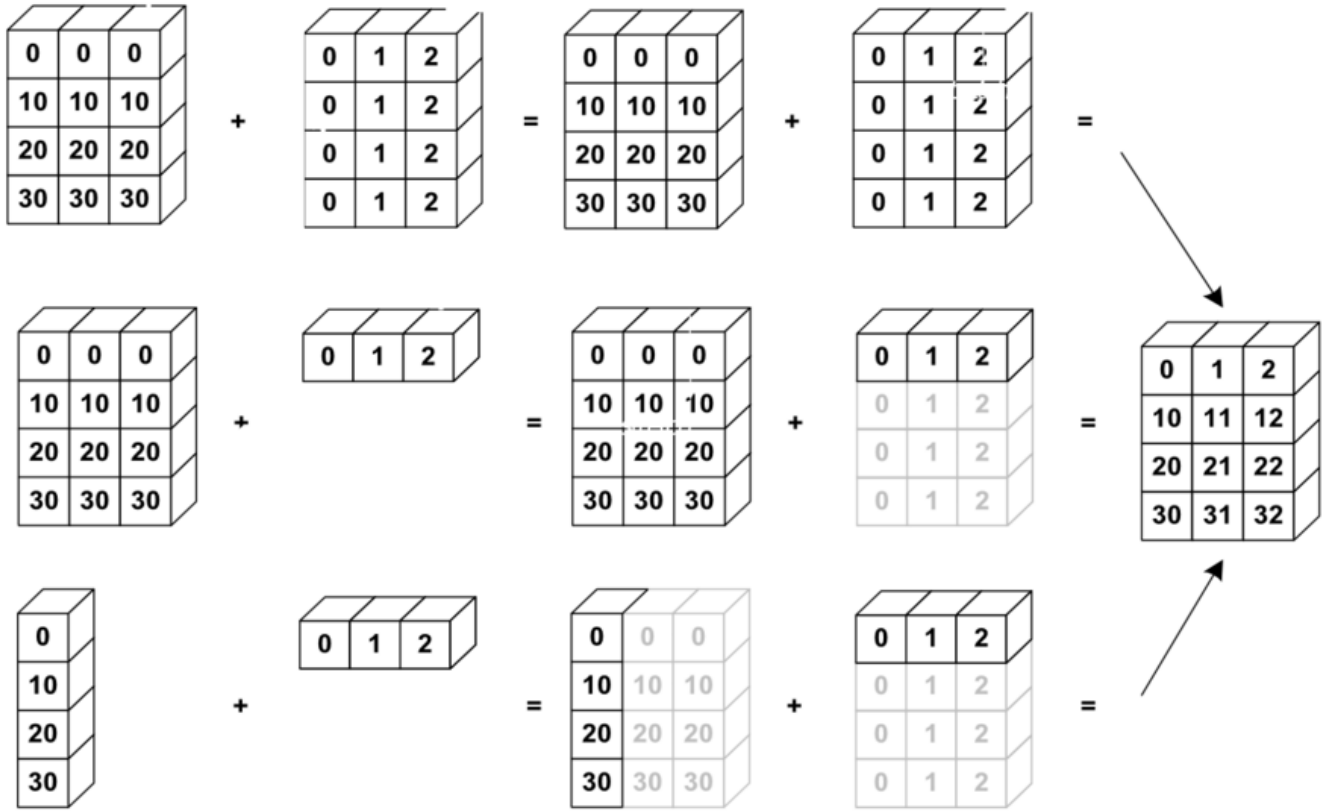
NumPy 2

Broadcasting

- 브로드캐스팅
  - 같은 크기의 배열을 가지도록 변형하는 것.



Broadcasting

```
data = np.array([[0,10,20,30]]).T
data+[0,1,2,3]
```

```
a = np.tile(np.arange(0, 40, 10), (3, 1)).T
b= np.array([0,1,2])
data=a+b
print(data)
```

*np.tile(A, reps) : A를 reps만큼 반복함

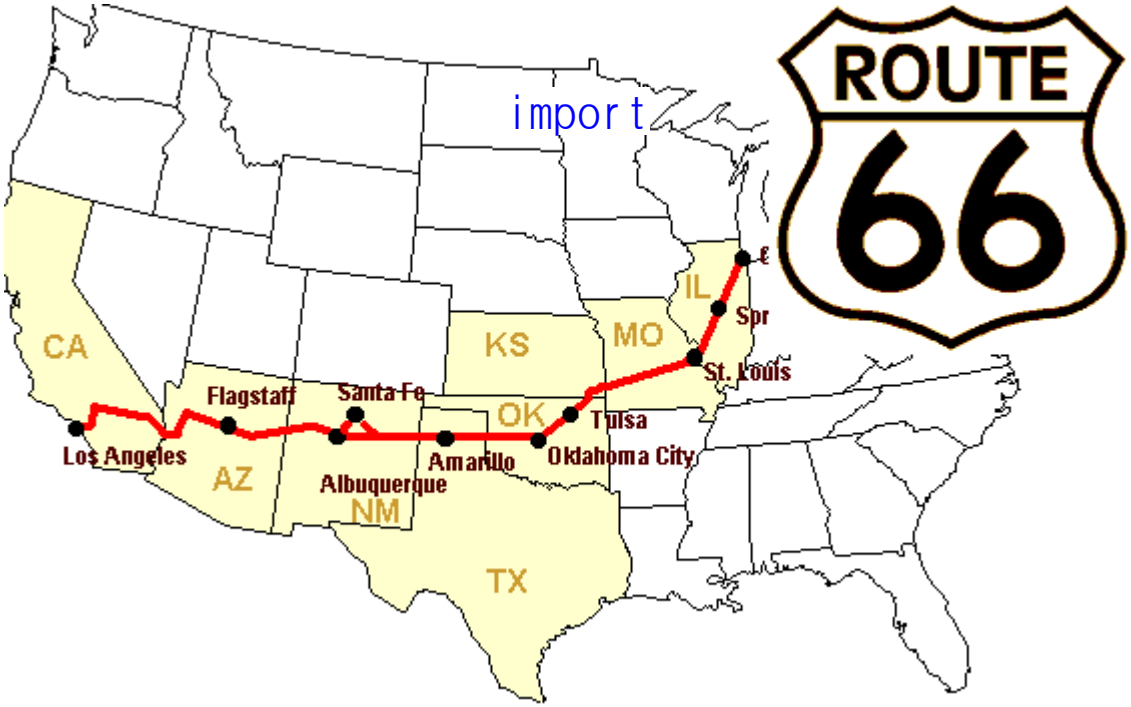
```
a = np.ones((4,5))
a[0]=2
print(a)
```

```
a = np.arange(0, 40, 10)
print(a.shape)
 # adds a new axis -> 2D array
b = a[:, np.newaxis]
print(b.shape)
```

Broadcasting 예

- City 사이의 거리 구하기

```
mileposts = np.array([0, 198, 303, 736, 871, 1175, 1475, 1544, 1913, 2448])
distance_array = np.abs(mileposts - mileposts[:, np.newaxis])
```



Broadcasting 예

- 그리드 또는 네트워크 기반 거리 계산

```
x,y = np.arange(5), np.arange(5)[:, np.newaxis]
distance = np.sqrt(x**2 + y**2)
print(distance)

plt.pcolor(distance)
plt.colorbar()
plt.show()
```

- x, y = np.**ogrid**[0:5, 0:5] # grid에 대한 계산을 다루기 유용
  - x : (5,1), y : (1,5) shape을 가짐
- x, y = np.**mgrid**[0:5, 0:5]
  - x,y : (5,5) shape을 가짐

Array Shape Manipulation

- Flattening

```
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.ravel()) #[1,2,3,4,5,6]
print(a.T)
print(a.T.ravel())
```

- Reshaping

```
a.shape
b = a.ravel()
b = b.reshape((2, 3))
```

Array Shape Manipulation

- Flattening

```
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.ravel()) #[1,2,3,4,5,6]
print(a.T)
print(a.T.ravel())
```

- Reshaping

```
a.shape
b = a.ravel()
b = b.reshape((2, 3))
b.reshape((2, -1)) # unspecified (-1) value is inferred
```

Array Shape Manipulation

- Adding a dimension

```
z = np.array([1, 2, 3])
print(z[:, np.newaxis])
print(z[np.newaxis, :])
```

- Resizing

```
a = np.arange(4)
a.resize((8,))
```

  – 다른 변수에의해 참조 되지 않아야 변경가능 함.
  (b=a (x))

Array Shape Manipulation

- Sorting data

```
a = np.array([[4, 3, 5], [1, 2, 1]])
b = np.sort(a, axis=1) #Sorts each row separately!
a.sort(axis=1)
```

```
a = np.array([4, 3, 1, 2])
j = np.argsort(a) # sorting with index

j_max = np.argmax(a) # finding maxima
j_min = np.argmin(a) # finding minima
```

Advanced operations

- Polynomials

$$3x^2 + 2x - 1$$

```
p = np.poly1d([3, 2, -1])
print(p(0))
print(p.roots)
print(p.order)
p = np.polynomial.Polynomial([-1, 2, 3])
```

Advanced operations

- Polynomials

```
x = np.linspace(0, 1, 20)
y = np.cos(x) + 0.3*np.random.rand(20)
p = np.poly1d(np.polyfit(x, y, 3))
t = np.linspace(0, 1, 200)
plt.plot(x, y, 'o', t, p(t), '-')
plt.show()
```

```
x = np.linspace(-1, 1, 2000)
y = np.cos(x) + 0.3*np.random.rand(2000)
p = np.polynomial.Chebyshev.fit(x, y, 90)
t = np.linspace(-1, 1, 200)
plt.plot(x, y, 'r.')
plt.plot(t, p(t), 'k-', lw=3)
plt.show()
```

Advanced operations

- Loading data files

```
data = np.loadtxt('populations.txt')
np.savetxt('pop2.txt', data)
data2 = np.loadtxt('pop2.txt')
```

- Images

```
img = plt.imread('images.png')
print(img.shape, img.dtype)
plt.imshow(img)
plt.show()
```