인공지능_ HW1

인공지능학부 215001 서가연

01. 다음 코드의 최종 결과를 쓰시오.

소스 코드 :

```python
## 넘파이 배열에 연산자를 적용하면 배열의 요소마다 연산자가 적용된다

import numpy as np
my_vector = np.array([1, 2, 3, 4, 5, 6])
selection = my_vector % 2 == 0
print(my_vector[selection])

```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/01.py
[2 4 6]
```

02. 다음 코드의 최종 결과를 쓰시오.

소스 코드:

```python
# 넘파이 배열끼리 연산을 할 때, 크기가 다르면 넘파이는
# 자동으로 배열의 크기를 확장 ==> 브로드캐스팅

import numpy as np

first_matrix = np.array([[1, 2, 3], [4, 5, 6]])
second_matrix = np.array([1, 2, 3])

print(first_matrix + second_matrix)
```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/02.py
[[2 4 6]
 [5 7 9]]
```

03. 크기가 10인 널 벡터를 생성하고 다섯 번째 요소는 1로 설정하는 코드를 작성하라

소스 코드 :

03.py > ...

```python
# np.zero(10) => 크기가 10인 널 벡터를 생성
# 다섯 번째 요소는 1로 설정하는 코드를 작성

import numpy as np
my_vector = np.zeros(10)
my_vector[4] = 1
print(my_vector)

```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/03.py
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

04. 10에서 19까지의 값을 가지는 1차원 배열을 생성하라

소스 코드:

04.py > ...

```python
## 10부터 19까지 가지는 1차원 배열 생성
## np.arange() 함수 사용

import numpy as np
my_vector = np.arange(10, 20)
print(my_vector)

```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/04.py
[10 11 12 13 14 15 16 17 18 19]
```

05. 0부터 9까지의 값으로 넘파이 1차원 배열을 채우고, 이를 거꾸로 하는 문장을 작성하라

소스 코드 :

```python
# 0부터 9까지 1차원 배열 생성
# 거꾸로 출력 => 슬라이싱 [::-1] 사용

import numpy as np
my_vector = np.arange(0, 10)
print(my_vector[::-1])
```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/05.py
[9 8 7 6 5 4 3 2 1 0]
```

06. 0부터 8까지의 값을 가지고 크기가 3x3인 행렬을 생성하라

소스 코드:

```python
## 0~8까지의 1차원 배열
## 3x3 행렬로 변환 => reshape 함수 이용

import numpy as np
my_vector = np.arange(0, 9)
my_matrix = my_vector.reshape(3, 3)
print(my_matrix)
```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/06.py
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

07. 난수로 채워진 3X3 넘파이 배열을 생성하라

소스 코드 :

```python
## 랜덤값으로 배열 생성하기 => np.random.rand

import numpy as np
np.random.seed(0)
my_matrix = np.random.rand(3,3)
print(my_matrix)
```

출력 결과 :

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/07.py
[[0.5488135  0.71518937 0.60276338]
 [0.54488318 0.4236548  0.64589411]
 [0.43758721 0.891773   0.96366276]]
```

08. 임의의 값으로 10X10 배열을 만들고, 최소값과 최대값을 찾아보자

소스 코드 :

```python
## 랜덤수로 10x10 행렬 생성 후 최소, 최대값 구하기
## max(), min() 함수 사용

import numpy as np

np.random.seed(0)
my_matrix = np.random.rand(10,10)
max_value = my_matrix.max()
min_value = my_matrix.min()
print(f'최소값 = {min_value} 최대값 = {max_value}')
```

출력 결과 :

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/08.py
최소값 = 0.004695476192547066 최대값 = 0.9883738380592262
```

09. 배열의 테두리에 1, 내부에 0을 가진 3x3 크기의 2차원 배열을 생성해보자.(슬라이싱 이용)

소스 코드 :

```python
# 배열의 테두리에 1, 내부에 0을 가진 2차원 배열 생성
## 슬라이싱을 사용하기

import numpy as np

my_matrix = np.ones((3, 3))
my_matrix[1:-1, 1:-1] = 0 # my_matrix[1, 1]과 동일한 결과
print(my_matrix)
```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/09.py
[[1. 1. 1.]
 [1. 0. 1.]
 [1. 1. 1.]]
```

10. 5x5 행렬을 만들어 체스 보드 패턴으로 채워보자.

소스 코드 :

```python
## 5x5 행렬 생성 => 체스 보드 패턴

import numpy as np

my_matrix = np.zeros((5, 5))
my_matrix[1::2, ::2] = 1
my_matrix[::2, 1::2] = 1
print(my_matrix)
```

출력 결과 :

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/10.py
[[0. 1. 0. 1. 0.]
 [1. 0. 1. 0. 1.]
 [0. 1. 0. 1. 0.]
 [1. 0. 1. 0. 1.]
 [0. 1. 0. 1. 0.]]
```

11. 3x3 난수로 행렬을 만들고 평균값과 표준 편차로 행렬을 정규화하여 보자.

소스 코드 :

11.py > ...

```python

## 랜덤값으로 3x3 행렬 생성
## 평균값과 표준편차 (x-mean)/std => 행렬 정규화

import numpy as np

np.random.seed(0)
my_matrix = np.random.rand(3, 3)
mean_value = my_matrix.mean()
std_value = my_matrix.std()
my_matrix = (my_matrix - mean_value)/std_value
print(my_matrix)
```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/11.py
[[-0.52562039  0.41707349 -0.21993774]
 [-0.54788979 -1.23477582  0.02444315]
 [-1.15583408  1.41760512  1.82493606]]
```

12. 넘파이를 사용하여 0에서 9까지의 값을 가진 벡터를 만들고 5에서 8 사이의 숫자 부호를 반전시켜보자

소스 코드 :

12.py > ...

```python
## 0~9까지의 벡터 생성
# 5-8 사이 숫자 부호를 반전 (-1 곱하기)

import numpy as np
my_vector = np.arange(0, 10)
# 5 6 7 8에만 -1을 곱하기
my_vector[5:9] *= -1
print(my_vector)

```

출력 결과 :

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/12.py
[ 0  1  2  3  4 -5 -6 -7 -8  9]
```

13. 넘파이로 3x3 크기의 2차원 배열을 생성하고, 모든 요소의 합, 각 열의 합, 각 행의 합을 계산해보자

소스 코드 :

13.py > ...

```python
## 3x3 크기의 2차원 배열 생성 => np.arange().reshape() 사용
## 모든 요소의 합, 각 열과 행의 합 계산 => sum() 사용

import numpy as np
my_matrix = np.arange(9).reshape(3, 3)
print(f'원본 배열: \n {my_matrix}')
print(f'모든 요소의 합: {my_matrix.sum()}')
print(f'각 행의 합: {my_matrix.sum(axis=0)}')
print(f'각 열의 합: {my_matrix.sum(axis=1)}')
```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/13.py
원본 배열:
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
모든 요소의 합: 36
각 행의 합: [ 9 12 15]
각 열의 합: [ 3 12 21]
```

14. 주어진 두 벡터의 내적을 계산하기 위해 넘파이 프로그램을 작성해보자

소스 코드:

14.py > ...

```python
## 주어진 두 벡터의 내적 계산
## @ 혹은 dot() 메소드 사용

import numpy as np
a = np.array((4, 5))
b = np.array((7, 10))
print(f'원본 벡터: \n {a} \n {b}')
print(f'벡터의 내적: {a@b}')  # a.dot(b)
```

출력 결과:

```
(main) C:\Users\Enc\Desktop\인공지능>C:/Users/Enc/anaconda3/envs/main/
python.exe c:/Users/Enc/Desktop/인공지능/14.py
원본 벡터:
 [4 5]
 [ 7 10]
벡터의 내적: 78
```

15. [2, 0, 3, 6, 4, 6, 8, 12, 10, 9, 18, 20, 22]와 같은 데이터를 이용하여 선 그래프를 그려보자.

소스 코드 :

15.py > ...

```python
## 주어진 데이터로 선 그래프 그리기

import numpy as np
import matplotlib.pyplot as plt

data = np.array((2, 0, 3, 6, 4, 6, 8, 12, 10, 9, 18, 20, 22))
x_axis = np.arange(len(data))

plt.plot(x_axis, data)
plt.show()
```

출력 결과: