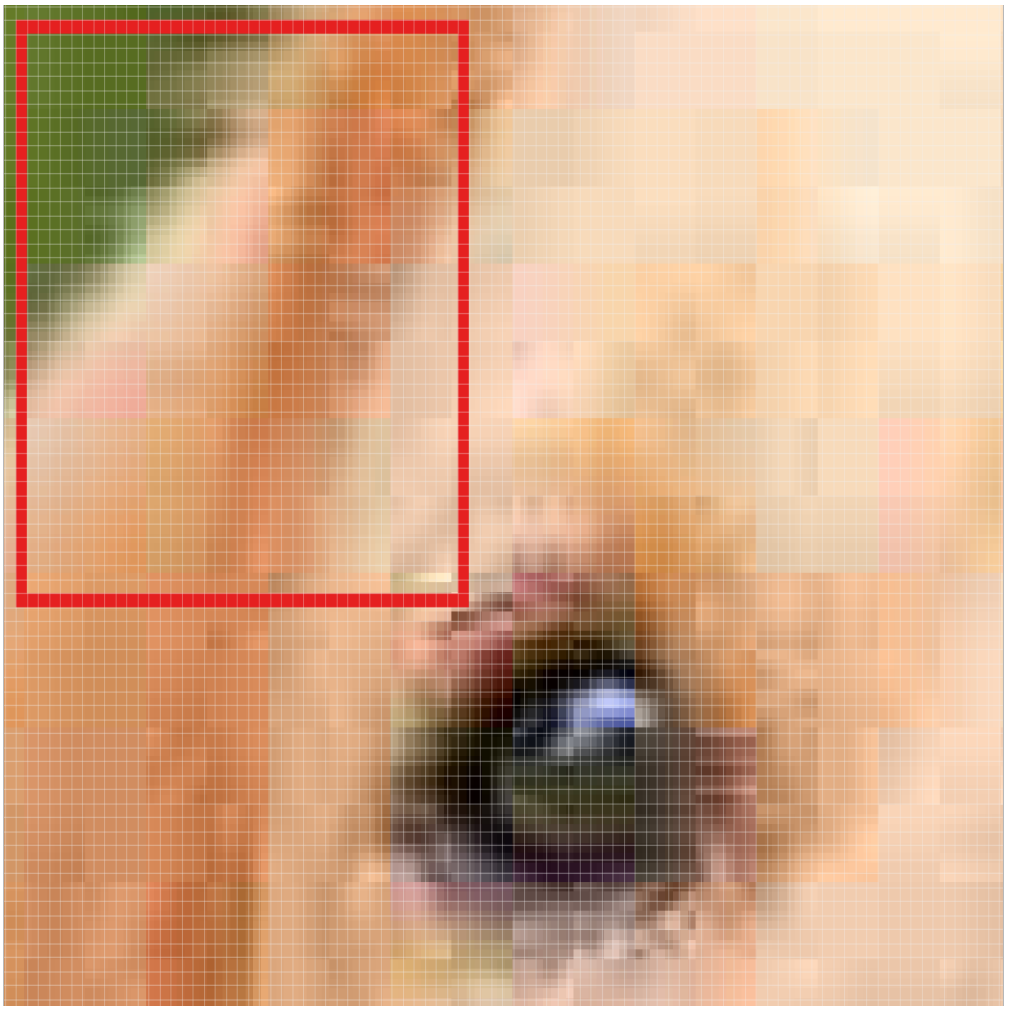
1.Filters:
The first step for a CNN is to break up the image into smaller pieces. We do this by selecting a width and height that defines a filter.

The filter looks at small pieces, or patches, of the image. These patches are the same size as the filter.

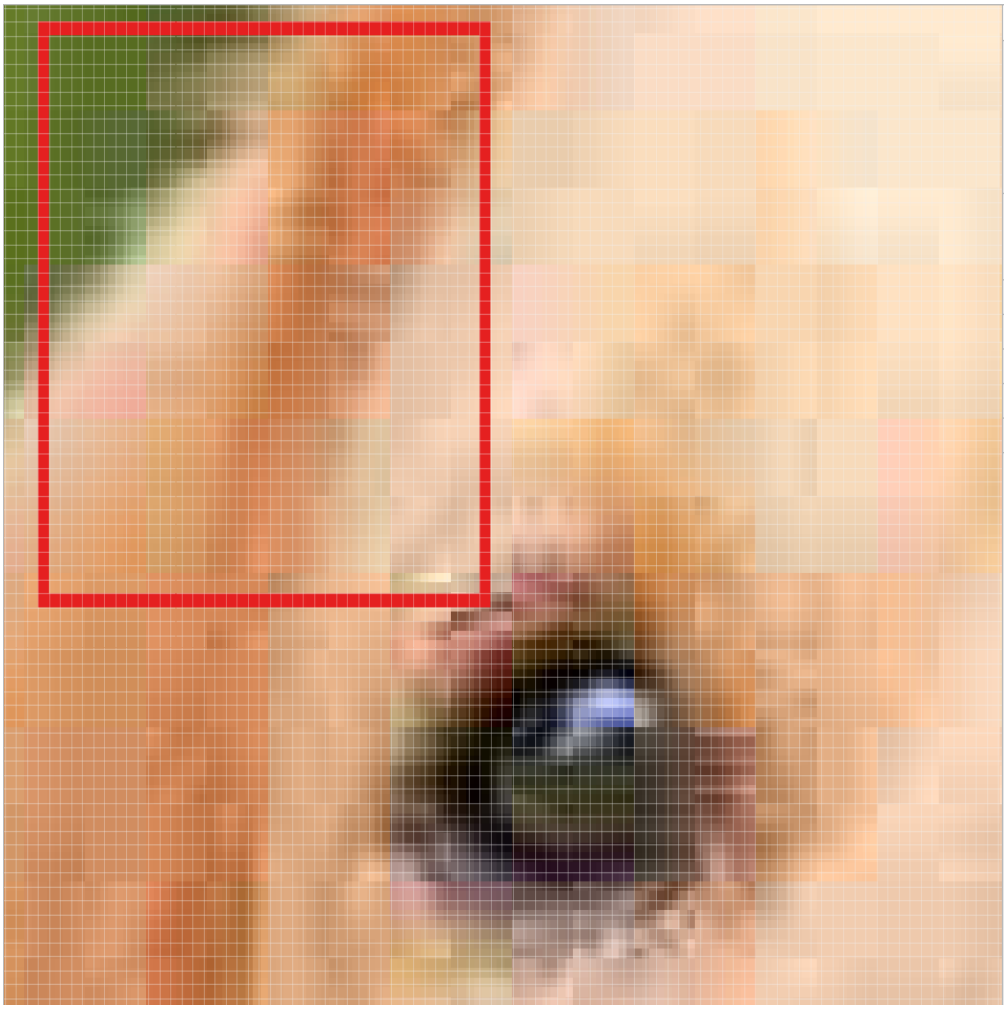
A CNN uses filters to split an image into smaller patches. The size of these patches matches the filter size.
We then simply slide this filter horizontally or vertically to focus on a different piece of the image.

The amount by which the filter slides is referred to as the **'stride'**. The stride is a hyperparameter which the engineer can tune. Increasing the stride reduces the size of your model by reducing the number of total patches each layer observes. However, this usually comes with a reduction in accuracy.

Let's look at an example. In this zoomed in image of the dog, we first start with the patch outlined in red. The width and height of our filter define the size of this square.



We then move the square over to the right by a given stride (2 in this case) to get another patch.



We move our square to the right by two pixels to create another patch.
What's important here is that we are **grouping together adjacent pixels** and treating them as a collective.

In a normal, non-convolutional neural network, we would have ignored this adjacency. In a normal network, we would have connected every pixel in the input image to a neuron in the next layer. In doing so, we would not have taken advantage of the fact that pixels in an image are close together for a reason and have special meaning.

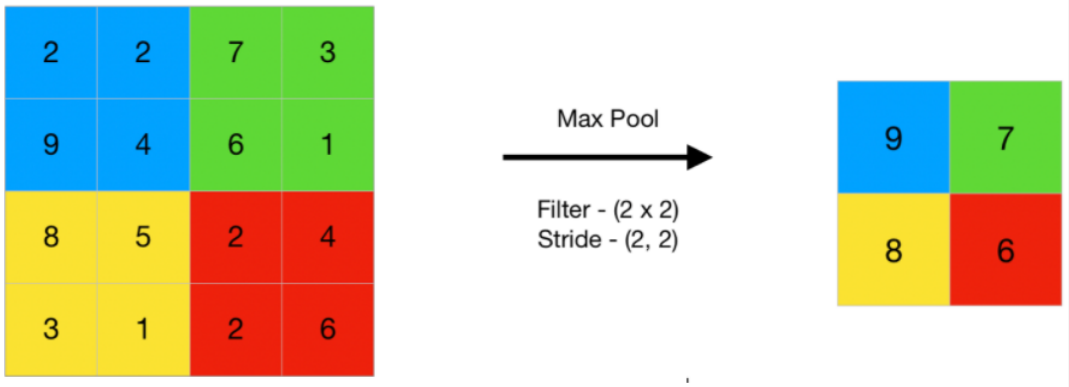
By taking advantage of this local structure, our CNN learns to classify local patterns, like shapes and objects, in an image.

2. Pooling:

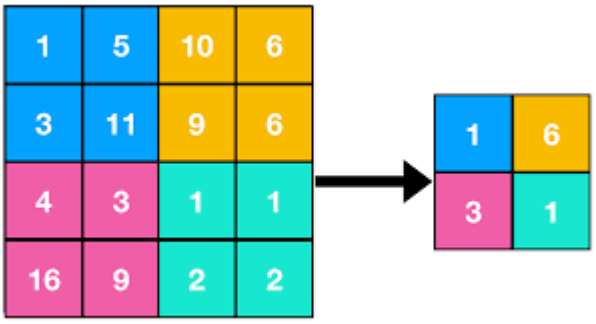
- Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.
- The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely

positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

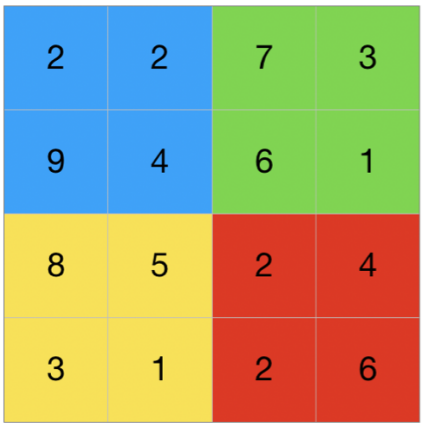
- There are 3 time of pooling:
  1. Max_pooling: It selects max elements from the region of the feature map covered by the filter. As we can see below if filter is of size (2*2) so 1st it will cover the blue region of the matrix and it will take the highest element from that region and it will store in the new matrix. After that it will take stride of 2 and move in the green region and will take the highest element from that region and will store it in a new matrix. Similarly it will go in yellow as well as red regions and take the highest element from that region.



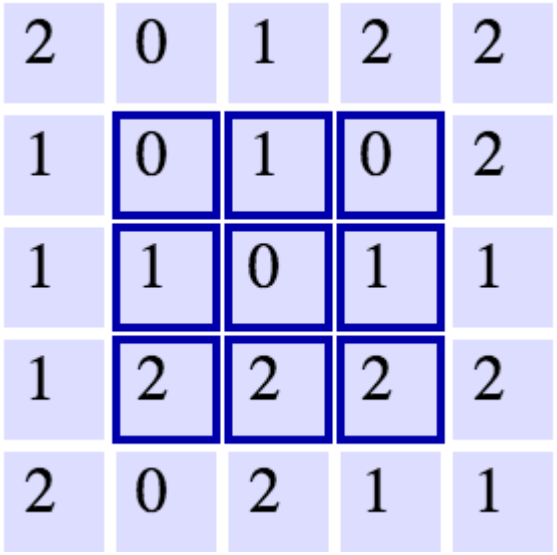

  2. Min_pooling:It goes in the same process as max_pooling, but the difference is ,“instead of maximum value it takes minimum value from the region. You can see from below



3. Avg_pooling: in this technique we take the average of all the elements in the region of the filter.

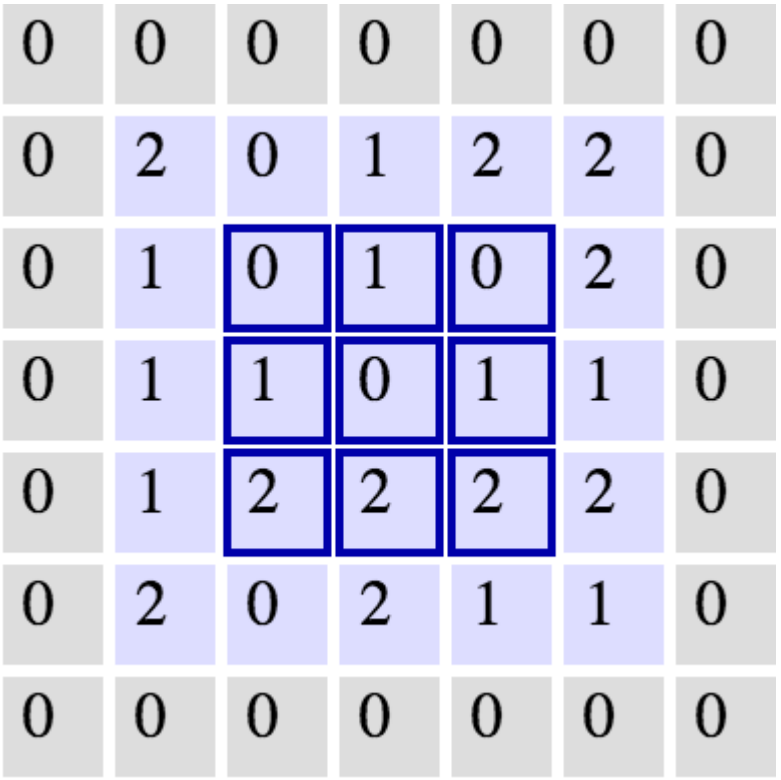



3.padding:



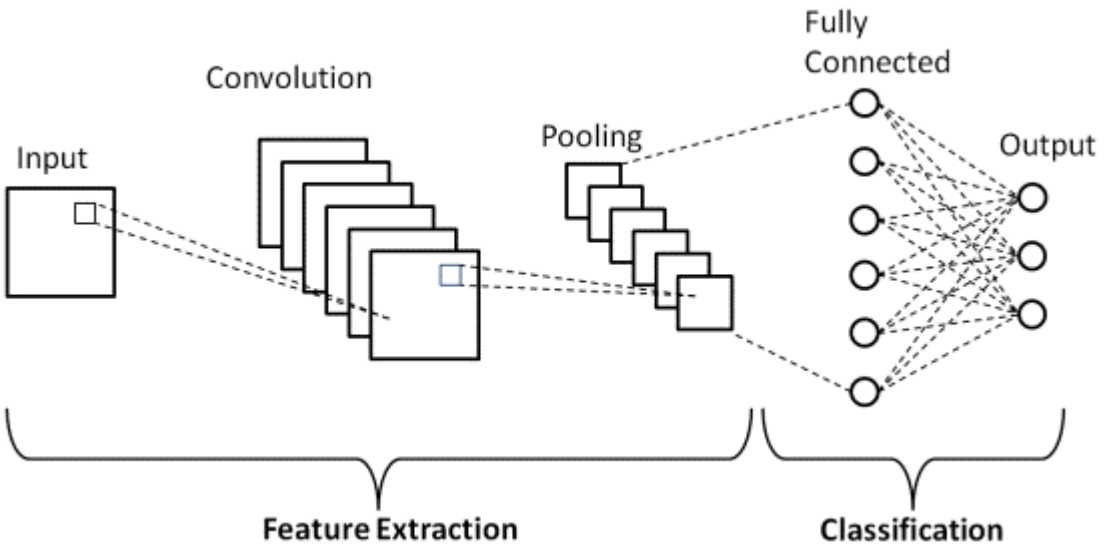

Let's say we have a 5x5 grid (as shown above) and a filter of size 3x3 with a stride of 1. What's the width and height of the next layer? We see that we can fit at most three patches in each direction, giving us a dimension of 3x3 in our next layer. As we can see, the width and height of each subsequent layer decreases in such a scheme.

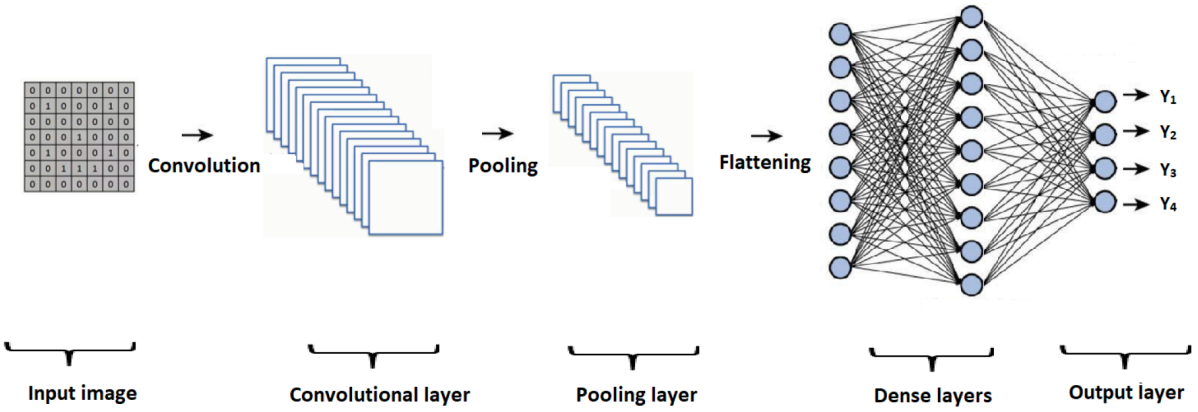
In an ideal world, we'd be able to maintain the same width and height across layers so that we can continue to add layers without worrying about the dimensionality shrinking and so that we have consistency. How might we achieve this? One way is to simple add a border of `0`s to our original `5x5` image. You can see what this looks like in the below image:




This would expand our original image to a `7x7`. With this, we now see how our next layer's size is again a `5x5`, keeping our dimensionality consistent.

Now let's try to understand **CNN through its architecture:**




Let's understand processes in CNN from the above figures

- First we take image  in the matrix form lets say 5*5 or 7*7, it can be of any dimensional
- Image identification is of two parks **1. Feature extraction 2. Image classification**
    1. **Feature extraction** : in this step we apply layers of Convolutional and pooling which completely depends on us. Whatever outcomes in the matrix we then flatten them in an array .
    2. **Classification**: flatten array from the feature extraction is passed as an input in a dense layer and classification of image is done.

Let's make a simple model to identify whether the given image is Dod or cat:(since my laptop's ram is 4gb i cant train large no. of data so i took 25 images of each cat and dogs for training and testing… so accuracy will we less)

## Steps:

1. Import all necessary libraries

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
import numpy as np
from keras.preprocessing import image

import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import ImageDataGenerator,load_img
from PIL import Image
```

Sequential is to make pipeline of layers
Conv2d is to apply convolutional layer
Maxpooling2d for dimensional reductionality
Flatten is for flatten the the matrix into array
Dense for dense network
ImageDataGenerator is for image augmentation .It is generally used when we have less dataset .

2. Initialising the CNN:

```
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a second convolutional layer




# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 1, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Here you can see first we created classifier method using Sequential() method ,
Then we add convolution layers and pooling. We can add more convolutional as well as pooling layers depending on us in any combination we want.

After that we applied flatten to make the matrix into a single array.
And that array is input in a dense layer where we use relu and atlast we use sigmoid function to classify the image. And then we applied an adam optimiser and loss function to learn the model.

3. Fitting the cnn to the images:

```
# Part 2 - Fitting the CNN to the images
```

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
training_set = train_datagen.flow_from_directory('C:\\parent',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory('C:\\parent',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')

classifier.fit_generator(training_set,
                         steps_per_epoch = 8000,
                         epochs = 4,
                         validation_data = test_set,
                         validation_steps = 2000)
```

Here we do image augmentation with the help of ImageDataGenerator we also rescale our image rgb value from 0-255 to 0-1. So that we can process our image quickly. Otherwise it will take lots of time to process our image.

Output: accuracy is less because data is small and layers are less.

```
Found 50 images belonging to 2 classes.
Found 50 images belonging to 2 classes.
WARNING:tensorflow:From <ipython-input-6-0b7cbaa974a7>:23: Model.fit_generator (from tensorflow.python.keras.engine.trainin
g) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/4
   2/8000 [..............................] - ETA: 7:54 - loss: 2.9019 - accuracy: 0.4000WARNING:tensorflow:Your input ran ou
t of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs`
batches (in this case, 32000 batches). You may need to use the repeat() function when building your dataset.
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate
at least `steps_per_epoch * epochs` batches (in this case, 2000 batches). You may need to use the repeat() function when bui
lding your dataset.
   2/8000 [..............................] - 1s 412ms/step - loss: 2.9019 - accuracy: 0.4000 - val_loss: 1.0017 - val_accura
cy: 0.5000
```

4. Lets predict our model we an image of cat:

```
test_image = image.load_img('C:\\Users\\Prakash Jha\\downloads\\cat1.jpg', target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'dog'
    print(prediction)
else:
    prediction = 'cat'
    print(prediction)
```

Output:

```
test_image = image.load_img('C:\\Users\\Prakash Jha\\downloads\\cat1.jpg', target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'dog'
    print(prediction)
else:
    prediction = 'cat'
    print(prediction)
```

```
cat
```

So we got the value cat. We are correct!

# 6. Implement NLP analysis of a restaurant review in python.

## Steps:

1. Import the important libraries:

```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
```

We will be using NLTK , numpy ,pandas ,sklearn libraries and we will also use RandomForestClassifier and kneighborsClassifier

2.Load and read the data:

```
dataset = pd.read_csv('C:\\Users\\Prakash Jha\\Downloads\\Restaurant_Reviews.tsv', delimiter = '\t')
```

```
dataset
```

| | Review | Liked |
|---|---|---|
| **0** | Wow... Loved this place. | 1 |
| **1** | Crust is not good. | 0 |
| **2** | Not tasty and the texture was just nasty. | 0 |
| **3** | Stopped by during the late May bank holiday of... | 1 |
| **4** | The selection on the menu was great and so wer... | 1 |
| **...** | ... | ... |
| **995** | I think food should have flavor and texture an... | 0 |
| **996** | Appetite instantly gone. | 0 |
| **997** | Overall I was not impressed and would not go b... | 0 |

3. Text cleaning and preprocessing:

We will remove punctuations, numbers and with help of stemming we will obtain root words. And we will also convert all words or sentences in lower case to avoid repeating words.

```
from nltk.stem.porter import PorterStemmer

# Initialize empty array
# to append clean text
corpus = []

# 1000 (reviews) rows to clean
for i in range(0, 1000):

    # column : "Review", row ith
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])

    # convert all cases to lower cases
    review = review.lower()

    # split to array(default delimiter is " ")
    review = review.split()

    # creating PorterStemmer object to
    # take main stem of each word
    ps = PorterStemmer()

    # loop for stemming each word
    # in string array at ith row
    review = [ps.stem(word) for word in review
                if not word in set(stopwords.words('english'))]

    # rejoin all string array elements
    # to create back into a string
    review = ' '.join(review)

    # append each string to create
    # array of clean text
    corpus.append(review)
```

4. Tokenization  and making bag of words using sparse  matrix

```
# To extract max 1500 feature.
# "max_features" is attribute to
# experiment with to get better results
cv = CountVectorizer(max_features = 1500)

# X contains corpus (dependent variable)
X = cv.fit_transform(corpus).toarray()

# y contains answers if review
# is positive or negative
y = dataset.iloc[:, 1].values
```

5. Splitting Corpus into Training and Test set. For this, we need class train_test_split from sklearn.cross_validation. Split can be made 70/30 or 80/20 or 85/15 or 75/25, here I choose 75/25 via “test_size”.
X is the bag of words, y is 0 or 1 (positive or negative).

```
# Splitting the dataset into
# the Training set and Test set


# experiment with "test_size"
# to get better results
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

6. Fitting a Predictive Model (here random forest and kNN):

```
from sklearn.ensemble import RandomForestClassifier

# n_estimators can be said as number of
# trees, experiment with n_estimators
# to get better results
model = RandomForestClassifier(n_estimators = 501,
                               criterion = 'entropy')

model.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=501)
```

```
y_pred = model.predict(X_test)

y_pred
```

```
array([1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0], dtype=int64)
```

```
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0], dtype=int64)
```

```
from sklearn import metrics
```

```
acc=metrics.accuracy_score(y_pred,y_test)
```

```
print(acc)
```

```
0.7533333333333333
```

Here random forest gives us accuracy of 75 %
Lets see what KNN gives\:

```
from sklearn.neighbors import KNeighborsClassifier
max=0
for k in range(1,30):
    knn_obj=KNeighborsClassifier(n_neighbors=k)
    knn_obj.fit(X_train,y_train)
    pred=knn_obj.predict(X_test)
    acc=metrics.accuracy_score(pred,y_test)
    print(k,"=",acc)
    if max<acc:
        max=acc
        kvalue=k
print("max accuracy is at value k is ",kvalue)
print("max accuracy is",max)
```

```
1 = 0.6533333333333333
2 = 0.64
3 = 0.7066666666666667
4 = 0.63
5 = 0.6533333333333333
6 = 0.6466666666666666
7 = 0.6466666666666666
8 = 0.62
9 = 0.64
10 = 0.5933333333333334
11 = 0.6166666666666667
12 = 0.59
13 = 0.62
14 = 0.56
15 = 0.61
16 = 0.5833333333333334
17 = 0.5933333333333334
18 = 0.5833333333333334
19 = 0.5833333333333334
20 = 0.5666666666666666
21 = 0.5733333333333334
22 = 0.5633333333333334
23 = 0.59
24 = 0.5533333333333333
25 = 0.5633333333333334
26 = 0.5533333333333333
27 = 0.5633333333333334
28 = 0.5533333333333333
29 = 0.56
max accuracy is at value k is  3
max accuracy is 0.7066666666666667
```

KNN gives accuracy of 70 %

So we will use the Random forest model which  has given us 75 % of accuracy.

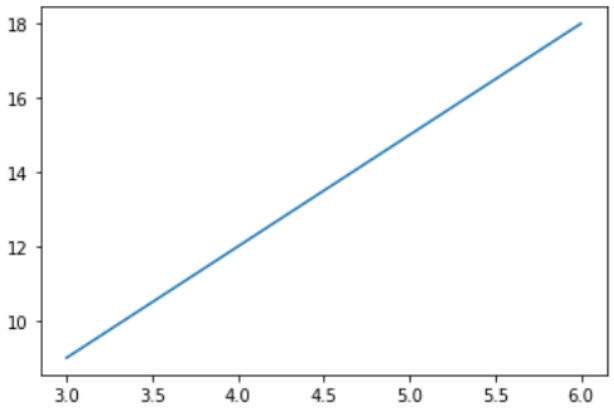
# 7. Create Line Graph, Bar chart, Histograms, Scatter plot, Pie Chart, 3D plots using matplotlib in python.

Import libraries:

```
import matplotlib.pyplot as plt
import numpy as np
```

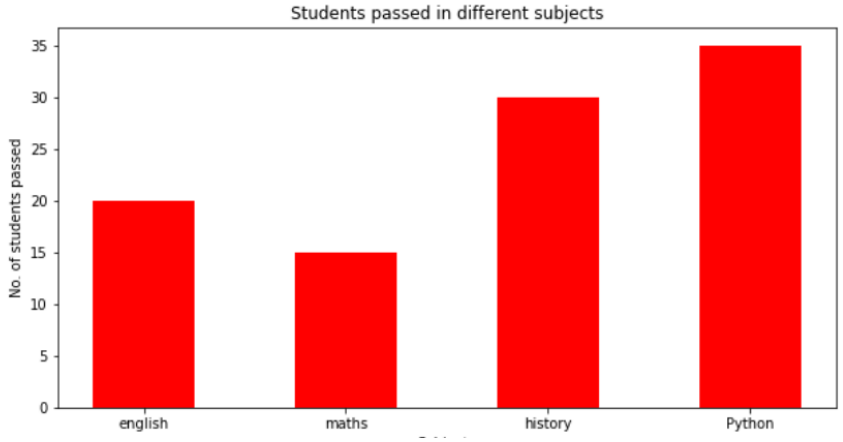
1. Line graph:

```
# line graph
x = np.array([3, 4, 5, 6])
y = x*3

plt.plot(x, y)
plt.show()
```



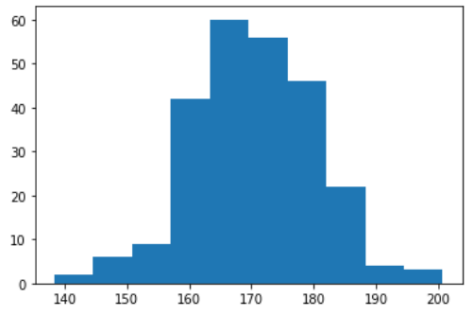
2. Bar graph:

```
#bar graph

data = {'english':20, 'maths':15, 'history':30,
        'Python':35}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))


plt.bar(courses, values, color ='Red',
        width = 0.5)

plt.xlabel("Subjects")
plt.ylabel("No. of students passed")
plt.title("Students passed in different subjects")
plt.show()
```



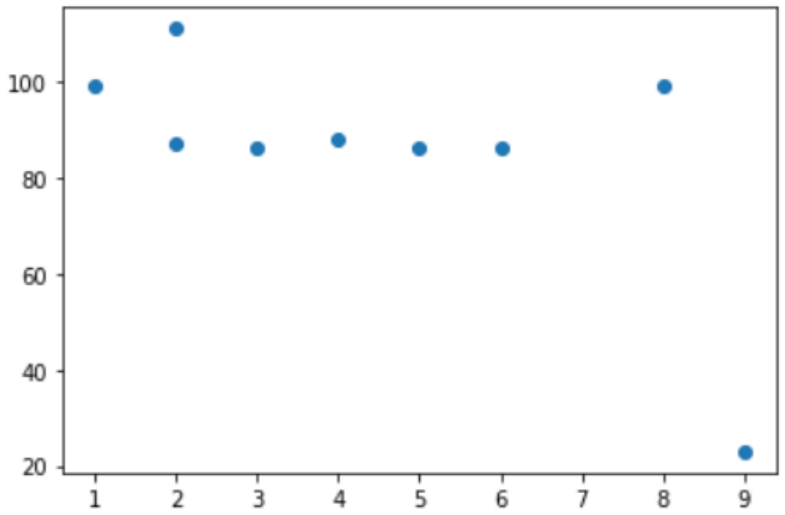

3. Histograms:

```
#histogram graph
x = np.random.normal(170, 10, 250) # 250 value is genrated with standard deviation of 10 and concentrated around 170

plt.hist(x)
plt.show()
```



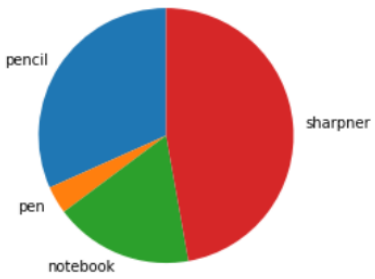
4. Scatter plot:

```
x = np.array([1,3,2,4,2,5,6,8,9])
y = np.array([99,86,87,88,111,86,86,99,23])

plt.scatter(x, y)
plt.show()
```




5. Pie Chart:

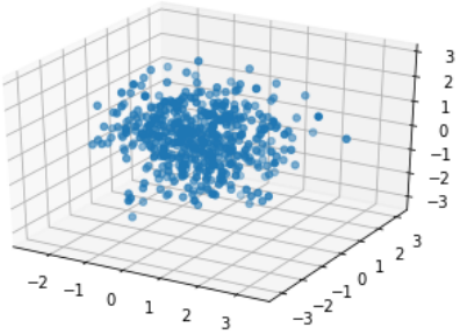
```
# pie chart

x = np.array([45, 5, 25, 67])
attributes = ["pencil", "pen", "notebook", "sharpner"]

plt.pie(x, labels = attributes, startangle = 90)
plt.show()
```



6. 3D plot:

```
# 3d plot
from mpl_toolkits.mplot3d import Axes3D #to make 3 axis
x=np.random.normal(size=500)
y=np.random.normal(size=500)
z=np.random.normal(size=500)
fig=plt.figure() #plot figure
ax=fig.add_subplot(111,projection='3d')
ax.scatter(x,y,z)
```

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1d7afbb4208>
```




# 8. Implement a class in python and explain all the properties of a class using OOPs in python.

## 1. Class:

- Class is the blueprint of the object , and it has all the variables and methods of the objects of the same kind.
- It uses class keyword to represent class

```
# class
class person:
    pass
```

- All classes have a function called __init__(), which is always executed when the class is being initiated.
- The __init__() function is called automatically every time the class is being used to create a new object.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

```
John
36
```

## 2.Object:

- It is an instantiation of a class
- When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

```
John
36
```

Here p1 is an object to the class person.

## 3.methods: Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

#Use the Person class to create an object, and then execute the printname method:

x = Person("John", "Doe")
x.printname()
```

```
John Doe
```

Here printname is method

## 4.Inheritance:

- It provides code reusability.
- In this child class inherits all the data variable and methods from its parent class
- Python support both multilevel and multiple inheritance

```
class Animal:
    def speak(self):
        print("Animal Speaking")
#child class Dog inherits the base class Animal
class Dog(Animal):
    def bark(self):
        print("dog barking")
d = Dog()
d.bark()
d.speak()
```

```
dog barking
Animal Speaking
```

Here class Dog inherits class Animal

```
class Animal:
    def speak(self):
        print("Animal Speaking")
#The child class Dog inherits the base class Animal
class Dog(Animal):
    def bark(self):
        print("dog barking")
#The child class Dogchild inherits another child class Dog
class DogChild(Dog):
    def eat(self):
        print("Eating bread...")
d = DogChild()
d.bark()
d.speak()
d.eat()
```

```
dog barking
Animal Speaking
Eating bread...
```

Above snap is example of Multilevel inheritance where DogChild inherits Dog which Inherits Animal.

```
class Calculation1:
    def Summation(self,a,b):
        return a+b;
class Calculation2:
    def Multiplication(self,a,b):
        return a*b;
class Derived(Calculation1,Calculation2):
    def Divide(self,a,b):
        return a/b;
d = Derived()
print(d.Summation(10,20))
print(d.Multiplication(10,20))
print(d.Divide(10,20))
```

```
30
200
0.5
```

Above snap explains about Multiple inheritance where class derived inherits class calculation1 and calculation2.