numpy는 np라는 이름으로 임포트하는 것

```
import numpy as np
```

numpy에서는 array라는 배열로 계산을 수행 여기에서는 [1, 2, 3, 4, 5]라는 요소를 가진 numpy의 배열 array를 준비

```
arr = np.array([1, 2, 3, 4, 5])
```

크기는 shape로 알 수 있음

```
arr.shape
```

```
(5,)
```

array에 대한 덧셈 연산

```
arr + 2
```

```
array([3, 4, 5, 6, 7])
```

모든 요소에 2를 더했음

덧셈 연산만이 아닌, 다른 연산자도 동일하게 방식으로 수행
제곱근

```
np.sqrt(arr)
```

```
array([1.        , 1.41421356, 1.73205081, 2.        , 2.23606798])
```

리스트와 마찬가지로 슬라이스로 요소를 추출할 수 있음

```
arr[:2]
```

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

```
arr[3:5]
```

```
array([4, 5])
```

numpy의 arange 함수가 자주 사용 arange 함수를 사용하여 [0, 1, 2, 3, 4]라는 요소를 가진 array를 간단히 작성할 수 있음

```
np.arange(5)
```

```
array([0, 1, 2, 3, 4])
```

arange 함수에는 start, end, step을 지정할 수 있음 [2, 4, 6, 8]이라고 2씩 숫자가 증가하는 array는 다음과 같이 작성할 수 있음

```
np.arange(2, 10, 2)
```

```
array([2, 4, 6, 8])
```

linspace 함수도 자주 사용
linspace 함수도 start부터 end까지의 array를 생성하는 함수이지만, step이 아닌 요소의 수를 지정 여기에서는 0부터 3까지 11 분할한 array를 작성

```
np.linspace(0, 3, 11)
```

```
array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ])
```

2차원 array도 간단히 생성

```
arr2 = np.array([[1, 3],
                 [5, 7]])
arr2
```

```
array([[1, 3],
       [5, 7]])
```

```
arr2.shape
```

```
(2, 2)
```

save 함수로 array를 저장할 수 있음 첫 번째 인수는 저장되는 이름이고, 두 번째 인수가 저장되는 array 임

```
np.save('test.npy', arr2)
```

읽어들이는 것은 load 함수로 가능

```
np.load('test.npy')
```

```
array([[1, 3],
       [5, 7]])
```