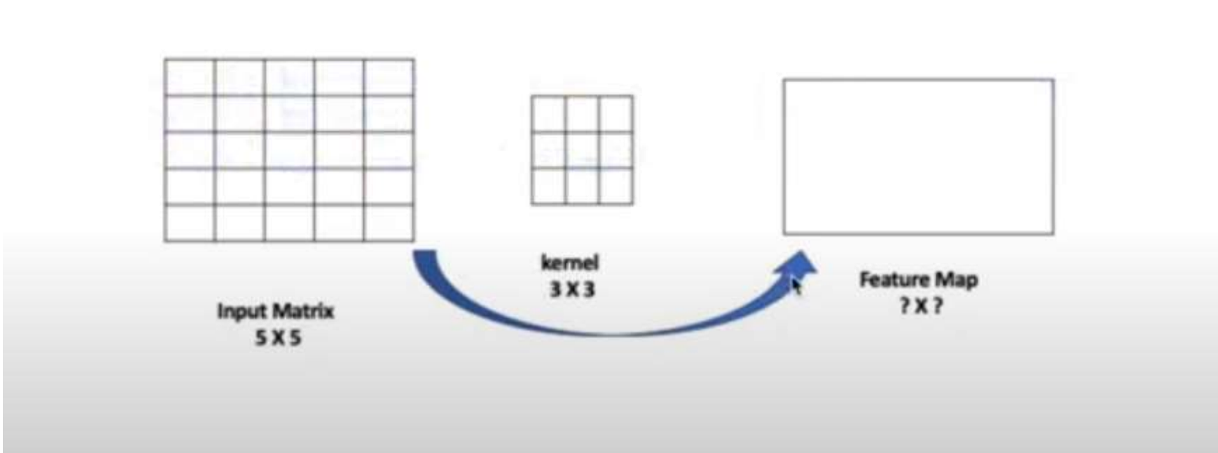
In [1]:
```
from numpy import asarray
from keras.models import Sequential
from keras.layers import Conv2D
# define input data
data = [[3,3,2,1,0],
        [0, 0, 1, 3, 1],
        [3, 1, 2, 2, 3],
        [2, 0, 0, 2, 2],
        [2, 0, 0, 0, 1]]
data = asarray(data)
data = data.reshape(1, 5, 5, 1)
kernel = [[[[0]],[[1]],[[2]]],
            [[[2]],[[2]],[[0]]],
            [[[0]],[[1]],[[2]]]]
weights = [asarray(kernel), asarray([0.0])]
```

```
E:\anaconda install\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufun
c size changed, may indicate binary incompatibility. Expected 192 from C heade
r, got 216 from PyObject
  return f(*args, **kwds)
E:\anaconda install\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufun
c size changed, may indicate binary incompatibility. Expected 192 from C heade
r, got 216 from PyObject
  return f(*args, **kwds)
E:\anaconda install\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufun
c size changed, may indicate binary incompatibility. Expected 192 from C heade
r, got 216 from PyObject
  return f(*args, **kwds)
E:\anaconda install\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufun
c size changed, may indicate binary incompatibility. Expected 192 from C heade
r, got 216 from PyObject
  return f(*args, **kwds)
```

In [4]:
```
from IPython.display import Image
Image(filename="kernel.png")
```

Out[4]:



In [7]:
```
from keras.models import Sequential
from keras.layers import Conv2D

model=Sequential()
model.add(Conv2D(1,(3,3),input_shape=(5,5,1)))
model.summary()
```

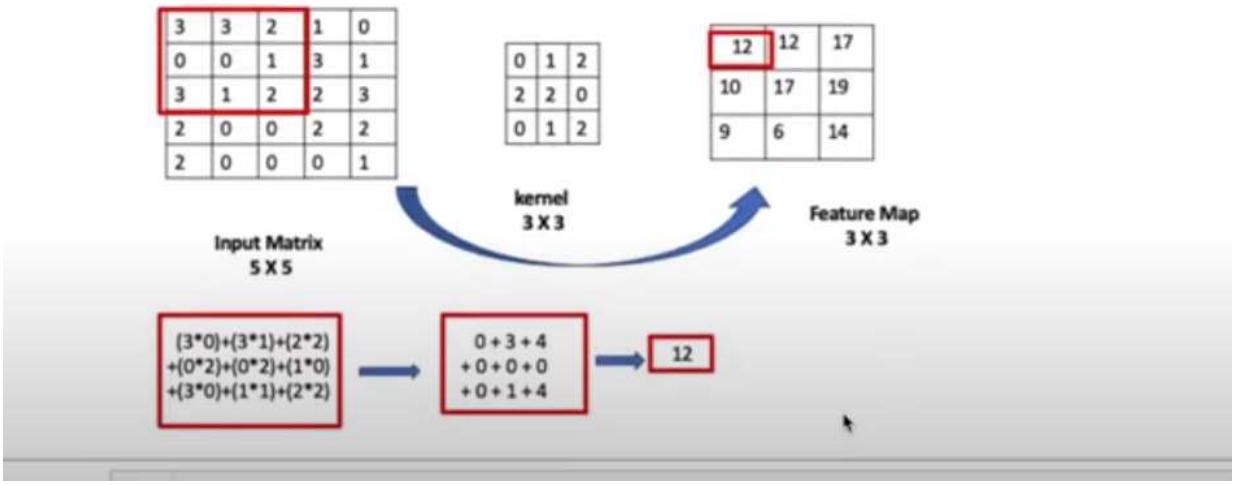
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 3, 3, 1)           10
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

In [8]:
```
model.set_weights(weights)
yhat=model.predict(data)
for r in range(yhat.shape[1]):
    print([yhat[0,r,c,0] for c in range(yhat.shape[2])])
```

```
[12.0, 12.0, 17.0]
[10.0, 17.0, 19.0]
[9.0, 6.0, 14.0]
```

In [9]:
```
from IPython.display import Image
Image(filename="kernel2.png")
```

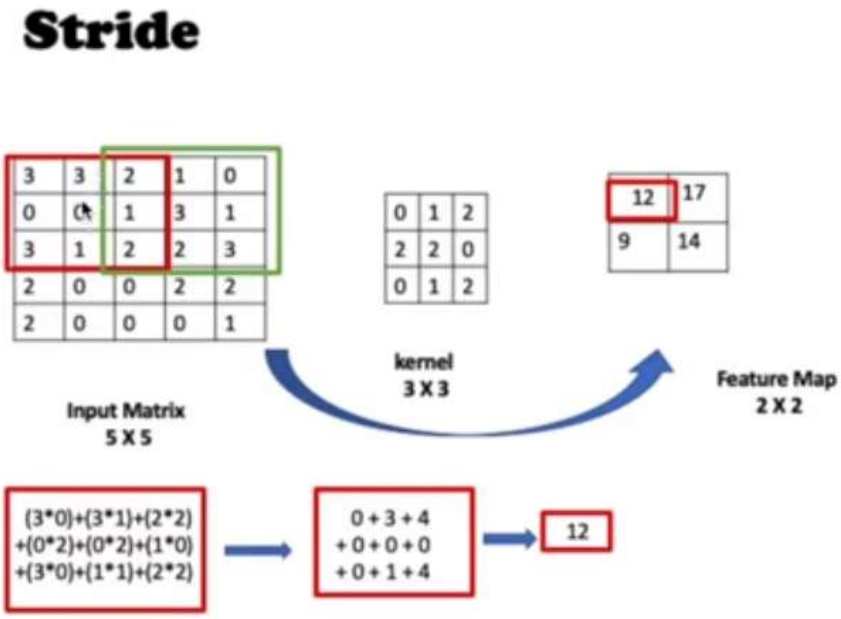
Out[9]:



In [11]:
```
#strides
# [(n-k)/s]+1
```

In [12]:
```
from IPython.display import Image
Image(filename="stride.png")
```

Out[12]:



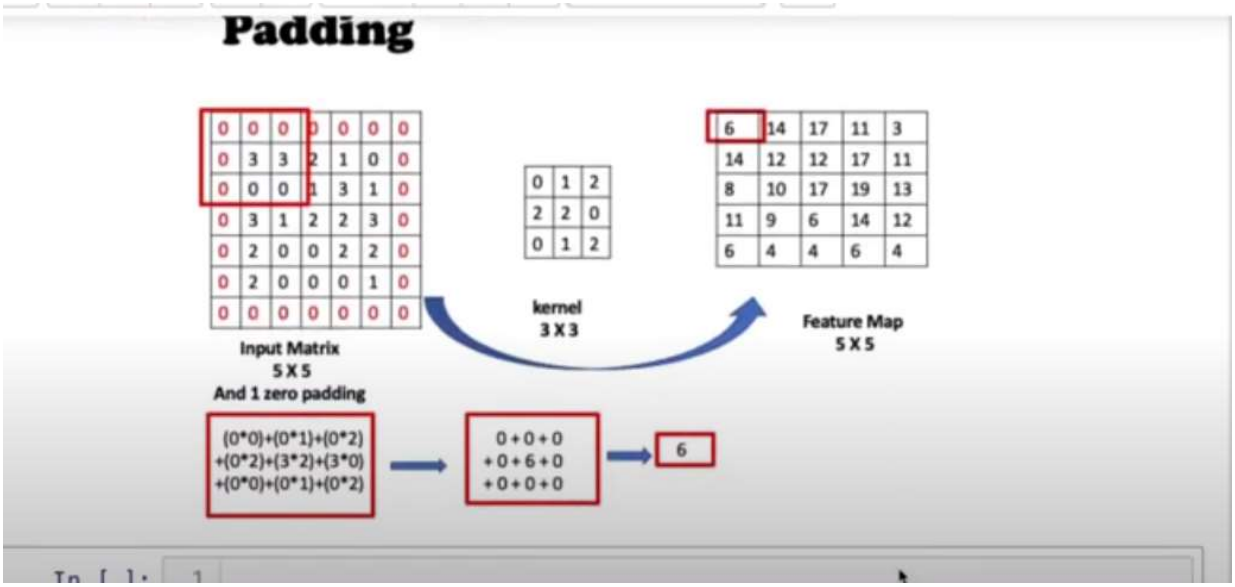
In [13]:
```
model=Sequential()
model.add(Conv2D(1,(3,3),strides=(2,2),input_shape=(5,5,1)))
model.summary()
model.set_weights(weights)
yhat=model.predict(data)
for r in range(yhat.shape[1]):
    print([yhat[0,r,c,0] for c in range(yhat.shape[2])])
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 2, 2, 1)           10
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
[12.0, 17.0]
[9.0, 14.0]
```

In [14]:
```
#padding avoid the loss information
#output matrix=[(n-k+2p)/s]+1
```

In [15]:
```
from IPython.display import Image
Image(filename="padding.png")
```

Out[15]:



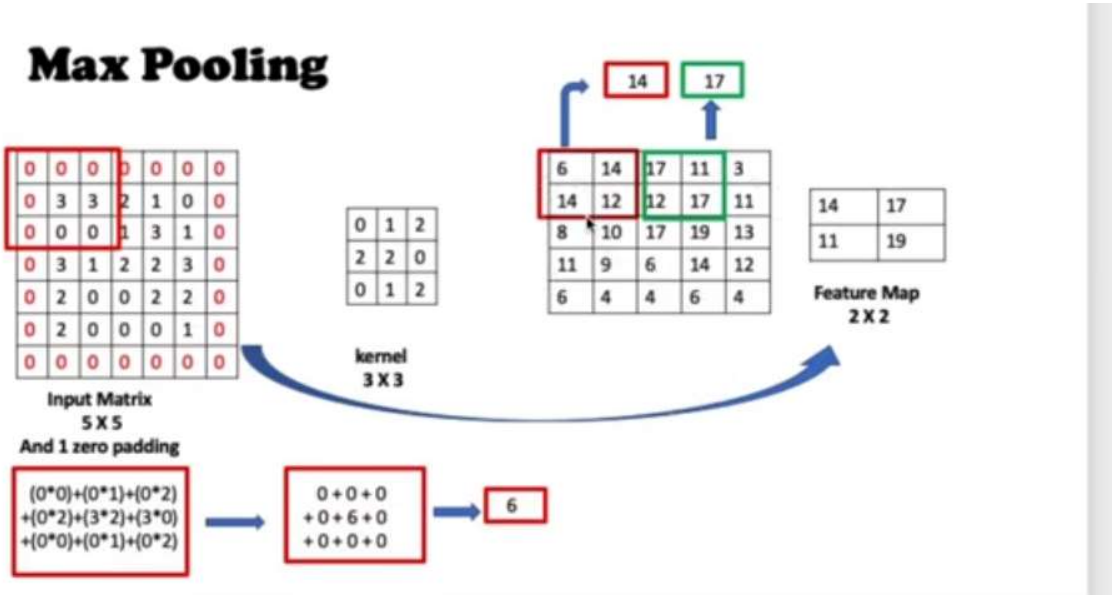
In [16]:
```
model=Sequential()
model.add(Conv2D(1,(3,3),padding="same",input_shape=(5,5,1)))
model.summary()
model.set_weights(weights)
yhat=model.predict(data)
for r in range(yhat.shape[1]):
    print([yhat[0,r,c,0] for c in range(yhat.shape[2])])
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 5, 5, 1)           10
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
[6.0, 14.0, 17.0, 11.0, 3.0]
[14.0, 12.0, 12.0, 17.0, 11.0]
[8.0, 10.0, 17.0, 19.0, 13.0]
[11.0, 9.0, 6.0, 14.0, 12.0]
[6.0, 4.0, 4.0, 6.0, 4.0]
```

maxpooling

In [19]:
```
from IPython.display import Image
Image(filename="maxpooling.png")
```

Out[19]:



In [26]:
```
from keras.layers import MaxPooling2D
model=Sequential()
model.add(Conv2D(1,(3,3),padding="same",input_shape=(5,5,1)))
model.add(MaxPooling2D(2,2))
model.summary()
model.set_weights(weights)
yhat=model.predict(data)
for r in range(yhat.shape[1]):
    print([yhat[0,r,c,0] for c in range(yhat.shape[2])])
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 5, 5, 1)           10
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2, 2, 1)           0
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_fu
nction.<locals>.predict_function at 0x0000022B45C66708> triggered tf.function r
etracing. Tracing is expensive and the excessive number of tracings could be du
e to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with d
ifferent shapes, (3) passing Python objects instead of tensors. For (1), please
define your @tf.function outside of the loop. For (2), @tf.function has experim
ental_relax_shapes=True option that relaxes argument shapes that can avoid unne
cessary retracing. For (3), please refer to https://www.tensorflow.org/tutorial
s/customization/performance#python_or_tensor_args (https://www.tensorflow.org/t
utorials/customization/performance#python_or_tensor_args) and https://www.tenso
rflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/pyth
on/tf/function) for  more details.
[14.0, 17.0]
[11.0, 19.0]
```

In [27]:
```
yhat
```

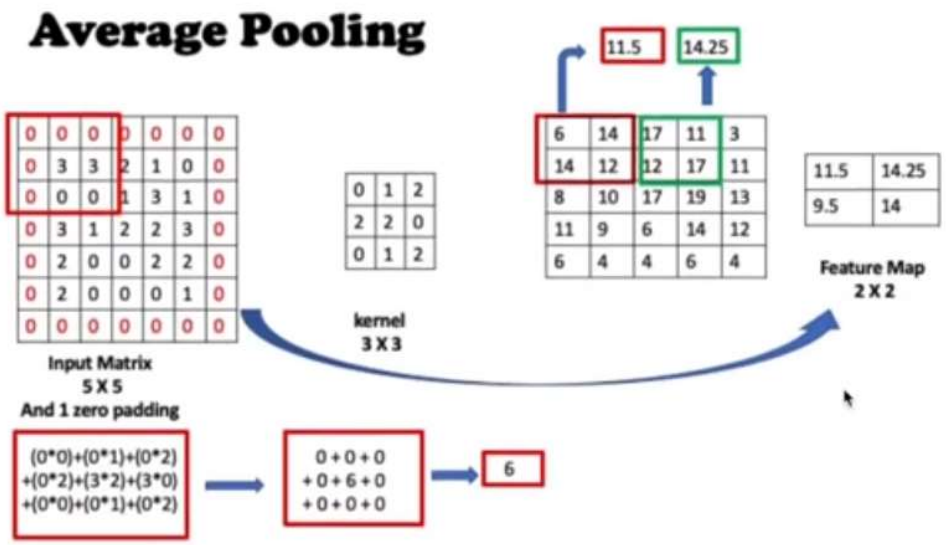
Out[27]:
```
array([[[[14.],
         [17.]],

        [[11.],
         [19.]]]], dtype=float32)
```

average polling

In [23]:
```
Image(filename="averagepooling.png")
```

Out[23]:



In [ ]:
```
1
```

In [24]:
```
from keras.layers import AveragePooling2D
model=Sequential()
model.add(Conv2D(1,(3,3),padding="same",input_shape=(5,5,1)))
model.add(AveragePooling2D(2,2))
model.summary()
model.set_weights(weights)
yhat=model.predict(data)
for r in range(yhat.shape[1]):
    print([yhat[0,r,c,0] for c in range(yhat.shape[2])])
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 5, 5, 1)           10
_________________________________________________________________
average_pooling2d (AveragePo (None, 2, 2, 1)           0
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000022B454DB0D8> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and https://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
[11.5, 14.25]
[9.5, 14.0]
```

In [25]:
```
from keras.layers import Flatten
model=Sequential()
model.add(Conv2D(1,(3,3),padding="same",input_shape=(5,5,1)))
model.add(AveragePooling2D(2,2))
model.add(Flatten())
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 5, 5, 1)           10
_________________________________________________________________
average_pooling2d_1 (Average (None, 2, 2, 1)           0
_________________________________________________________________
flatten (Flatten)            (None, 4)                 0
=================================================================
Total params: 10
Trainable params: 10
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
```