

성명: 김기진(K2020008) --> [파이썬 실습 과제 수행]

Python 코드(<https://cs231n.github.io/python-numpy-tutorial/>)를 직접 작성해 보고 실행 결과를

한글(참조) 사이트: <http://aikorea.org/cs231n/python-numpy-tutorial/>

1. 퀵 정렬 알고리즘 구현 예시

- 퀵 정렬(quick sort) 알고리즘 (<https://gmlwjd9405.github.io/2018/05/10/algorithm-quick-sor>)

```

1  def quicksort(arr):
2      if len(arr) <= 1:
3          return arr
4      pivot = arr[len(arr) // 2]
5      left = [x for x in arr if x < pivot]
6      middle = [x for x in arr if x == pivot]
7      right = [x for x in arr if x > pivot]
8      return quicksort(left) + middle + quicksort(right)
9
10 print(quicksort([20,40,80,8,1,2,1,19,18]))

```

↳ [1, 1, 2, 8, 18, 19, 20, 40, 80]

2. 기본 데이터 유형

1) 숫자

```

1  x = 6
2  print(type(x)) # K2020008 : 숫자 형식
3  print(x)      # K2020008 : "6"
4  print(x + 1)  # K2020008 : 덧셈 "7"
5  print(x - 1)  # K2020008 : 뺄셈 "5"
6  print(x * 2)  # K2020008 : 곱셈 "12"
7  print(x ** 2) # K2020008 : 제곱 "36"
8
9  x += 1
10 print(x) # K2020008 : "7"
11 x *= 2
12 print(x) # K2020008 : "14"
13 y = 2.5
14 print(type(y)) # K2020008 : 숫자 형식
15 print(y, y+1, y*3, y**3)

```

↳

```
<class 'int'>
6
7
5
12
36
7
14
<class 'float'>
2.5 3.5 7.5 15.625
```

2)참,거짓(True, False) -> AND, OR, NOT, XOR 구현 예시 입니다.

```
1 t = True
2 f = False
3 print(type(t)) # K2020008 :bool 형식
4 print(t and f)
5 print(f or t)
6 print(not t)
7 print(t != f)
```

```
↳ <class 'bool'>
False
True
False
True
```

3)문자열

- single quotes(' '), double quotes(" ") 로 사용가능
- 포매팅, 등 구현 예시 입니다.

```
1 hello = 'Hello'
2 world = "world"
3 print(hello)
4 print(len(hello)) # K2020008 : 문자열의 길이를 구하는 함수 (len)
5 hw = hello + ' ' + world # K2020008 : 문자열 연결
6 print(hw)
7 hw11 = '%s %s %d ' % (hello, world, 11) # K2020008 : 문자열 포매팅
8 print(hw11)
```

```
↳ Hello
5
Hello world
Hello world 11
```

3)문자열

- 문자열 객체 주요 메소드

```
1 s = "hello"
```

```

1 s = 'hello'
2 print(s.capitalize()) # K2020008 : 앞글자만 대문자로 변경
3 print(s.upper()) # K2020008 : 소문자를 대문자로 변경
4 print(s.rjust(7)) # K2020008 : 오른쪽 정렬 (전체 7문자)
5 print(s.lstrip) # K2020008 : lstrip(): 문자열 왼쪽을 자름
6 print(s.center(7)) # K2020008 : 가운데 정렬 (전체 7문자)
7 print(s.replace("l", "ell")) # K2020008 : 특정 문자열 변경(대체)
8 print(" world".strip()) # K2020008 : 문자열 양쪽 끝을 자른다. 제거할 문자를 인자로 전달

```

```

↳ Hello
HELLO
hello
<built-in method lstrip of str object at 0x7fcc05cb1c70>
hello
he(ell(ello
world

```

2. 컨테이너(Containers) 유형

1) 리스트(Lists)

```

1 alist = [3, 5, 2] # K2020008 : 리스트는 [] 기호를 사용하여 표현
2 print(alist, alist[2])
3 print(alist[-1]) # K2020008 : 리스트의 마지막 출력값 (alist[2] == alist[-1])
4 alist[2] = "Kim" # K2020008 : 리스트에 다른 형식의 데이터도 입력 가능
5 print(alist)
6 alist.append(".Gi.Jin") # K2020008 : 리스트의 끝 자리에 데이터 입력 가능
7 print(alist)
8 x = alist.pop() # K2020008 : 인덱스에 위치한 값을 리턴하면서 삭제 (인자가 없으면 맨 뒤 값을 pc
9 print(x, alist)

```

```

↳ [3, 5, 2] 2
2
[3, 5, 'Kim']
[3, 5, 'Kim', '.Gi.Jin']
.Gi.Jin [3, 5, 'Kim']

```

2) 리스트(Lists)의 슬라이싱 기본 예제

```

1 nums = list(range(5)) # K2020008 : 리스트 초기화 (0~4)
2 print(nums)
3 print(nums[2:4]) # K2020008 : 슬라이싱 인덱스 2 ~ 4 까지
4 print(nums[2:]) # K2020008 : 슬라이싱 인덱스 2 ~ 끝 까지
5 print(nums[:2]) # K2020008 : 슬라이싱 인덱스 0 ~ 2 까지
6 print(nums[:]) # K2020008 : 슬라이싱 인덱스 처음부터 ~ 끝 까지
7 print(nums[:-1]) # K2020008 : 슬라이싱 인덱스 처음부터 ~ 끝에서 -1 번째 까지
8 nums[2:4] = [2, 10] # K2020008 : 슬라이싱 인덱스 2~4까지 신규 데이터 할당
9 print(nums)

```

```

↳

```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 2, 10, 4]
```

3)리스트(Lists)의 루프(Loops) 기본 예제

```
1 animals = ['cat', 'dog', 'monkey']
2 for animal in animals:
3     print(animal)
```

```
↳ cat
dog
monkey
```

4)리스트(Lists)의 루프(Loops) -> enumerate, append 기본 예제

```
1 animals = ['dog', 'cat', 'pig', 'rabbit']
2 for idx, animal in enumerate(animals): # K2020008 : 리스트가 있는 경우 순서와 리스트의 값을 전
3     print('#%d: %s' % (idx, animal))
4     print('=====')
5
6
7 nums = [0,1,2,3,4,5]
8 squares = [] # K2020008 : 리스트 초기화
9 for x in nums:
10     squares.append(x**2) # K2020008 : 리스트의 제공
11
12 print(squares) # K2020008 : [0,1,2,3,4,5] -> [0, 1, 4, 9, 16, 25]
13
14 squares = [x ** 2 for x in nums] # K2020008 : 리스트의 제공의 다른 표현
15 print(squares) # K2020008 : [0,1,2,3,4,5] -> [0, 1, 4, 9, 16, 25]
16
17
18 even_squares = [x ** 2 for x in nums if x % 2 == 0] # K2020008 : 짝수의 제공
19 print(even_squares) # K2020008 : 짝수의 제공 -> [0, 4, 16]
```

```
↳ #0: dog
=====
#1: cat
=====
#2: pig
=====
#3: rabbit
=====
[0, 1, 4, 9, 16, 25]
[0, 1, 4, 9, 16, 25]
[0, 4, 16]
```

5) 딕셔너리(Dictionaries) -> 임의의 키값을 데이터와 매핑시킬 때 사용하는 구조체입니다

```

1 d = {'cat': 'cute', 'dog': 'furry'} # 딕셔너리 생성
2 print(d['cat']) # K2020008 : "cat"의 value -> "cute"
3 print('cat' in d) # K2020008 : "cat" 이라는 키를 가지고 있는지 확인 -> "True"
4 d['fish'] = 'wet' # K2020008 : 딕셔너리에 신규 키와 값을 생성 'fish': 'wet'
5 print(d['fish']) # K2020008 : "fish"의 값 -> wet
6 # print(d['monkey']) # K2020008 : KeyError: monkey라는 키값이 있는지
7 print(d.get('monkey', 'N/A')) # K2020008 : get "monkey"의 키 값이 없으면 -> "N/A"
8 print(d.get('fish', 'N/A')) # K2020008 : get "fish"의 키값이 있으면 -> "wet"
9 del d['fish'] # K2020008 : "fish" 키값을 딕셔너리에서 제거
10 print(d.get('fish', 'N/A')) # K2020008 : get "fish"의 키값이 없으면 "N/A"

```

```

☞ cute
True
wet
N/A
wet
N/A

```

6) 딕셔너리(Dictionaries) -> Loops 구현 예시

```

1 d = {'person': 2, 'cat': 4, 'spider': 8} # K2020008 : 키와 값의 정의
2 for animal in d:
3     legs = d[animal]
4     print('A %s has %d legs' % (animal, legs)) # K2020008 : 결과 -> "A person has 2 legs", "A
5
6 nums = [0, 1, 2, 3, 4]
7 even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0} # K2020008 : 딕셔너리 키와 값의 생성
8 print(even_num_to_square) # K2020008 : 딕셔너리 결과 생성 -> "{0: 0, 2: 4, 4: 16}"
9

```

```

☞ A person has 2 legs
A cat has 4 legs
A spider has 8 legs
{0: 0, 2: 4, 4: 16}

```

7) 셋(sets) 집합 -> 집합은 {} 기호(괄호)를 사용, 중복데이터 사용 제거됨, 구현 예시

```

1 animals = {'cat', 'dog', 'tiger'}
2 print('cat' in animals) # K2020008 : 셋 집합내에 "cat" 있는지 여부 확인 "True"
3 print('fish' in animals) # K2020008 : 셋 집합내에 "fish" 있는지 여부 확인 "False"
4 animals.add('fish') # K2020008 : 셋 집합내에 "fish" 추가
5 print('fish' in animals) # K2020008 : 셋 집합내에 "fish" 있는지 여부 확인 "True"
6 print(len(animals)) # K2020008 : 셋 항목의 갯수 "4"
7 animals.add('cat') # K2020008 : 셋 집합내에 "fish" 추가 -> 중복은 제거됨
8 print(len(animals)) # K2020008 : 셋 항목의 갯수 "4"
9 animals.remove('cat') # K2020008 : 셋 집합내에 "cat" 제거
10 print(len(animals)) # K2020008 : 셋 항목의 갯수 "3"

```

```

↳ True
False
True
4
4
3

```

8)셋(sets) 집합 -> enumerate, 연산 사용 예시

```

1 animals = {'cat', 'fish', 'tiger'}
2 for idx, animal in enumerate(animals): # K2020008 :
3     print('#%d: %s' % (idx+1, animal)) # K2020008 : 출력 -> "#1: fish", "#2: fish", "#3: fish"
4
5 from math import sqrt
6 nums = {int(sqrt(x)) for x in range(30)} # K2020008 :셋은 리스트와 디션러리와 같이 쉽게 생성
7 print(nums)

```

```

↳ #1: fish
#2: cat
#3: tiger
{0, 1, 2, 3, 4, 5}

```

9)튜플(Tuples) -> 읽기 전용으로 변경 불가, 사용 예시

```

1 d = {(x, x + 1): x for x in range(10)} # K2020008 : 튜플 포함한 딕셔너리 생성
2 t = (5, 6) # K2020008 : 튜플 생성
3 print(type(t)) # K2020008 : 타입 "<class 'tuple'>"
4 print(d) # K2020008 : 딕셔너리 튜플 생성 결과 : {(0, 1): 0, (1, 2): 1, (2, 3): 2, (3, 4): 3,
5 print(d[t]) # K2020008 : (5, 6)의 값 5
6 print(d[(1, 2)]) # K2020008 : (1, 2)의 값 1

```

```

↳ <class 'tuple'>
{(0, 1): 0, (1, 2): 1, (2, 3): 2, (3, 4): 3, (4, 5): 4, (5, 6): 5, (6, 7): 6, (7, 8): 7, (8, 9): 8, (9, 10): 9}
5
1

```

3.함수(Functions)

함수는 def 키워드를 사용 -> 구현 예시

```

1 ✓ def sum(a, b): # K2020008 : sum 함수 (더하기 기능)
2 ✓     if a < 0:
3         return 'a 값이 0 보다 작습니다'
4
5 ✓     if b < 0:
6         return 'b 값이 0 보다 작습니다'
7
8     return (a+b)
9
10 ✓ for idx, n in enumerate(range(5)):

```

```

11     hap = sum(idx, n)
12     print('더하기 {0} + {0} = {0}' % (idx, n, hap))
13
14
15 ✓ def hello(name, loud=False): # K2020008 : hello 함수 (인자를 조건으로 문자열 출력))
16 ✓     if loud:
17         print('HELLO, %s!' % name.upper())
18 ✓     else:
19         print('Hello, %s' % name)
20
21     hello('Bob') # K2020008 : 출력 "Hello, Bob"
22     hello('Fred', loud=True) # K2020008 : 출력 "HELLO, FRED!"

```

```

☞ 더하기 {0} + {0} = {0}
   더하기 {1} + {1} = {2}
   더하기 {2} + {2} = {4}
   더하기 {3} + {3} = {6}
   더하기 {4} + {4} = {8}
   Hello, Bob
   HELLO, FRED!

```

4.클래스(Class)

클래스는 객체(인스턴스)를 생성하기 위해 필요하다, 객체지향 프로그래밍(OOP)을 위해 사용된다

- 클래스 상세설명 참조 : <https://withcoding.com/82>

```

1 class Greeter(object):
2
3     # K2020008 : 클래스 생성자
4     def __init__(self, name):
5         self.name = name # K2020008 : 생성자 인스턴스 변수
6
7     # K2020008 : 인스턴스 메소드
8     def greet(self, loud=False):
9         if loud:
10             print('HELLO, {}'.format(self.name.upper())) # K2020008 : .format 형식으로 출력
11         else:
12             print('Hello, %s' % (self.name))
13
14 g = Greeter('Fred') # K2020008 : Greeter 클래스의 인스턴스를 생성한다
15 g.greet()           # K2020008 : 메소드 콜 (파라미터 디폴트): "Hello, Fred"
16 g.greet(loud=True) # K2020008 : 메소드 콜 (파라미터 loud = True) "HELLO, FRED!"

```

```

☞ Hello, Fred
   HELLO, FRED!

```

5.Numpy

numpy는 과학 계산을 위한 라이브러리로서 다차원 배열을 처리하는데 필요한 여러 유용한 기능을 제공한다

1)Numpy 배열

numpy에서 배열은 동일한 타입의 값들을 가지며, 배열의 차원을 rank 라 하고, 각 차원의 크기를 다. 예를 들어, 행이 2이고 열이 3인 2차원 배열에서 rank는 2 이고, shape는 (2, 3) 이 된다.

```

1 import numpy as np
2
3 a = np.array([10, 20, 30]) # K2020008 : 리스트의 3개 요소를 갖는 배열 생성
4 print(type(a))           # K2020008 : "<class 'numpy.ndarray'>"
5 print(a.shape)           # K2020008 : rank는 1이되고, shape "(3,)"
6 print(a[0], a[1], a[2])  # K2020008 : 출력 "10 20 30"
7 a[0] = 5                 # K2020008 : 배열의 요소 변경
8 print(a)                 # K2020008 : 출력 "[5, 20, 30]"
9
10 b = np.array([[1,2,3],[4,5,6]]) # K2020008 : 배열 2*3 배열 생성
11 print(b.shape)             # K2020008 : shape"(2, 3)"
12 print(b[0, 0], b[0, 1], b[1, 0]) # K2020008 : 출력 "1 2 4"

```

```

↳ <class 'numpy.ndarray'>
   (3,)
   10 20 30
   [ 5 20 30]
   (2, 3)
   1 2 4

```

2)Numpy는 많은 함수 제공 -> 예제는 이들 함수들을 사용하여 numpy 배열을 생성한 예이다

```

1 import numpy as np
2
3 a = np.zeros((2,2))
4 print(a)
5 # K2020008 : 출력:
6 # K2020008 : [[ 0.  0.]
7 # K2020008 : [ 0.  0.]]
8
9 a = np.ones((2,3)) # K2020008 : 2*3 행렬을 1로 초기화
10 print(a)
11 # K2020008 : 출력:
12 # K2020008 : [[ 1.  1.  1.]
13 # K2020008 : [ 1.  1.  1.]]
14
15 a = np.full((2,3), 5) # K2020008 : 2*3 행렬을 5로 초기화
16 print(a)
17 # K2020008 : 출력:
18 # K2020008 : [[5 5 5]
19 # K2020008 : [5 5 5]]
20
21 a = np.eye(3) # K2020008 : 대각선으로는 1이고 나머지는 0인 2차원 배열을 생성한다.
22 print(a)
23 # K2020008 : 출력:
24 # K2020008 : [[ 1.  0.  0.]
25 # K2020008 : [ 0.  1.  0.]
26 # K2020008 : [ 0.  0.  1.]]
27
28 a = np.array(range(20)).reshape((4,5)) # K2020008 : reshape()은 배열을 다차원(4*5)으로 변환

```



```

29 print(a)
30 # K2020008 : 출력:
31 # K2020008 : [[ 0  1  2  3  4]
32 # K2020008 : [ 5  6  7  8  9]
33 # K2020008 : [10 11 12 13 14]
34 # K2020008 : [15 16 17 18 19]]
35
36 e = np.random.random((2,2)) # K2020008 : random()은 배열을 다차원(2*2)을 random 값으로 초기화
37 print(e)
38
39 # K2020008 : 출력:
40 # K2020008 : "[[0.41218293 0.37651471]
41 # K2020008 : [0.24827098 0.75018894]]"
42

```

```

[[0. 0.]
 [0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]]
[[5 5 5]
 [5 5 5]]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
[[0.69923376 0.73025406]
 [0.36373847 0.19059536]]

```

3)Array indexing -> 인덱싱은 각 차원별로 선택되어지는 배열요소의 인덱스들을 일렬로 나열하

```

1 import numpy as np
2 # [[ 1  2  3  4]
3 # [ 5  6  7  8]
4 # [ 9 10 11 12]]
5 a = np.array([[1,2,3,4],[5,6,7,8], [9,10,11,12]]) # K2020008 : 2차원 shape(3,4) 생성
6 # [[2 3]
7 # [6 7]]
8 b = a[:2, 1:3]
9 print(a[0, 3]) # K2020008 : 출력 "4"
10 b[0, 1] = 99 # K2020008 : b[0, 0] => a[0, 2]
11 print(a[0, 2]) # K2020008 : 출력 "99"

```

```

4
99

```

4)numpy 슬라이싱->numpy 배열은 파이썬 리스트와 마찬가지로 슬라이스(Slice)를 지원한다. (

```

1 import numpy as np
2
3 lst = [

```

```

4      [1, 2, 3],
5      [4, 5, 6],
6      [7, 8, 9]
7  ]
8  arr = np.array(lst)
9  print(lst)
10
11  # K2020008 : 슬라이스
12  a = arr[0:2, 0:2]
13  print(a)
14  # K2020008 : 출력:
15  # K2020008 : [[1 2]
16  # K2020008 : [4 5]]
17
18  a = arr[1:, 1:]
19  print(a)
20  # K2020008 : 출력:
21  # K2020008 : [[5 6]
22  # K2020008 : [8 9]]
23
24  # K2020008 : 2차원 shape(3,4) 생성
25  # K2020008 : [[ 1  2  3  4]
26  # K2020008 : [ 5  6  7  8]
27  # K2020008 : [ 9 10 11 12]]
28  a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
29
30  row_r1 = a[1, :] # K2020008 : 1 차원 : 스칼라
31  row_r2 = a[1:2, :] # K2020008 : 2 차원 : 벡터
32  print(row_r1, row_r1.shape) # K2020008 : 출력 "[5 6 7 8] (4,)"
33  print(row_r2, row_r2.shape) # K2020008 : 출력 "[[5 6 7 8]] (1, 4)"
34
35  col_r1 = a[:, 1] # K2020008 : 1 차원 : 스칼라
36  col_r2 = a[:, 1:2] # K2020008 : 2 차원 : 벡터
37  print(col_r1, col_r1.shape) # K2020008 : 출력 "[ 2  6 10] (3,)"
38  print(col_r2, col_r2.shape) # K2020008 : 출력 "[[ 2]
39  # K2020008 : [ 6]
40  # K2020008 : [10]] (3, 1)"
41
42  a = np.array([[1,2], [3,4], [5,6]]) # K2020008 : 2차원 shape(3,2) 생성
43  print(a.shape)
44  print(a[[0, 1, 2], [0, 1, 0]]) # K2020008 : 출력 "[1 4 5]"
45
46  print(np.array([a[0, 0], a[1, 1], a[2, 0]])) # K2020008 : 출력 "[1 4 5]"
47
48  # K2020008 : a[[0, 1, 2], [0, 1, 0]]와 np.array([a[0, 0], a[1, 1], a[2, 0]])은 같은 식임
49  print(a[[0, 0], [1, 1]]) # K2020008 : 출력 "[2 2]"
50
51  # K2020008 : a[[0, 0], [1, 1]]와 np.array([a[0, 1], a[0, 1]])은 같은 식임
52  print(np.array([a[0, 1], a[0, 1]])) # K2020008 : 출력 "[2 2]"

```



```

[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[[1 2]
 [4 5]]
[[5 6]
 [8 9]]
[5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)
(3, 2)
[1 4 5]
[1 4 5]
[2 2]
[2 2]

```

5)numpy 정수 인덱싱 (integer indexing)

```

1  import numpy as np
2
3  # K2020008 : 배열 생성 (4*3)
4  a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
5
6  print(a) # # K2020008 : 출력 "array([[ 1,  2,  3],
7          #                      [ 4,  5,  6],
8          #                      [ 7,  8,  9],
9          #                      [10, 11, 12]])"
10
11 # K2020008 : 지수 배열 생성
12 b = np.array([0, 2, 0, 1])
13
14 print(a[np.arange(4), b]) # K2020008 : 출력 "[ 1  6  7 11]"
15
16 a[np.arange(4), b] += 20
17
18 print(a) # K2020008 : 출력 "array([[21,  2,  3],
19          #                      [ 4,  5, 16],
20          #                      [17,  8,  9],
21          #                      [10, 21, 12]])"

```

```

[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[ 1  6  7 11]
[[21  2  3]
 [ 4  5 26]
 [27  8  9]
 [10 31 12]]


```

6)numpy 부울린 인덱싱 (boolean indexing)->배열 각 요소의 선택여부를 True, False로 표현하는

```

1  import numpy as np
2
3  a = np.array([[1,2], [3, 4], [5, 6]])
4
5  bool_idx = (a > 2) # K2020008 : 배열 요소 중 2 보다 큰 값을 True 혹은 False 설정
6  print(bool_idx)   # K2020008 : 출력 "[False False]
7                    #                    [ True  True]
8                    #                    [ True  True]]"
9
10 # K2020008 : 1차원 배열 인덱싱
11 # K2020008 : 1차원 중 True 것만 찾아서 출력
12 print(a[bool_idx]) # K2020008 : 출력 "[3 4 5 6]"
13
14 # K2020008 : a[bool_idx]과 a[ a > 2 ] 동일한 결과를 출력 한다
15 print(a[a > 2])   # K2020008 : 출력 "[3 4 5 6]"

```



[[False False]
[True True]
[True True]]
[3 4 5 6]
[3 4 5 6]

7)DataTypes->numpy는 큰수의 데이터 타입을 제공 한다 (아래 예시)

```

1  import numpy as np
2
3  x = np.array([1, 2]) # K2020008 : numpy 데이터 타입
4  print(x.dtype)      # K2020008 : 출력 "int64"
5
6  x = np.array([1.0, 2.0]) # K2020008 : numpy 데이터 타입
7  print(x.dtype)       # K2020008 : 출력 "float64"
8
9  x = np.array([1, 2], dtype=np.int64) # K2020008 : numpy 데이터 타입의 강제 설정
10 print(x.dtype)       # K2020008 : 출력 "int64"

```


int64
float64
int64

8)numpy 연산->numpy를 사용하면 배열간 연산을 쉽게 실행할 수 있다. 연산은 +, -, *, / 등의 연산. subtract(), multiply(), divide() 등의 함수를 사용할 수도 있다 (아래 예시)

```

1  import numpy as np
2  x = np.array([[1,2],[3,4]], dtype=np.float64)
3  y = np.array([[5,6],[7,8]], dtype=np.float64)
4
5  # K2020008 : 각 요소 더하기
6  print(x + y)
7  print(np.add(x, y))
8  # K2020008 : [[ 6.  8.]
9  # K2020008 : [10. 12.]]
10

```

```

11 # K2020008 : 각 요소 빼기
12 c = x - y
13 # K2020008 : c = np.subtract(x, y)
14 print(c)
15 # K2020008 : [[-4. -4.]
16 # K2020008 : [-4. -4.]]
17 # 각 요소 곱하기
18 # c = x * y
19 c = np.multiply(x, y)
20 print(c)
21 # K2020008 : [[ 5. 12.]
22 # K2020008 : [21. 32.]]
23 # 각 요소 나누기
24 # c = x / y
25 c = np.divide(x, y)
26 print(c)
27 # K2020008 : [[0.2          0.33333333]
28 # K2020008 : [0.42857143 0.5          ]]
29
30 # 각 요소 제곱근(루트)
31 print(np.sqrt(c))
32 # K2020008 : [[0.4472136  0.57735027]
33 # K2020008 : [0.65465367 0.70710678]]

```

```

☞ [[ 6.  8.]
   [10. 12.]]
   [[ 6.  8.]
   [10. 12.]]
   [[-4. -4.]
   [-4. -4.]]
   [[ 5. 12.]
   [21. 32.]]
   [[0.2          0.33333333]
   [0.42857143 0.5          ]]
   [[0.4472136  0.57735027]
   [0.65465367 0.70710678]]

```

9) numpy에서 vector와 matrix의 product를 구하기 위해서 dot() 함수를 사용한다. 두개의 matri

```

1 import numpy as np
2
3 x = np.array([[1,2],[3,4]])
4 y = np.array([[5,6],[7,8]])
5
6 v = np.array([9,10])
7 w = np.array([11, 12])
8
9 # K2020008 : 벡터의 내적 : 둘다 값은 동일 (약간 이해 부족)
10 print(v.dot(w))
11 print(np.dot(v, w))
12
13 # K2020008 : 행렬과 벡터의 곱 : 1차원 [29 67] (약간 이해 부족)
14 print(x.dot(v))
15 print(np.dot(x, v))

```

```

16
17 # K2020008 : 행렬과 행렬의 곱 : 2차원 (약간 이해 부족)
18 print(x.dot(y))
19 print(np.dot(x, y))
20 # K2020008 : [[19 22]
21 # K2020008 : [43 50]]
22

```

```

↳ 219
219
[29 67]
[29 67]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]

```

10)Numpy는 배열 연산에 유용하게 쓰이는 많은 함수를 제공 (sum 함수)

```

1 import numpy as np
2
3 x = np.array([[1,2],[3,4]])
4
5 print(np.sum(x)) # K2020008 : 모든 요소를 합한 값을 연산; 출력 "10"
6 print(np.sum(x, axis=0)) # K2020008 : 행 = 0 각 열에 대한 합을 연산; 출력 "[4 6]"
7 print(np.sum(x, axis=1)) # K2020008 : 열 = 0 각 행에 대한 합을 연산; 출력 "[3 7]"

```

```

↳ 10
[4 6]
[3 7]

```

11)Numpy는 전치행렬을 위해서 간단하게 배열 객체의'T'속성을 사용

```

1 import numpy as np
2
3 x = np.array([[1,2], [3,4]])
4 print(x) # K2020008 : 출력 "[[1 2]
5 # [3 4]]"
6 print(x.T) # K2020008 : 출력 "[[1 3] -> 행과 열을 바꾼다(전치)
7 # [2 4]]"
8
9 # K2020008 : 1차원 배열을 전치할 경우 변화 없음
10 v = np.array([1,2,3])
11 print(v) # K2020008 : 출력 "[1 2 3]"
12 print(v.T) # K2020008 : 출력 "[1 2 3]"

```

```

↳ [[1 2]
 [3 4]]
[[1 3]
 [2 4]]
[1 2 3]
[1 2 3]

```

12)브로드캐스팅(Broadcasting)은 Numpy에서 shape가 다른 배열 간에도 산술 연산이 가능하

```

1  import numpy as np
2
3  # K2020008 :행렬 x의 각 행에 벡터 v를 더한 뒤,결과를 행렬 y에 저장
4  x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
5  v = np.array([1, 0, 1])
6  y = np.empty_like(x)   # x와 동일한 shape를 가지는 비어있는 행렬 생성
7
8  # K2020008 :명시적 반복문을 통해 행렬 x의 각 행에 벡터 v를 더하는 방법
9  for i in range(4):
10     y[i, :] = x[i, :] + v
11
12  print(y)
13  # K2020008 : 출력 결과
14  # [[ 2  2  4]
15  #   [ 5  5  7]
16  #   [ 8  8 10]
17  #   [11 11 13]]
18

```

```

[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]

```

13)벡터 'v'를 행렬 'x'의 각 행에 더하는 것은 'v'를 여러 개 복사해 수직으로 쌓은 행렬 'vv'를 만들 (계산 예시)

```

1  import numpy as np
2
3  # K2020008 : 벡터 v를 행렬 x의 각 행에 더한 뒤,
4  # K2020008 : 그 결과를 행렬 y에 저장하고자 합니다
5  x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
6  v = np.array([1, 0, 1])
7  vv = np.tile(v, (4, 1)) # K2020008 : v의 copy 4개를 위로 저장 쌓은 것이 vv
8  print(vv)               # K2020008 : 출력 "[[1 0 1]
9                           #               [1 0 1]
10                          #               [1 0 1]
11                          #               [1 0 1]]"
12  y = x + vv # K2020008 : x와 vv의 요소별 합
13  print(y)   # K2020008 : 출력 "[[ 2  2  4
14                          #       [ 5  5  7]
15                          #       [ 8  8 10]
16                          #       [11 11 13]]"

```

```


```

```
[[1 0 1]
 [1 0 1]
 [1 0 1]
 [1 0 1]]
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

14)Numpy 브로드캐스팅을 이용한다면 v의 복사본을 여러 개 만들지 않아도 동일한 연산을 할

```
1 import numpy as np
2
3 # K2020008 : 벡터 v를 행렬 x의 각 행을 더한뒤 행렬 y에 저장
4 x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
5 v = np.array([1, 0, 1])
6 y = x + v # K2020008 : 브로드캐스팅을 이용하여 v를 x의 각 행에 더하기
7 print(y) # K2020008 : 출력 "[[ 2  2  4]
8          #          [ 5  5  7]
9          #          [ 8  8 10]
10         #          [11 11 13]]"
```

```
↳ [[ 2  2  4]
    [ 5  5  7]
    [ 8  8 10]
    [11 11 13]]
```

15)브로드캐스팅을 지원하는 함수를 universal functions 하고 한다 (예시 코드)

```
1 import numpy as np
2
3 # K2020008 : 벡터의 외적 계산
4 v = np.array([1,2,3]) # K2020008 : v = shape(3,)
5 w = np.array([4,5])    # K2020008 : w = shape(2,)
6 # K2020008 : 외적을 계산 -> v를 shape가 (3,1)인 행벡터로 변경
7 # K2020008 : w에 맞춰 브로드캐스팅한뒤 결과물로 shape가 (3,2)인 행렬
8
9 print(np.reshape(v, (3, 1)) * w)
10 # K2020008 : 행렬은 v와 w 외적의 결과입니다:
11 # K2020008 : 출력
12 # [[ 4  5]
13 #  [ 8 10]
14 #  [12 15]]
15
16 # K2020008 : 벡터를 행렬의 각 행에 더하기
17 x = np.array([[1,2,3], [4,5,6]])
18 # K2020008 : x는 shape가 (2, 3)이고 v는 shape가 (3,)이므로 이 둘을 브로드캐스팅하면 shape가 (2
19
20 print(x + v)
21 # K2020008 : 출력
22 # [[2 4 6]
23    [5 7 9]]
```



```

23 # [[ 5 / 9]]
24
25 # K2020008 : 벡터를 행렬의 각 행에 더하기
26 # K2020008 : x.shape(2, 3)이고 w.shape가 (2,)
27 # K2020008 : x의 전치행렬은 shape가 (3,2)이며 w와 브로드캐스팅이 가능하고 결과로 shape가 (3,2)
28 # K2020008 : 전치행렬 shape(2,3)
29 print((x.T + w).T)
30 # K2020008 : 출력
31 # [[ 5  6  7]
32 #   [ 9 10 11]]
33
34 # K2020008 : 다른 방법은 w를 shape가 (2,1)인 열벡터로 변환하는 것입니다;
35 # K2020008 : 그런 다음 이를 바로 x에 브로드캐스팅해 더하면
36 # K2020008 : 동일한 결과가 나옵니다.
37 print(x + np.reshape(w, (2, 1)))
38
39 # K2020008 : 행렬의 스칼라배:
40 # K2020008 : x.shape(2, 3) Numpy는 스칼라를 shape가 ()인 배열로 취급합니다;
41 # K2020008 : 그렇기에 스칼라 값은 (2,3) shape
42 print(x * 2)
43 # K2020008 : 출력
44 # [[ 2  4  6]
45 #   [ 8 10 12]]

```

```

[[ 4  5]
 [ 8 10]
 [12 15]]
[[2 4 6]
 [5 7 9]]
[[ 5  6  7]
 [ 9 10 11]]
[[ 5  6  7]
 [ 9 10 11]]
[[ 2  4  6]
 [ 8 10 12]]

```

6.SciPy

1)numpy를 바탕으로 만들어진 SciPy는 numpy 배열을 다루는 많은 함수를 제공하며 다양한 과

- 이미지 작업

```

1 import os
2 import numpy as np
3 import imageio
4 from skimage import data, io, transform
5 # K2020008 : scipy 1.2.0부터 지원 (imread, imsave, imresize) 함수 미지원 (제거됨)
6 # K2020008 : scipy.misc.imread -> imageio.imread 대신 사용
7 # K2020008 : scipy.misc.imsave -> imageio.imwrite 대신 사용
8 # K2020008 : scipy.misc.imresize -> skimage.transform.resize 대신 사용
9
10 # K2020008 : Google 드라이버 임포트 (이미지 경로 지정을 위해)
11 from google.colab import drive

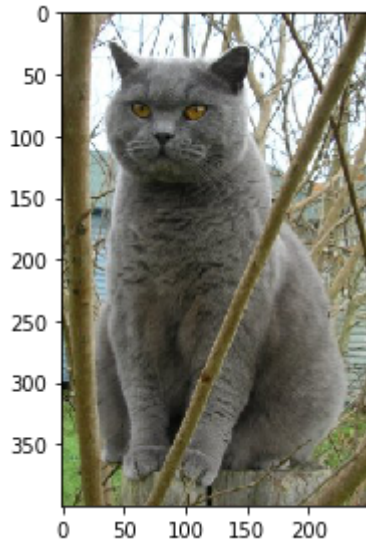
```

```

11 from google.colab import drive
12 drive.mount('/content/drive', force_remount=True)
13
14 # K2020008 : Google 드라이브에서 JPEG 이미지를 numpy 배열로 읽어들이기
15 # K2020008 : img = scipy.misc.imread('cat.jpg') -> 미사용
16 img = io.imread('/content/drive/My Drive/00.AI_TM/cat.jpg')
17 print(img.dtype, img.shape) # K2020008 : 출력 "uint8 (400, 248, 3)"
18 io.imshow(img)

```

Mounted at /content/drive
 uint8 (400, 248, 3)
 <matplotlib.image.AxesImage at 0x7f5c6d72ab70>



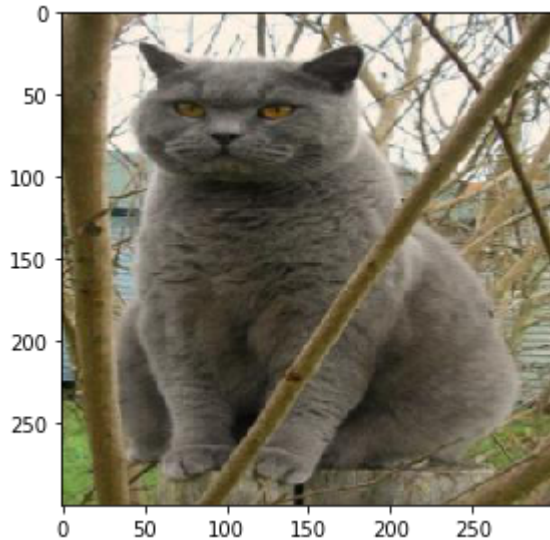
```

1 # K2020008 : 각각의 색깔 채널을 다른 상수값으로 스칼라배함으로써
2 # K2020008 : 이미지의 색을 변화시킬 수 있습니다.
3 # K2020008 : 이미지의 shape는 (400, 248, 3)입니다;
4 # K2020008 : 여기에 shape가 (3,)인 배열 [1, 0.95, 0.9]를 곱합니다;
5 # K2020008 : numpy 브로드캐스팅에 의해 이 배열이 곱해지며 붉은색 채널은 변하지 않으며,
6 # K2020008 : 초록색, 파란색 채널에는 각각 0.95, 0.9가 곱해집니다
7 img_tinted = img * [1, 0.95, 0.9]
8
9 # K2020008 : 색변경 이미지를 400x400픽셀로 크기 조절.
10 # K2020008 : img_tinted = imresize(img_tinted, (300, 300)) -> 미사용
11 img_tinted = transform.resize(img_tinted, (300, 300))
12 # io.imshow(transform.resize(img_tinted, (300, 300)))
13 # K2020008 : 색변경 이미지를 디스크에 기록하기
14 # K2020008 : imsave('assets/cat_tinted.jpg', img_tinted) -> -> 미사용
15 imageio.imwrite('/content/drive/My Drive/00.AI_TM/cat_tinted.jpg', img_tinted)
16 img = io.imread('/content/drive/My Drive/00.AI_TM/cat_tinted.jpg')
17 print(img.dtype, img.shape) # K2020008 : 출력 "uint8 (400, 248, 3)"
18 io.imshow(img)

```

↳

WARNING:root:Lossy conversion from float64 to uint8. Range [0.0, 255.00000000000003]. Convert
 uint8 (300, 300, 3)
 <matplotlib.image.AxesImage at 0x7f5c6d9bbdd8>



2)두 점 사이의 거리(Distance between points)

```

1  import numpy as np
2  from scipy.spatial.distance import pdist, squareform
3
4  # K2020008 : 각 행이 2차원 공간에서의 한 점을 의미하는 행렬을 생성:
5  x = np.array([[0, 1], [1, 0], [2, 0]])
6  print(x)
7  # K2020008 : 출력
8  # [[0 1]
9  #   [1 0]
10 #   [2 0]]
11
12 # K2020008 : x가 나타내는 모든 점 사이의 유클리디안 거리를 계산.
13 # K2020008 : d[i, j]는 x[i, :]와 x[j, :]사이의 유클리디안 거리를 의미하며,
14 # K2020008 : d는 아래의 행렬입니다:
15 d = squareform(pdist(x, 'euclidean'))
16 print(d)
17 # K2020008 : 출력
18 # [[ 0.          1.41421356  2.23606798]
19 #   [ 1.41421356  0.          1.          ]
20 #   [ 2.23606798  1.          0.          ]]

```

```

☞ [[0 1]
   [1 0]
   [2 0]]
   [[0.          1.41421356  2.23606798]
   [ 1.41421356  0.          1.          ]
   [ 2.23606798  1.          0.          ]]

```

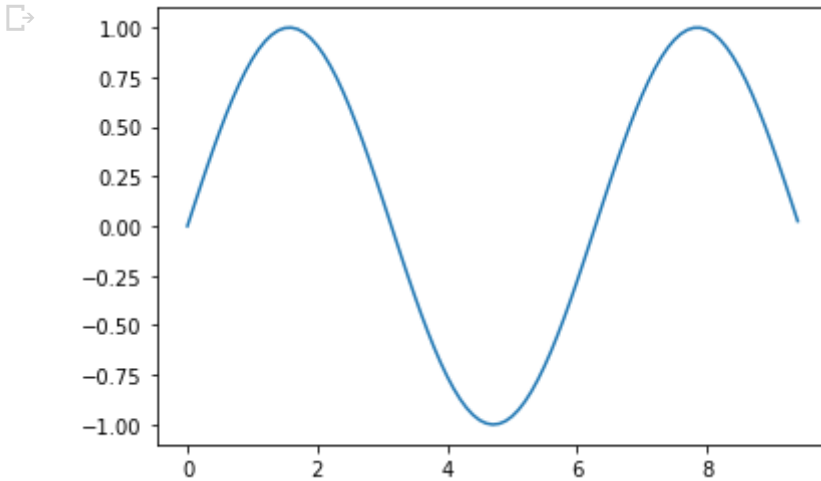
7. Matplotlib

1)'plotting' 라이브러리입니다. 이번에는 MATLAB의 plotting 시스템과 유사한 기능을 제공하는

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # K2020008 : 사인과 코사인 곡선의 x,y 좌표를 계산
5  x = np.arange(0, 3 * np.pi, 0.1)
6  y = np.sin(x)
7
8  # K2020008 : matplotlib를 이용해 점들을 그리기
9  plt.plot(x, y)
10 plt.show() # K2020008 : 그래프로 보여주기

```



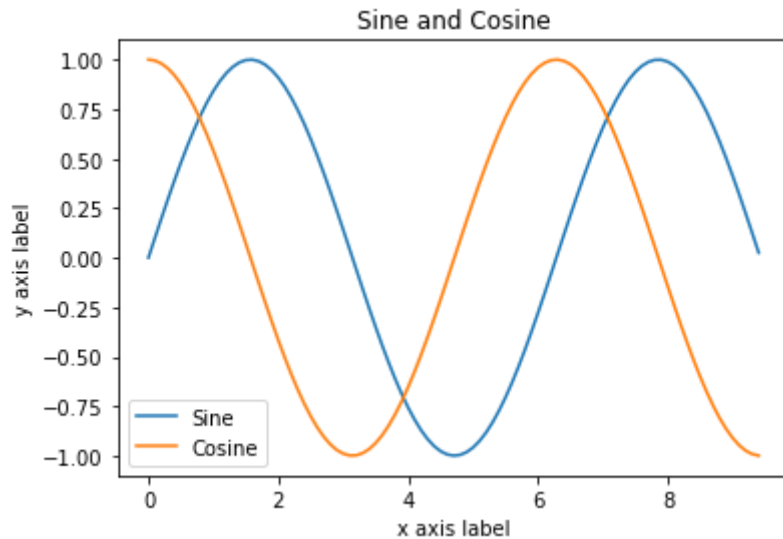
2) 추가적인 작업을 통해 여러 개의 그래프와 제목, 범주, 축 이름 표현을 그려 보기 (예시)

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # K2020008 : 사인과 코사인 곡선의 x,y 좌표를 계산
5  x = np.arange(0, 3 * np.pi, 0.1)
6  y_sin = np.sin(x)
7  y_cos = np.cos(x)
8
9  # K2020008 : matplotlib를 이용해 점들을 그리기
10 plt.plot(x, y_sin)
11 plt.plot(x, y_cos)
12 plt.xlabel('x axis label')
13 plt.ylabel('y axis label')
14 plt.title('Sine and Cosine')
15 plt.legend(['Sine', 'Cosine'])
16 plt.show()

```





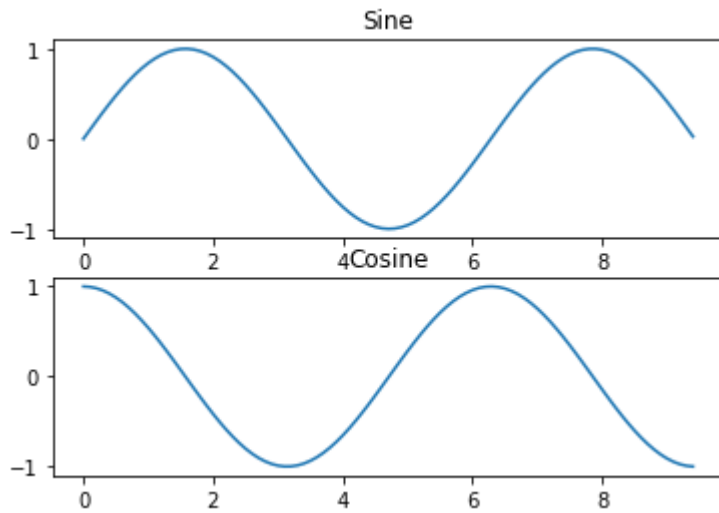
3)'subplot'함수를 통해 다른 내용도 동일한 그림 위에 나타낼 수 있습니다. 여기 간단한 예시가

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # K2020008 : 사인과 코사인 곡선의 x,y 좌표를 계산
5  x = np.arange(0, 3 * np.pi, 0.1)
6  y_sin = np.sin(x)
7  y_cos = np.cos(x)
8
9  # K2020008 : 높이가 2이고 너비가 1인 subplot 구획을 설정하고,
10 # K2020008 : 첫 번째 구획을 활성화.
11 plt.subplot(2, 1, 1)
12
13 # K2020008 : 첫 번째 그리기
14 plt.plot(x, y_sin)
15 plt.title('Sine')
16
17 # K2020008 : 두 번째 subplot 구획을 활성화 하고 그리기
18 plt.subplot(2, 1, 2)
19 plt.plot(x, y_cos)
20 plt.title('Cosine')
21
22 # K2020008 : 그림 보이기.
23 plt.show()

```





4) `imshow` 함수를 사용해 이미지를 나타낼 수 있습니다. (예시)

```

1  import numpy as np
2  import imageio
3  from skimage.transform import resize
4  import matplotlib.pyplot as plt
5
6  # K2020008 : scipy 1.2.0부터 지원 (imread, imsave, imresize) 함수 미지원 (제거됨)
7  # K2020008 : scipy.misc.imread -> imageio.imread 대신 사용
8  # K2020008 : scipy.misc.imsave -> imageio.imwrite 대신 사용
9  # K2020008 : scipy.misc.imresize -> skimage.transform.resize 대신 사용
10
11 # K2020008 : Google 드라이버 임포트 (이미지 경로 지정을 위해)
12 from google.colab import drive
13 drive.mount('/content/drive')
14
15 # K2020008 : Google 드라이버에서 JPEG 이미지를 numpy 배열로 읽어들이기
16 # K2020008 : img = scipy.misc.imread('cat.jpg') -> 미사용
17 img = imageio.imread('/content/drive/My Drive/00.AI_TM/cat.jpg')
18 img_tinted = img * [1, 0.95, 0.9]
19
20 # K2020008 : 원본 이미지 나타내기
21 plt.subplot(1, 2, 1)
22 plt.imshow(img)
23
24 # K2020008 : 색변화된 이미지 나타내기
25 plt.subplot(1, 2, 2)
26
27 # K2020008 : imshow를 이용하며 주의할 점은 데이터의 자료형이
28 # K2020008 : uint8이 아니라면 이상한 결과를 보여줄 수도 있다는 것입니다.
29 # K2020008 : 그러므로 이미지를 나타내기 전에 명시적으로 자료형을 uint8로 형변환 해줍니다.
30 plt.imshow(np.uint8(img_tinted))
31 plt.show() # K2020008 : 그래프로 보여주기

```



Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cc

