In [1]:
```
import torch
from torch import nn
```

In [2]:
```
def comp_conv2d(conv2d, X):
    X = X.reshape((1, 1) + X.shape)
    Y = conv2d(X)
    return Y.reshape(Y.shape[2:])

conv2d = nn.LazyConv2d(1, kernel_size=3, padding=1)
X = torch.rand(size=(8, 8))
comp_conv2d(conv2d, X).shape
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/torch/nn/modules/lazy.py:181: UserWarning: Lazy modules are a new feature under heavy development so changes to the API or functionality can happen at any moment.
  warnings.warn('Lazy modules are a new feature under heavy development '
```

Out[2]:
```
torch.Size([8, 8])
```

In [3]:
```
conv2d = nn.LazyConv2d(1, kernel_size=(5, 3), padding=(2, 1))
comp_conv2d(conv2d, X).shape
```

Out[3]:
```
torch.Size([8, 8])
```

In [4]:
```
conv2d = nn.LazyConv2d(1, kernel_size=3, padding=1, stride=2)
comp_conv2d(conv2d, X).shape
```

Out[4]:
```
torch.Size([4, 4])
```

In [5]:
```
conv2d = nn.LazyConv2d(1, kernel_size=(3, 5), padding=(0, 1), stride=(3, 4))
comp_conv2d(conv2d, X).shape
```

Out[5]:
```
torch.Size([2, 2])
```

Discussion: 이 단원에서는 Padding과 Stride의 개념에 대해 설명하고 있다. Padding은 이미지의 경계 픽셀이 잘 사용되지 않는 문제를 해결하기 위해 사용되는 것으로, CNN에서 연속적인 합성곱 연산을 거칠 때 출력 이미지가 점점 작아지면서 경계 근처의 중요한 정보가 손실되는 것을 막는다. 이미지의 가장자리에 0으로 된 픽셀을 추가하는 것이다. Stride는 커널이 이미지 위를 이동하는 간격을 의미한다. 수업 시간에 이것으로 간단한 수식을 계산했던 것으로 어렴풋이 기억에 남는데, 출력 크기를 줄이거나 계산 속도를 향상시키고자 할때 유용하다. 출력 크기 = [입력 - 커널 + (2x패딩)/스트리이드] + 1로, 이때 분수에서의 소숫점은 버리는 것으로 알고 있다.

Exercise: Given the final code example in this section with kernel size (3,5), padding (0,1), and stride (3,4), calculate the output shape to check if it is consistent with the experimental result.

입력은 8x8인 것을 아므로, 한번 공식을 대입해보면.

출력 높이 = (8-3/3)(반내림) + 1 = 1+1 = 2, 출력 너비 (8-5+2/4) + 1 = 1+ 1 = 2, 따라서 2x2로 일치한다. 사실 뭐 컴퓨터가 틀릴 일은 없으니 당연한 결과이긴 하다.