```
# Experiment 5 :
## Aim : To perform convolution of two Signals using python
## Software used  : Python
## IDE : Google Colab

# Completed By
Name : Atul Kumar Dwivedi

Roll Number :11111111

Branch : Electronics and Communication Engineering

Semester : 4

Lab: Signals and Systems (KEC 451)

Date of Completion 15/02/2023
```

Experiment 5 :

Aim : To perform convolution of two Signals using python

Software used : Python

IDE : Google Colab

Completed By

Name : Atul Kumar Dwivedi

Roll Number :11111111

Branch : Electronics and Communication Engineering

Semester : 4

Lab: Signals and Systems (KEC 451)

Date of Completion 15/02/2023

```
import numpy as np

#convolution function
def conv1D(x,w, p=0 , s=1):
  '''
  x : input vector
  w : filter
  p : padding size
  s : stride
  '''
  assert len(w) <= len(x), "x should be bigger than w"
  assert p >= 0, "padding cannot be negative"

  w_r = np.array(w[::-1]) #rotation of w
  x_padded = np.array(x)

  if p > 0 :
    zeros = np.zeros(shape = p)
    x_padded = np.concatenate([zeros, x_padded, zeros]) #add zeros around original vector

  out = []
  #iterate through the original array s cells per step
  for i in range(0, int((len(x_padded) - len(w_r))) + 1 , s):
    out.append(np.sum(x_padded[i:i + w_r.shape[0]] * w_r)) #formula we have seen before
  return np.array(out)
```

Let's try running this function on some real data and see the results. Let's compare the result with NumPy's built-in function that calculates the convolution result automatically.

```
x = [1,1,2]
w = [1,1,1]

conv1D(x,w,2,1)
```

```
array([1., 2., 4., 3., 2.])
```

using built in function

```
np.convolve(x , w)
```

```
array([1, 2, 4, 3, 2])
```

Colab paid products - Cancel contracts here