연습문제

11.1 자신의 컴퓨터에 넘파이을 설치하자. 그리고 넘파이의 버전을 다음과 같이 출력해 보자. 밑줄 친 부분에 출력된 내용을 적으시오.

```
>>> import numpy as np
>>> np.__version__
_____
```

11.2 크기가 10인 1차원 배열을 생성하여라. 이 배열의 모든 원소는 0 값을 가진다.

(1) 0을 10번 입력하여 이 배열 a를 생성하여라. 그리고 이 배열 a를 출력하여라.

실행결과

```
a = [0 0 0 0 0 0 0 0 0 0]
```

(2) np.zeros() 함수를 사용하여 배열 b를 생성하여라. 이 배열을 b를 출력하여라.

실행결과

```
b = [0 0 0 0 0 0 0 0 0 0]
```

(3) 배열 a와 배열 b의 형을 출력하고 그 결과를 밑줄 친 부분에 적으시오.

```
>>> type(a)
_____
>>> type(b)
_____
```

11.3 1에서 10까지 값을 가지는 크기가 10인 1차원 배열 n_arr을 생성하여라. 이 배열이 메모리에서 차지하는 크기는 얼마인가? 다음의 밑줄 친 부분에 알맞은 코드를 입력하여 다음과 같이 바이트 단위로 출력하여라(힌트: itemsize와 size 속성 값을 곱한다).

```
>>> n_arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], dtype = 'int32')
>>> _____
n_arr의 크기 : 40 bytes
```

(1) dtype = 'int32' 대신 dtype = 'int64'로 고칠 경우의 n_arr의 크기는 얼마인가?(바이트 단위로 출력하여라)

(2) np.array([1.5, 2.2, 3.2, 4.9], dtype = 'float64'] 배열의 크기는 얼마인가?(바이트 단위로 출력하여라)

11.4 1에서 10까지의 정수 값을 가지는 크기가 10인 1차원 배열 a를 arange() 함수를 사용하여 생성하여 출력하여라.

(1) 다음과 같이 출력하여라.

실행결과

```
a = [ 1 2 3 4 5 6 7 8 9 10]
```

(2) 다음과 같이 역순으로 출력하여라.

실행결과

```
a = [10 9 8 7 6 5 4 3 2 1]
```

(3) (2)의 결과에 reshape() 함수를 사용하여, 배열 a를 (2,5) 형태의 다음과 같은 행렬로 변경하여 출력하여라.

실행결과

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

(4) (3)의 결과에 reshape() 함수를 사용하여, 배열 a를 (5, 2) 형태의 다음과 같은 행렬로 변경하여 출력하여라.

실행결과

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

11.5 1에서 16까지의 정수값을 가지는 4x4 크기의 배열을 만들어서 다음과 같이 출력하시오. 이 때 arange()와 reshape()을 이용하여 배열을 생성하시오.

실행결과

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

11.6 random.random() 함수를 사용하여 10개의 난수값을 가진 배열 a를 생성하여 다음과 같이 출력하시오. 그리고 이 수들 중에서 최댓값, 최솟값, 평균값을 각각 출력하시오.

실행결과

```
a =  [0.79457055 0.79563394 0.61542215 0.881637   0.36674677 0.86430664
0.43726605 0.27081378 0.26353765 0.39242491]
최댓값 : 0.8816369959416799
최솟값 : 0.2635376504756979
평균값 : 0.5682359433240208
```

11.7 사용자로부터 2이상의 수 n을 입력으로 받아서, 입력된 수를 바탕으로 다음과 같은 nxn 크기의 다차원 배열 a를 생성하는 프로그램을 작성하시오. 이때 배열의 내용은 0과 1의 값이 체크 판 패턴으로 교차하여 나타나도록 하여라.

실행결과

```
n을 입력하시오 : 4
[[1 0 1 0]
 [0 1 0 1]
 [1 0 1 0]
 [0 1 0 1]]
```

실행결과

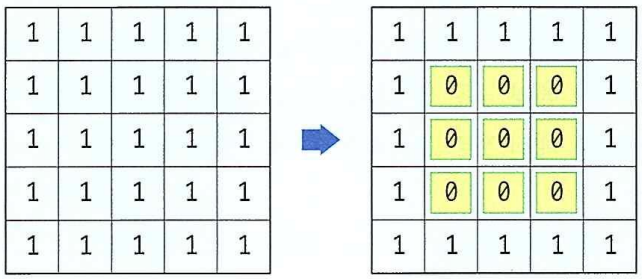
```
n을 입력하시오 : 5
[[1 0 1 0 1]
 [0 1 0 1 0]
 [1 0 1 0 1]
 [0 1 0 1 0]
 [1 0 1 0 1]]
```

11.8 사용자로부터 2 이상의 수 n을 입력으로 받아서, 입력된 수를 바탕으로 다음과 같은 대각선 성분의 값이 1에서 n까지 증가하는 다차원 배열 a를 생성하는 프로그램을 작성하시오. 이때 대각선 성분 이외의 값은 모두 0으로 하시오.

실행결과

```
n을 입력하시오 : 6
[[1 0 0 0 0 0]
 [0 2 0 0 0 0]
 [0 0 3 0 0 0]
 [0 0 0 4 0 0]
 [0 0 0 0 5 0]
 [0 0 0 0 0 6]]
```

11.9 사용자로부터 2이상의 수 n을 입력으로 받아서 모든 원소의 값을 1로 가지는 (n, n) shape 행렬 a를 만드시오. 그리고 아래 그림과 같이 이 행렬 a의 경계 값을 제외한 내부의 값을 모두 0으로 가지는 행렬 b를 만드시오(힌트: np.ones((n, n), dtype = 'int32')로 모든 원소를 1로 가지는 행렬을 생성한 후 [1:-1, 1:-1] 인덱스를 사용하시오).



실행결과

```
n을 입력하시오 : 5
a 행렬
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
b 행렬
[[1 1 1 1 1]
 [1 0 0 0 1]
```

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

11.10 11.9 문제와 달리 a 행렬로부터 경계 값이 모두 0이고 내부가 1인 행렬 c를 다음과 같이 생성하시오. 이때 배열의 슬라이싱을 사용하시오.

실행결과

```
n을 입력하시오 : 5
a 행렬
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
c 행렬
[[0 0 0 0 0]
 [0 1 1 1 0]
 [0 1 1 1 0]
 [0 1 1 1 0]
 [0 0 0 0 0]]
```

11.11 다음과 같은 두 배열 a와 b가 있을 경우 a 배열의 원소 내에 b 배열의 원소가 있는지 없는지를 판단하여 출력하는 프로그램을 작성하시오. 이때 a 배열내의 원소가 b에 있을 경우는 True, 없을 경우는 False를 출력하여라.

실행결과

```
a 배열 : [ 0 10 20 40 60 80]
b 배열 : [ 0 20]
[ True False True False False False]
```

11.12 임의의 shape으로 이루어진 행렬 x에 대하여 각 성분의 출현 횟수를 구하여 다음과 같이 출력하는 프로그램을 작성하시오.

실행결과

```
x : [[10 20 40 60] [10 20 40 40]]
10 : 2번
20 : 2번
40 : 3번
60 : 1번
```

실행결과

```
x : [[ 80 120 40] [ 60 80 120] [ 40 40 40]]
40 : 4번
60 : 1번
80 : 2번
120 : 2번
```

11.13 두 배열 a, b가 다음과 같은 원소를 가질 때 , 두 배열의 뺄셈을 수행하여 아래와 같이 출력하시오.

실행결과

```
a 배열 : [ 0 10 20 40 60 80]
b 배열 : [ 0 20]
a - b : [ 10 40 60 80]
```

(1) 집합 연산을 사용하지 말고, for 문과 numpy.append() 메소드를 사용하여 이 문제를 해결하시오.

(2) numpy.setdiff1d(a, b) 함수를 사용하여 이 문제를 해결하시오.

11.14 다음과 같이 사용자로부터 정수 n을 입력으로 받아서 (n, n) shape의 정방행렬을 만들고 이 행렬의 대각선 성분의 값들이 모두 1로 만드시오(다음과 같이 X자 패턴이 되도록 할 것).

실행결과

```
n을 입력하시오 : 5
[[1 0 0 0 1]
 [0 1 0 1 0]
 [0 0 1 0 0]
 [0 1 0 1 0]
 [1 0 0 0 1]]
```

```
n을 입력하시오 : 6
[[1 0 0 0 0 1]
 [0 1 0 0 1 0]
 [0 0 1 1 0 0]
 [0 0 1 1 0 0]
 [0 1 0 0 1 0]
 [1 0 0 0 0 1]]
```

11.15 다음과 같이 사용자로부터 정수 n을 입력으로 받아서 (n, n) shape의 정방행렬을 만드시오.

(1) 이 행렬의 대각선 성분과 그 아래 값들이 다음과 같이 모두 1이 되도록 만드시오.

실행결과

```
n을 입력하시오 : 5
[[1 0 0 0 0]
 [1 1 0 0 0]
 [1 1 1 0 0]
 [1 1 1 1 0]
 [1 1 1 1 1]]
```

(2) 이때 모든 행렬의 원소의 합을 다음과 같이 구하시오.

실행결과

```
행렬의 모든 원소의 합: 15
```

(3) 이 행렬의 행 방향 성분의 합을 다음과 같이 구하시오(0축 방향).

실행결과

```
행렬의 행 방향 성분의 합 :
[5 4 3 2 1]
```

(4) 이 행렬의 열 방향 성분의 합을 다음과 같이 구하시오(1축 방향).

실행결과

```
행렬의 열 방향 원소의 합 :
[1 2 3 4 5]
```

11.16 다음과 같은 행렬 a가 있다. 이 행렬 a의 각 성분 값들을 인덱싱하고 concatenate() 함수를 이용하여 b와 같은 형태의 행렬로 만들어라.

실행결과

```
a = [[0 1 2] [3 4 5] [0 2 4] [6 8 10]]
b = [[0 1 2 0 2 4] [3 4 5 6 8 10]]
```

11.17 다음과 같은 5×5 크기의 2차원 배열 a가 있다. 이 배열을 슬라이싱하여 다음과 같은 배열을 생성하여라.

```
배열 a :
[[0 0 0 0 0]
 [0 1 1 1 0]
 [0 1 2 1 0]
 [0 1 1 1 0]
 [0 0 0 0 0]]
```

(1) 배열 b

실행결과

```
[[1 1 1]
 [1 2 1]
 [1 1 1]]
```

(2) 배열 c

실행결과

```
[[0 0 0]
 [0 1 1]
 [0 1 2]]
```

(3) 배열 d

실행결과

```
[[0 0 0]
 [1 1 1]
 [1 2 1]]
```

(4) 배열 e

실행결과

```
[[1 1 0]
 [2 1 0]
 [1 1 0]]
```

(5) 배열 f

실행결과

```
5) 배열 f :
[0 1 2 1 0]
```

11.18 다음과 같은 2차원 행렬 A, B가 있을 경우 아래 연산이나 행렬식 값을 각각 구하시오(단, AB는 행렬 곱 연산임).

```
행렬 A :
[[1 1 -1]
 [3 2 3]
 [6 7 8]]
```

```
행렬 B :
[[1 4 -6]
 [0 1 1]
 [3 -1 6]]
```

(1) 2A
(2) A+B
(3) A−B
(4) AB
(5) 3A−2B
(6) A의 행렬식
(7) B의 행렬식
(8) A 행렬의 행 성분의 합
(9) A 행렬의 열 성분의 합
(10) B 행렬의 행 성분의 합
(11) B 행렬의 열 성분의 합

11.19 0에서 23까지의 값을 순서대로 가지는 (4, 3, 2) 형태의 3차원 배열 a가 다음과 같이 있다. 이 배열에 대하여 10번째와 20번째 원소를 구하는 식을 만드시오.

```
a = np.arange(0, 24).reshape(4, 3, 2)
```

이 식을 이용하여 n과 배열 a를 매개변수로 받아서 원소를 반환하는 get_nth()라는 이름의 함수를 만들고, 10번째와 20번째 배열 원소를 다음과 같이 출력하시오.

실행결과

```
10번째 원소 : 9
20번째 원소 : 19
```

(힌트: 아래의 코드와 같이 a.shape[0], a.shape[1], a.shape[2]를 이용하시오)

```
>>> a.shape[0], a.shape[1], a.shape[2]
(4, 3, 2)
```

11.20 다음과 같은 관계식을 만족시키는 해 x, y, z를 linalg.solve() 함수를 이용하여 구하시오.

$$
\begin{aligned}
2x + y + z &= 16 \\
x + 2y + z &= 9 \\
x + y + 2z &= 3
\end{aligned}
$$

(1) 위 식의 해를 다음과 같이 출력하시오.

실행결과

```
x = 9.0, y = 2.0, z = -4.0
```

(2) 선형방정식에서 왼쪽 변의 방정식의 계수는 다음과 같은 행렬로 표기할 수 있다.

```
A = np.array([[2, 1, 1], [1, 2, 1], [1, 1, 2]])
```

이 행렬의 행렬식을 linalg.det() 함수를 사용하여 다음과 같이 구하여라.

실행결과

```
det(A) = 4.0
```