NumPy (2)

```
import  numpy as np
```

[NumPy 배열 (array) 요소의 타입 확인]

```
x = np.array([1, 2, 3])
print( type(x))   # NumPy 배열  의 타입 확인
print( x.dtype )  # NumPy 배열 (array) 요소의 타입 확인
```

실행결과

```
<class 'numpy.ndarray'>
int32
```

[NumPy 배열 (array) 요소의 타입 지정]

```
"""
dtype 접두사   사용 예

b 불리언 b (참 혹은 거짓)
i 정수
u 부호 없는 정수
f 부동소수점
c 복소 부동소수점
O 객체
S 바이트 문자열
U 유니코드 문자열
"""
x = np.array([1, 2, 3], dtype='f')


"""
  무한대를 표현  np.inf(infinity)
  정의할 수 없는 np.nan(not a number)
"""
```

[NumPy 배열 초기화]

```
""" 0로 초기화된 배열  """

a = np.zeros(5)
b = np.zeros((2, 3))
c = np.zeros((5, 2), dtype="i")

print( a)
print( b)
print( c)
```

실행결과

```
[0. 0. 0. 0. 0.]
[[0. 0. 0.]
 [0. 0. 0.]]
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]
```

```
"""문자열 배열도 가능. 단  문자열 크기 지정필요
만약 더 큰 크기의 문자열을 할당하면 잘릴 수 있다."""
d = np.zeros(5, dtype="U4") #unicode 문자 4글자

d[0] = "abc"
d[1] = "abcd"
d[2] = "ABCDE"      #unicode 문자 4글자까지만 저장됨
d[3] = "ABCDEF"     #unicode 문자 4글자까지만 저장됨
d[4] = "ABCDEFG"    #unicode 문자 4글자까지만 저장됨
print(d)
```

실행결과

```
['abc' 'abcd' 'ABCD' 'ABCD' 'ABCD']
```

```
""" 1로 초기화된 배열  """

a = np.ones(5)
b = np.ones((2, 3))
c = np.ones((5, 2), dtype="i")
print(a)
```

실행결과

```
[1. 1. 1. 1. 1.]
```

```
"""arange 명령은 NumPy 버전의 range 명령.  특정한 규칙에 따라 증가하는 수열
"""
a = np.arange(10)  # 0 .. n-1     시작, 끝(포함하지 않음)
print(a)

a =np.arange(3, 21, 2)  # 시작, 끝(포함하지 않음), 단계
print(a)
```

실행결과

```
[0 1 2 3 4 5 6 7 8 9]
[ 3  5  7  9 11 13 15 17 19]
```

```
""" inspace 명령이나  선형 구간  을 지정한 구간의 수만큼 분할한다 """

n = np.linspace(0, 100, 5)  # 시작, 끝(포함), 갯수
print(n)
```

실행결과

```
[  0.   25.   50.   75. 100.]
```

```
"""배열의 내부 데이터는 보존한 채 형태만 바꾸려면 reshape 메서드"""

"""예)12 개의 원소를 가진 1 차원 행렬은 3x4 형태의 2 차원 행렬로 """
a = np.arange(12)  # 0 .. n-1
print(a)
b = a.reshape(3, 4)
print(b)
```

실행결과

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
""" reshape 명령의 매개인자 중 하나는 -1 이라는 숫자로 대체할 수 있음. -1: 해당 숫자는 다른 값에서
계산되어 사용"""
a = np.arange(12)
print( a.reshape(3, -1)  )      #3 행 * ? 열
print (a.reshape(2, 2, -1) )    #2 깊이 * 2 행 * ? 열
print (a.reshape(2, -1, 2) )    #2 깊이 * ? 행 * 2 열
```

실행결과

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

[[[ 0  1  2]
  [ 3  4  5]]
 [[ 6  7  8]
  [ 9 10 11]]]
```

```
[[[ 0  1]
  [ 2  3]
  [ 4  5]]
 [[ 6  7]
  [ 8  9]
  [10 11]]]
```

```
x = np.arange(5)
print( x )
print( x.reshape(1, 5) )
print( x.reshape(5, 1))
```

실행결과

```
 [0 1 2 3 4]
 [[0 1 2 3 4]]
 [[0]
  [1]
  [2]
  [3]
  [4]]
```

```
""" 다차원 배열을 무조건 1 차원으로 펼치기 위해서는 flatten 나 ravel 메서드  """
print (a.flatten() )
print (a.ravel() )
```

실행결과

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

```
"""
배열 연결

행의 수나 열의 수가 같은 두 개 이상의 배열을 연결하여(concatenate)
더 큰 배열을 만들 때는 다음과 같은 명령을 사용한다.
"""
a1 = np.ones((2, 3))
a2 = np.zeros((2,  3))
print(a1)
print(a2)
print("---------------")

print( np.hstack([a1,a2])) # 세로로 연결
print( np.vstack([a1,a2])) # 가로로 연결
```

실행결과

```
[[1. 1. 1.]
 [1. 1. 1.]]
[[0. 0. 0.]
 [0. 0. 0.]]
---------------

[[1. 1. 1. 0. 0. 0.]
 [1. 1. 1. 0. 0. 0.]]
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [0. 0. 0.]
 [0. 0. 0.]]
```