Python
기초 수학

Python
기초 수학

1. 집합

```
#리스트 안의 원소 유무 판별하기
a = [3,1,6,4,9]
print(1 in a)
print(0 in a)
```

```
# 파이썬의 set 자료형이 집합을 지원하는 자료형이지만
# 여기서는 개념을 익히기 위해 직접 코딩하는 방식으로 실습


#합집합
a = [1,2,3]
b = [4,5,6]
c = a + b
print (c)
```

```
#중복값이 있는 경우
a = [1,2,3]
b = [3,4,5]
print (a + b)
```

```
#중복값을 제거
c = []
```

```
for i in a:
    if i not in b:
        c.append(i)
c = c + b
c.sort()
print (c)
```

```
#a와 b의 합집합
a = ['사과', '포도', '오렌지']
b = ['배', '자두', '귤']

c = a + b
print(c)
```

```
b[1] = '포도'
print(b)
```

```
#중복값 확인
a+b
#중복값을 제거
c = []
for i in a:
    if i not in b:              # i가 b에 속하지 않으면
        c.append(i)             # 합집합 c에 i를 추가
c = c + b
c.sort()
print (c)
```

```
#교집합
a = [1,2,3]
b = [3,4,5]

c = [ ]
for i in a:
    if i in b:          # i가 b에 속하면
        c.append(i)  # 교집합 c에 추가
print (c)
```

```
#차집합
a = [1,2,3]
b = [3,4,5]

c = a                          # 얕은 복사, c가 a를 가리킴
for i in b:
    if i in a:          #i가 a에 속하면
        c.remove(i)  #c에서 i를 삭제

#얕은 복사이므로 c와 a가 같음
print(c)
print(a)
```

```
#차집합
a = [1,2,3]
b = [3,4,5]
```

```
c = a + [ ]                # 리스트 복사(깊은 복사)
#c=a.copy()
for i in b:
    if i in a:          #i가 a에 속하면
        c.remove(i)  #c에서 i를 삭제

print(c)
print(a)
```

#여집합
#ex)a의 여집합 : 전체 집합 중 a에 속하지 않는 모든 요소들의 집합

```
#전체집합 a에서 b의 여집합을 구하는 코드
a = [0,1,2,3,4,5,6,7,8,9]
b = [1,3,5,7,9]
result = a + [ ]                # 전체집합을 복사
for i in b:
    if i in a:                # i가 a에 속하면
        result.remove(i)    # result에서 i를 삭제

print(result)
```

```
#합집합 함수
def union(a,b):
    c = [ ]
    for i in a:
```

```
            if i not in b:              # i가 b에 속하지 않으면
                c.append(i)             # 합집합 c에 i를 추가
    c = c + b
    c.sort()
    return c

a = [1,3,4,5,6]
b = [2,1,6,3,9]
union(a,b)
```

```
#교집합 함수
def intersection(a,b):
    c = []
    for i in a:
        if i  in b:
            c.append(i)
    c.sort()
    return c

a = [1,3,4,5,6]
b = [2,1,6,3,9]
intersection(a,b)
```

```
#차집합
def complement(a,b):
    c = a + []
    for i in b:
```

```
            if i in a:
                c.remove(i)
        c.sort()
        return c


a = [1,3,4,5,6]
b = [2,1,6,3,9]

complement(a,b)
```

```
#집합 클래스
class MySet:
    def setdata(self,a,b):
        self.a = a
        self.b = b
    #합집합
    def union(self):
        result = self.b + []
        for i in self.a:
            if i not in self.b:
                result.append(i)
        result.sort()
        return result
    #교집합
    def intersection(self):
        result = []
```

```
            for i in self.a:
                if i  in self.b:
                    result.append(i)
            result.sort()
            return result
        #차집합
        def complement(self):
            result = self.a + []
            for i in self.b:
                if i in self.a:
                    result.remove(i)
            result.sort()
            return result



c = MySet()

a = [1,3,4,5,6]
b = [2,1,6,3,9]

c.setdata(a,b)
print ('합집합:', c.union() )
print ('교집합:', c.intersection() )
print ('차집합:', c.complement() )
```

```
a = ['사과', '포도', '망고', '수박', '복숭아']
```

```
b = ['복숭아', '건포도', '오렌지', '사과', '배']

d = MySet()
d.setdata(a,b)

print(d.union())
print(d.intersection())
print(d.complement())
```

```
#숫자의 자리수 출력
def digit(n):
    str_number = str(n) #스트링으로 변환
    return len(str_number) #글자수 리턴

print(digit(1234))
print(digit(12))
print(digit(1234567))
```

```
#숫자 2개를 입력받아 두 숫자에 공통된 숫자를 출력하는 함수
#  1234,  345 => 3과 4가 중복됨
def number_inter(a,b):
    a1 = str(a)
    b1 = str(b)
    c = []
    c = intersection(a1, b1)
    c.sort()
```

```
        return c

a = 1234
b = 3456
c = number_inter(a,b)
print (c)
```

```
# xor 함수 구현(같으면 0, 다르면 1)
# 0 0 => 0
# 0 1 => 1
# 1 0 => 1
# 1 1 => 0

def xor(a,b):
        if a + b == 1:
                return 1
        else:
                return 0

print (xor(0,0))
print (xor(0,1))
print (xor(1,0))
print (xor(1,1))
```

```
#a가 b의 부분집합인지 판별하는 함수
def issubset(a,b):
        a.sort()
```

```
    if intersection(a,b) == a:
        print ('a는 b의 부분집합')
    else:
        print ('a는 b의 부분집합이 아니다.')

a = [1,2]
b = [1,2,3,4,5]
issubset(a,b)

b.remove(2)
issubset(a,b)
```

```
#리스트의 마지막 원소를 맨 앞으로 이동시키는 함수
# [1,2,3,4] => [4,1,2,3]
def rotate(items):
    result = []
    #마지막 요소
    result.append(items[-1])
    #마지막 요소를 제외한 모든 값들
    result = result + items[:-1]
    return result

a = [1,2,3,4]
rotate(a)
```

2. 자연수와 정수, 유리수와 무리수

```
import numpy as np

print (np.sqrt(100))
print (np.sqrt(16))
print (np.sqrt(7))
print (np.sqrt(-4)) #곱해서 음수가 되는 값은 없으므로 nan
```

```
a = 3.7
print(type(a))

b = round(a) #반올림
print(b)
print(type(b))
```

```
a = 3.7
b = int(a)
print(b)
print(float(b)) #정수형을 실수형으로 변환
```

```
a = [1,2,3,4,5]
print(min(a)) #최소값
print(max(a)) #최대값
```

```
# 곱셈
a = np.array([1,2,3])
b = np.array([4,5,6])
print(np.multiply(a,b))
print(np.max(a))
print(np.min(b))
print (np.argmax(a)) #최대값의 인덱스
print (np.argmin(b)) #최소값의 인덱스
```

```
print(np.random.randint(9)) #0~9 미만의 임의의 숫자값
print(np.random.randint(100, 150, size = 5)) #100~150 미만의 5개의 난수
```

```
#1~100 미만의 중복되지 않은 숫자 10개를 만드는 함수
def make_numbers():
    numbers = []
    for i in range(10):
        num = np.random.randint(1,100)
        if num not in numbers:    # 리스트에 없는 숫자이면 추가
            numbers.append(num)
    return numbers

make_numbers()
# 위 코드는 중복값을 제거하여 숫자 10개가 완성되지 않으므로 while로 변경하여 작성
#    while len(numbers)<10:
```

```
#소수 : 1과 자기 자신으로만 나누어지는 숫자  2,3,5,7
#a의 값이 소수인지 판별
a = 5
prime = True # 초기값을 소수로 설정
for i in range(2,a):                    # 2부터 a-1까지
    if a % i == 0:                      # a를 i로 나눈 나머지가 0이면
        prime = False                   # 소수가 아님으로 설정

if prime == True:
    print (a,"==> 소수")
else:
    print (a,"==> 소수가 아님")
```

```
#소수 구하기 함수1
def prime_num(a):
    prime = True # 초기값을 소수로 설정
    for i in range(2,a):                    # 2부터 a-1까지
        if a % i == 0:                      # a를 i로 나눈 나머지가 0이면
            prime = False                   # 소수가 아님으로 설정

    if prime == True:
        print (a,"==> 소수")
    else:
        print (a,"==> 소수가 아님")
```

```
prime_num(10)
prime_num(11)
```

```
#소수 구하기 함수2
def prime_num(a):
    b = range(2, a)  #2부터 a-1까지
    c = 0
    for i in b:
        if a % i == 0:
            c += 1
    if c > 0:
        d = False
    else:
        d = True
    return d

a = range(2,101)                    # 2~100
numbers = []
for i in a:
    c = prime_num(i)                # i가 소수이면 True 아니면 Fa
lse
    if c == True:                   # c가 True이면
        numbers.append(i)  # 소수 리스트에 i를 추가

print(numbers)
```

```
#약수 구하기(약수=인수, 예를 들어 12의 약수(인수)는 : 1,2,3,4,6,12)
a = 10
nums = []
for i in range(1,a):
    if a % i == 0:              # i로 나눈 나머지가 0이면 i 추가
        nums.append(i)

nums.append(a)                  # 마지막 수 추가(자기자신)
print(nums)
```

```
#약수 구하기(함수)
def factor_nums(a):
    nums = []
    for i in range(1,a):
        if a % i == 0:              # i로 나눈 나머지가 0이면 i 추가
            nums.append(i)

    nums.append(a)                  # 마지막 수 추가
    return nums

print(factor_nums(10))
print(factor_nums(15))
```

```
#소인수분해
```

```
# 약수=인수
# 12의 약수(인수) : 1,2,3,4,6,12
# 소인수 : 인수 중에서 소수인 숫자
# 12의 소인수 : 2,3

# 12를 2로 나누면 6이 되고, 6을 2로 나누면 3이 되고, 3을 2로 나누면 1이 되고 1을 2로 나누면 0이 됨
# 12를 3으로 나누면 4가 되고, 4를 3으로 나누면 1이 되고, 1을 3으로 나누면 0이 됨
# 24를 더 이상 나누어지지 않는 수로 계속 나누면?
# 24 = 2 x 2 x 2 x 3

a = 24
primes = []
for i in range(2,a): # 2 ~ a-1
    while a % i == 0:           # a를 i로 나눈 나머지가 0이면 반복
        primes.append(i)        # i를 추가
        a /= i                  # a를 i로 나눈 값

if primes == []:                # 빈값이면 그 수 자체가 소인수
    primes.append(a)

print(primes)
print(list(set(primes))) #중복값을 제거하고 싶을 경우 set 사용
```

```
#최대공약수 구하기
#1.두 수의 약수 구하기
#2.두 약수의 교집합
#3.교집합 중 최대값

# 교집합 함수
def intersection(a,b):
    c = [ ]
    for i in a:
        if i  in b:
            c.append(i)
    return c

#최대 공약수
def max_common_factor(a, b):
    # 약수 구하기
    c = factor_nums(a)          # a의 약수
    d = factor_nums(b)          # b의 약수
    print(c)
    print(d)
    #교집합
    e = intersection(c,d)       # c와 d의 교집합
    print(e)
    #교집합 중 가장 큰 수 리턴
    return max(e)

print(max_common_factor(10,15))
```

```
print(max_common_factor(8,20))
#최소공배수
```

```
def myfunc(a,b):
    t=0
    if a>b:
        t=a
    else:
        t=b

    result=0
    while t: # t가 0이 아니면 반복 처리됨
        if t % a == 0 and t % b == 0: # 최소공배수이면
            result=t #결과를 저장하고
            break #반복문 종료
        t += 1 # t를 증가시킴
    return result

print(myfunc(2,3))
print(myfunc(100,25))
```

```
# 리스트 1,2,3,4를 숫자값 1234로 바꾸기
a = [1,2,3,4]
nums = ""
for i in range(len(a)):
    # str(a[i]) 리스트의 i번째 값을 문자로 바꿈
    nums = nums + str(a[i])
```

```
print(type(nums),nums)
number = int(nums)
print(type(nums),nums)
```

```
# 리스트 1,2,3,4를 숫자값 1234로 바꾸기(함수)
def list_to_num(a):
    nums = ''
    for i in range(len(a)):
        # str(a[i]) 리스트의 i번째 값을 문자로 바꿈
        nums = nums + str(a[i])
    nums = int(nums)
    return nums

list_to_num([1,2,4,5,8,91])
```

```
#피보나치 수열 : 1,2부터 시작하여 마지막 두 수를 더한 수를 그 다음에 놓는 수열
# 1,2,3,5,8
def pibonacci(n):
    nums = [1,2]
    for i in range(2,n):
        nums.append(nums[-1]+nums[-2])
    #print(nums)
    return nums

#10개의 피보나치 수열
```

```
print(pibonacci(10))
```

```
#1~10까지의 난수 5개의 곱
import numpy as np

n = 1
for i in range(5):
    temp = np.random.randint(1,11)
    print(temp,end=" ")
    n *= temp
n
```

```
#유리수와 무리수

#유리수 : 분수로 나타낼 수 있는 수
#  정수, 유한소수, 무한소수 중 분수로 나타낼 수 있는 수
#  유한소수 : 1/2, 3/4처럼 나누어 떨어지는 소수
#  무한소수 : 2/3처럼 나누어 떨어지지 않는 소수

#무리수 : 순환하지 않는 무한소수

from fractions import Fraction

#분수 표현
print (Fraction(5/2))
print (Fraction(2.5))
print (Fraction(5,2))
```

```
#분자
print(Fraction(7,9).numerator)
#분모
print(Fraction(7,9).denominator)
#기약분수 구하기(더이상 약분이 되지 않는 분수)
print(Fraction(2,4))
```

```
#분수의 사칙연산
print(Fraction(1,4) * Fraction(4,7))
print(Fraction(1,4) / Fraction(4,7))
print(Fraction(1,4) + Fraction(4,7))
print(Fraction(1,4) - Fraction(4,7))
```

```
#분수의 제곱
print(Fraction(1,4)**2)
#분수 형태로 출력되므로 float() 함수를 이용하여 실수형으로 변환
a = Fraction(9, 16)
print(float(a))
```

```
#무리수 : 순환하지 않는 무한소수
print(np.sqrt(3))
print(np.pi)
#무리수도 사칙연산이 가능함
print(np.pi * np.sqrt(3))
```

```
#복소수 : 실수부+허수부로 구성된 수
```

```
#파이썬에서는 허수부를 j를 이용하여 표현

#복소수형 상수 대입
c1 = 1+7j
#실수부만 선택
print(c1.real)
#허수부만 선택
print(c1.imag)
#실수부가 2, 허수부가 3인 복소수를 만드는 코드
c2 = complex(2, 3)
print(c2)
```

3. 방정식과 부등식

```
#방정식 : 변수가 특정값일때만 만족하는 식
import numpy as np

def plus(x):
    return x+5

print(plus(7))
print(plus(np.array([1,2,3,4,5])))
```

```
#람다식을 이용한 함수
# lambda 입력값:수행할 명령어
plus = lambda x: x+5
print(plus(7))

a = lambda x, y: x+y
print(a(2,6))
```

```
# 1~10 짝수 판별 람다식
even = lambda x : x % 2 == 0
even(np.array(range(1,11)))
```

```
# 체질량 지수(bmi) : 몸무게(kg) / 키의 제곱(m)
```

```
bmi = lambda weight, height : weight / (height)**2
bmi(68,1.75)
```

```
def fat_or_not(weight, height):
    a = bmi(weight, height)
    print("bmi:",a)
    if a > 35:
        print ('고도비만입니다.')
    elif 30 <= a < 35:
        print ('중등도비만입니다.')
    elif 25 <= a < 30:
        print ('경도비만입니다.')
    elif 23 <= a < 25:
        print ('과체중입니다.')
    elif 18.5 <= a < 23:
        print ('정상입니다.')
    else:
        print ('저체중입니다.')

fat_or_not(68,1.75)
```

```
# filter 함수를 사용하면 람다식의 결과값을 선택할 수 있음

# 1~10 짝수 판별 람다식
even = lambda x : x % 2 == 0
even(np.array(range(1,11)))
```

```
# 1~10 짝수 리스트
list(filter(even, range(1,11)))
```

```
#리스트 중에서 짝수인 값(함수)
nums=[2, 1, 3, 5, 8]

def even(n):
    if n % 2 == 0:
        return True
    else:
        return False

result = []
for i in nums:
    if even(i):
        result.append(i)

print(result)
```

```
#리스트 중에서 짝수인 값(람다식)
nums=[2, 1, 3, 5, 8]
list(filter(lambda x: x % 2 == 0, nums))
```

```
#for 문을 이용한 리스트 만들기
a = [2*x for x in [3,5,6,8,9]]
print(a)
```

```
# 1~10을 3으로 나눈 나머지 값
b = [x % 3 for x in range(1,11)]
print(b)
```

```
#약수 구하기(함수)
def factor_nums(a):
    nums = []
    for i in range(1,a):
        if a % i == 0:          # i로 나눈 나머지가 0이면 i
추가
            nums.append(i)

    nums.append(a)              # 마지막 수 추가
    return nums

# 교집합 함수
def intersection(a,b):
    c = []
    for i in a:
        if i  in b:
            c.append(i)
    return c

#최대 공약수
def max_common_factor(a, b):
    # 약수 구하기
    c = factor_nums(a)          # a의 약수
```

```
    d = factor_nums(b)              # b의 약수
    #print(c)
    #print(d)
    #교집합
    e = intersection(c,d)           # c와 d의 교집합
    #print(e)
    #교집합 중 가장 큰 수 리턴
    return max(e)
```

```
# 10~20까지의 2의 배수 중 15와의 최대공약수를 원소로 갖는 튜플 리스트
c = [(x,max_common_factor(x, 15)) for x in range(10,21,2)]
c
```

```
# for와 if를 사용한 리스트

# 리스트 중에서 짝수인 값에 2를 곱한 리스트
a = [2*x for x in range(1,11) if x % 2 == 0]
a
```

```
#삼각형의 넓이를 구하는 함수
def triangle_area(a,h):
    return 1/2*a*h

a = 5
h = 10
triangle_area(a,h)
```

#높이와 밑변이 1~5까지 변할 때의 삼각형의 넓이를 계산하여 2차원 리스트에 저장

```
a = np.array(range(1,6))
h = np.array(range(1,6))
area = np.zeros((len(a), len(h)))
print(area.shape)

for i in a:
    for j in h:
        area[i-1,j-1] = triangle_area(i,j)

area
```

#람다식으로 작성한 코드

```
triangle_area2 = lambda a, h: 0.5*a*h

triangle_area2(10,5)
```

#해발 100미터 올라갈 때마다 기온이 1도씩 떨어진다고 가정
#경포대 해수욕장의 기온이 38도일 경우, 해발 2000미터의 기온을 구하시오.

```
temp = 38- 1*(2000/100)
temp
```

#일차방정식 : 미지수가 1개인 방정식

2x - 3 = 5

```
x = (5+3)/2
print(x)

# 3x = -7x - 10
# 3x + 7x = -10
# (3+7)x=-10
# x = -10/(3+7)

x = -10/(3 + 7)
print(x)
```

```
#부등식

#x의 원소에 대해 3*1 + 2 <=11 인지 판별하는 부등식 작성

x = [2,3,5,7]
a = [ ]
for i in x:
    if 3*i + 2 <= 11:
        a.append(True)
    else:
        a.append(False)
a
```

```
#for 리스트로 작성
x = [2,3,5,7]
a = [3*i + 2 <= 11 for i in x]
```

```
a
```

```
#부등식을 만족하는 원소만 출력
x = [2,3,5,7]
a = [i for i in x if 3*i + 2 <= 11 ]
a
```

```
# 부등식 4x - 5 > 13

x = [2,3,5,7]
a = [i for i in x if 4*i -5 > 13]
print(a)

# 부등식 2x + 1 < 15
a = [i for i in x if 2*i + 1 < 15]
print(a)
```

```
# x의 원소 : -5~5
# 부등식 x + 5 >= 4
# np.arange(start,stop,step)
x = np.arange(-5,6)
b = lambda a: a + 5 >= 4
print(b(x))

#위 부등식을 만족하는 값들만 출력
c = list(filter(b, x))
print(c)
```

```
# 부등식 x - 4 <= -3
n = np.arange(-10,11,1)
y = list(filter(lambda a: a - -4 <= -3, n ))
y
```

```
# 부등식 x + 5 >= 4
a = []
for i in x:
    if i + 5 >= 4:
        a.append(i)
print (a)
```

```
# -5에서 5까지의 정수
# 부등식 : 3x - 7 <= 3

a = []
for i in range(-5,6):
    if 3*i - 7 <= 3:
        a.append(i)
a
```

```
[i for i in range(-5,6) if 3*i - 7 <= 3]
```

```
list(filter(lambda i: 3*i -7 <= 3, range(-5,6,1)))
```

#연립부등식 : 일차 부등식 2개를 묶어서 한 쌍으로 나타낸 것

어떤 정수의 두 배에서 3을 빼면 11보다 작고, 이 정수의 세 배에서 6을 빼면 9보다 크다. 이 때의 정수를 구하시오.
원소 : -10~10

#교집합 함수

```
def intersection(a,b):
    c = []
    for i in a:
        if i  in b:
            c.append(i)
    c.sort()
    return c


x1 = list(filter(lambda a: 2*a - 3 < 11,   np.arange(-10,11,1)))
x2 = list(filter(lambda a: 3*a - 6 > 9,   np.arange(-10,11,1)))

intersection(x1,x2)
```

4. 일차함수

```
a = [1,2,3]
b = ['a','b','c']
c = ['딸기', '복숭아', '참외']
d = list(zip(a,b,c))
d
```

```
# zip 함수를 직접 코딩으로 작성

def my_zip(a,b):
    if len(a) != len(b):
        print('두 입력값의 길이가 다릅니다.')
    else:
        n = len(a)
        c = []
        for i in range(n):
            c.append((a[i],b[i]))
        return c

a = [1,2,3]
b = ['a','b','c']
my_zip(a,b)
```

```
#산점도
%matplotlib inline
import matplotlib.pyplot as plt

x = 1
y = 2
plt.scatter(x,y)
```

```
x = [1,2,3,4,5]
y = [2,4,6,8,10]
plt.scatter(x,y)
```

```
#포인트 사이즈 변경
plt.scatter(x,y, s = 100)
```

```
plt.scatter(x,y, s = 100, c = 'g') # green
```

```
plt.scatter(x,y, s = 100, c = 'g', alpha = 0.5) #투명도 조절
```

```
plt.plot(x,y) #선그래프
```

```
plt.plot(x,y, linewidth = 4)
```

```
plt.plot(x,y, c = 'r', linewidth = 4)
```

```
plt.plot(x,y, c = 'r', linewidth = 4, alpha = 0.5)
```

```
#산점도와 선 그래프
plt.scatter(x,y, c = 'g', s = 100)
plt.plot(x,y, c = 'g', alpha = 0.5)
```

```
#그리드 표시
plt.scatter(x,y, c = 'g', s = 100)
plt.plot(x,y, c = 'g', alpha = 0.5)
plt.grid()
```

```
#그래프 사이즈
plt.figure(figsize = (5,5))
plt.scatter(x,y, c = 'g', s = 100)
plt.plot(x,y, c = 'g', alpha = 0.5)
plt.grid()
```

```
plt.figure(figsize = (5,5))
plt.scatter(x,y, c = 'g', s = 100)
plt.plot(x,y, c = 'g', alpha = 0.5)
plt.grid()
#x,y축의 범위
plt.xlim([0,6])
plt.ylim([0,12])
```

```
plt.figure(figsize = (5,5))
#label 범례에 표시할 문자열
plt.scatter(x,y, c = 'g', s = 100, label = 'dot graph')
plt.plot(x,y, c = 'g', alpha = 0.5, label = 'line graph')
```

```
plt.grid()
plt.legend() #범례
```

```
plt.figure(figsize = (5,5))
plt.scatter(x,y, c = 'g', s = 100, label = 'dot graph')
plt.plot(x,y, c = 'g', alpha = 0.5, label = 'line graph')
plt.grid()
plt.legend()
#x,y축의 눈금 텍스트의 사이즈,색상 조절
plt.xticks(fontsize = 14, c = 'b')
plt.yticks(fontsize = 25, c = 'y')
```

```
plt.figure(figsize = (5,5))
plt.scatter(x,y, c = 'g', s = 100, label = 'dot graph')
plt.plot(x,y, c = 'g', alpha = 0.5, label = 'line graph')
plt.grid()
plt.xticks(fontsize = 14, c = 'b')
plt.yticks(fontsize = 25, c = 'y')
plt.legend(fontsize = 20,  loc = 4)#범례 폰트 사이즈,위치 조절
```

```
plt.figure(figsize = (5,5))
plt.scatter(x,y, c = 'g', s = 100, label = 'dot graph')
plt.plot(x,y, c = 'g', alpha = 0.5, label = 'line graph')
plt.grid()
plt.xticks(fontsize = 14, c = 'b')
plt.yticks(fontsize = 25, c = 'y')
```

```
plt.legend(fontsize = 20,   loc = 4)
plt.title('y = 2x', fontsize = 30) #그래프의 타이틀
```

```
plt.figure(figsize = (5,5))
plt.scatter(x,y, c = 'g', s = 100, label = 'dot graph')
plt.plot(x,y, c = 'g', alpha = 0.5, label = 'line graph')
plt.grid()
plt.xticks(fontsize = 14, c = 'b')
plt.yticks(fontsize = 25, c = 'y')
plt.legend(fontsize = 20,   loc = 4)
plt.title('y = 2x', fontsize = 30)
#x,y축 라벨
plt.xlabel('Sun light', c = 'y', fontsize = 19)
plt.ylabel('Apple sweet', c = 'm', fontsize = 30)
```

```
plt.xlim([0,10])
plt.ylim([0,10])
plt.grid()
#x축 수직선
plt.axvline(x = 5, c = 'r', lw = 10)  # x = 5에  빨간선 긋기
#y축 수평선
plt.axhline(y = 2, c = 'b')  # y = 2에 파란색 수평선 긋기
```

```
import numpy as np
# y=ax

x = np.arange(-10,10,1)
```

```
#일차함수의 기울기 2, -2
for a in [2, -2]:
    plt.plot(x, a*x, label = 'y = {0}x'.format(a))

plt.grid()
plt.legend(fontsize = 15, loc = 8)
```

#위 그래프에서는 x축보다 y축의 범위가 2배 더 큰 것이 반영이 되지 않고 있습니다. 아래와 같이 figsize를 조절하여 표현해 보겠습니다.

```
x = np.arange(-10,10,1)
plt.figure(figsize = (5,10))
for i in [0.5,  0, 2,  8]: #기울기
    plt.plot(x, i*x, label = f'y = {i}x')

plt.xticks(np.arange(min(x),max(x)+1,1))
plt.grid()
plt.legend(fontsize = 14, loc = 4)
```

```
x = np.arange(-10,10,1)
plt.figure(figsize = (5,10))
#음수의 기울기
for i in [-1/2,  0, -2,  -8]:
    plt.plot(x, i*x, label = 'y = {0}x'.format(i))

plt.grid()
```

```
plt.legend(fontsize = 14, loc = 3)
```

```
#일차함수의 상수항(절편)

x = np.arange(-10,10)
b = np.arange(-5, 5, 2)
for i in b:
    plt.plot(x, x + i, label = f'y = x + {i}')

plt.grid()
plt.legend(fontsize = 13)
```

```
# y = a / x 그래프

a = 1
x = np.arange(-10,11, 1)
plt.figure(figsize = (5,8))
plt.scatter(x, a/x)
plt.axvline(x = 0, c = 'r') #수직선
plt.axhline(y = 0, c = 'r') #수평선
plt.grid()
```

```
# y=3x + 2 그래프
# x의 범위 : -5 ~ 10

x = np.arange(-5,11)
y = 3*x + 2
```

```
plt.plot(x,y)
plt.legend(fontsize = 14)
plt.grid()
```

```
# y = |x|   절대값 x
# x의 범위 : -10~10

x = np.arange(-10,11)
y = np.abs(x)
plt.plot(x,y)
plt.grid()
```

```
# y=x (x>=0)
# y=0 (x<0)
# x의 범위 : -10~10

x = np.arange(-10,11)
y = np.zeros(len(x))
for i in range(len(x)):
    if x[i] >= 0:
        y[i] = x[i]

plt.plot(x,y)
plt.grid()
```

```
#y=1 ( x>=0 )
#y=0 ( x<0 )
```

```
#x의 범위 : -10~10

x = np.arange(-10,11)
y = np.zeros(len(x))
for i in range(len(x)):
    if x[i] >= 0:
        y[i] = 1

plt.plot(x,y)
plt.grid()
```

```
# x의 범위 : 1~100까지의 자연수
# y = x의 약수의 갯수

def factor_nums(a):
    nums = []
    for i in range(1,a):
        if a % i == 0:          # i로 나눈 나머지가 0이면 i
추가
            nums.append(i)

    nums.append(a)              # 마지막 수 추가
    return nums

x = np.arange(1,101)
y = []
for i in x:
```

```
        nums = factor_nums(i)
        y.append(len(nums))

    plt.plot(x,y)
    plt.grid()
    plt.xlim(0,105)
    plt.ylim(0,15)
```

```
    # x의 범위 : -10~10
    # y=3x + 5
    # z=-2y - 10

    x = np.arange(-10, 10)
    y = 3*x + 5
    z = -2*y - 10
    plt.plot(x,z)
    plt.grid()
```

```
    # 화씨온도(y) = 섭씨온도(x) x 9/5 + 32
    # x의 범위 : -20 ~ 40

    from matplotlib import font_manager, rc
    #한글 처리를 위해 폰트 설정
    font_name = font_manager.FontProperties(fname="c:/Windows/
Fonts/malgun.ttf").get_name()
    rc('font', family=font_name)
```

```
x = np.arange(-20,41)
y = x*9/5 + 32
plt.plot(x,y)
#음수 부호의 글꼴이 깨질 경우
plt.rcParams['axes.unicode_minus'] = False
plt.xlabel('섭씨', fontsize = 14)
plt.ylabel('화씨', fontsize = 14)
plt.grid()
```

```
# 임의의 정수들을 입력받아 모두 더한 값이 0보다 크면 더한 값을 출력하고
# 작으면 0을 출력하는 함수

def func1(*nums):
    result = np.sum(nums)
    if result < 0:
        result = 0
    return result

print(func1(3,4,5,6,7))

print(func1(-9,3,5,4,-6))
```

```
# y=2**x (2의 x승)
# x의 범위 : -10 ~ 10

x = np.arange(-10,11)
```

```
y = 2.0**x
plt.plot(x,y)
plt.grid()
```

```
# y = x의 제곱근
# x의 범위 : 0~10

x = np.arange(0,11,0.1)
y = np.sqrt(x)
plt.plot(x,y)
plt.grid()
```

```
# 1달러=1110원이고 1위안=0.15달러일 때
# 1위안은 몇원입니까?

# 1달러:1110원=0.15달러:x원
# x=1110 x 0.15

rate = 0.15*1110
print ('1위안은 {0}원이다.'.format(rate))
```

```
#은행 예금 이율이 3%라고 할 때
# 금액=원금x이율**연수
# 1년 기다리면 10000x1.03
# 2년 기다리면 10000x(1.03)의 제곱
# 3년 기다리면 10000x(1.03)의 세제곱 이라고 할 때 함수 작성
```

```
def saving(money, years, rate):
    result = money*(1+rate/100)**years
    return result
```

#만원을 3%, 5%, 7%이율로 예금하였을때 10년이 될때까지 매년 총액수를 함수로 그리시오.

```
money = 10000
years = np.arange(0,11)
rate = [3, 5, 7]
total_money = np.zeros((len(rate), len(years)))
for i in range(len(rate)):
    for j in range(len(years)):
        total_money[i,j] = saving(money, years[j], rate[i])
    plt.plot(years, total_money[i,:], label = '{}%'.format(rate[i]))

plt.legend(fontsize = 14)
plt.grid()
plt.xlabel('Years', fontsize = 14)
plt.ylabel('Total money', fontsize = 14)
```

```
# y=x
# x의 범위 : -5~5 난수 발생

x = np.arange(0,21)
a = np.random.randint(low = -5, high = 5, size = len(x))
```

```
y = x + a
plt.plot(x, y)
```

#한 부부가 서울에서 5억짜리 주택을 구입하고자 한다.
#월급이 400만원이고 식비/생활비로 100만원, 자녀 교육비 50만원을 쓰고 남은 돈으로 집을 사려고 할 때
#몇 년 몇개월이 걸릴지 계산하시오.

```
price = 500000000
saving = 4000000-1000000-500000

n = price / saving
year = n // 12
month = n % 12
print ('{0}년 {1}개월'.format(year, month))
```

#저축액이 100만원 ~ 1000만원일 때 몇 개월만에 5억 주택을 구입할 수 있는지 플로팅하시오.

```
price = 50000
saving = np.arange(100, 1100, 100)
n = price / saving
print(list(zip(saving,n)))
plt.plot(saving, n)
plt.grid()
plt.xlabel("저축")
plt.ylabel("개월수")
#100만원 500개월, 1000만원 50개월
```

#a는 20세부터 매년 1000만원을 벌고, b는 27세부터 매년 2000만원을 벌고, c는 36세부터 4000만원을 번다면
#0~60세 범위의 수입에 대하여 플로팅하시오.

```
years = np.arange(0,61,1)

a = (years-20)*1000
b = (years-27)*3000
c = (years-36)*10000

a[:20] = 0
b[:27] = 0
c[:36] = 0

plt.plot(years, a, label = 'a')
plt.plot(years, b, label = 'b')
plt.plot(years, c, label = 'c')
plt.grid()
plt.legend(fontsize = 14)
```

5. 확률과 통계

#도수 분포표

```
import pandas as pd
import numpy as np

#체중 데이터
weight = [22, 24, 26, 30, 32, 40, 35, 45, 20, 29, 34, 36, 3
6, 38, 39, 48, 43, 37, 33, 31, 29, 39, 26, 29]
# 체중구간 20~24, 25~29, 30~34, 35~39, 40~44, 45~50
hist = np.zeros(6)
for i in weight:
    if i//5 == 4:
        hist[0] += 1
    elif i//5 == 5:
        hist[1] += 1
    elif i//5 == 6:
        hist[2] += 1
    elif i//5 == 7:
        hist[3] += 1
    elif i//5 == 8:
        hist[4] += 1
    elif i//5 == 9:
        hist[5] += 1
```

```
print(hist)
```

```
#pandas의 시리즈는 행 인덱스를 자유롭게 지정할 수 있음
index = ['20~25', '25~30', '30~35', '35~40', '40~45', '45~50']
a = pd.Series(hist, index = index)
print(a)

print(type(a))

#시리즈에 이름 지정
a.name = 'A반의 체중 도수분포표'
print(a)
```

```
print(len(weight))
print(weight[:5])
height = [124, 125, 128, 130, 134, 140, 131, 143, 122, 129, 136, 139, 141, 135, 142, 150, 149, 141, 127, 131, 130, 125, 135, 126]
d = {'weight' : weight, 'height' : height}
e = pd.DataFrame(d)
print(e.head())
print(e['weight'])
```

```
f = np.zeros((len(weight), 2))
f[:,0] = weight
```

```
f[:,1] = height
g = pd.DataFrame(f, columns = ['weight', 'height'])
print(g.head())
g.name = 'A반의 몸무게와 키'
print(g.name)
```

```
#도수분포
import numpy as np

weight = [22, 24, 26, 30, 32, 40, 35, 45, 20, 29, 34, 36, 3
6, 38, 39, 48, 43, 37, 33, 31, 29, 39, 26, 29]
bins = np.arange(20,55,5)  # 도수분포구간
print(bins)

#도수, 구간 = np.histogram(data, 도수분포구간(bin))
#도수 : 각 구간의 빈도수
hist, b = np.histogram(weight, bins)
print('도수:',hist)
print('구간:',b)
```

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.hist(weight, bins)
plt.xlabel('Weight (kg)', fontsize = 14)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
```

```
plt.hist(weight, bins, rwidth = 0.8) # rwidth 막대 폭 조절
plt.xlabel('Weight (kg)', fontsize = 14)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
```

```
plt.hist(weight, bins, rwidth = 0.8, color = 'green')
plt.xlabel('Weight (kg)', fontsize = 14)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
```

```
plt.hist(weight, bins, rwidth = 0.8, color = 'green', alpha = 0.
5)
plt.grid()
plt.xlabel('Weight (kg)', fontsize = 14)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
```

```
#경우의 수 찾기

import itertools

#주사위를 두번 던질 때의 경우의 수(순서가 중요하고 중복
이 허용되지 않는 경우)
#permutations 순열
#permutations(범위, 횟수)
event = list(itertools.permutations(range(1,7), 2))
print(len(event)) #경우의 수
```

```
print(event)
```

#주사위를 두번 던질 때의 경우의 수(순서가 중요하고 중복이 허용될 경우)

```
#product 순열
event = list(itertools.product(range(1,7), range(1,7)))
print(len(event)) #경우의 수
print(event)
```

```
#순서는 중요하지 않고, 중복은 허용 안 될 때
# (1,2) (2,1) 하나로 처리
list(itertools.combinations([1,2,3,4], 2))
```

```
#순서는 중요하지 않고 중복이 허용될 때

list(itertools.combinations_with_replacement([1,2,3,4], 2))
```

#주사위를 두번 던져 합이 5의 배수가 될 경우의 수 및 확률을 구하시오.

```
from fractions import Fraction

nums = []
for i in range(1,7):
    for j in range(1,7):
        if (i + j) % 5 == 0 :
            nums.append((i,j))
```

```
print(nums)

n = len(nums)
p = Fraction(n,36)
print(n)
print(p)
print(float(p))
```

```
#1~1000까지의 정수에서 3과 5의 배수들의 합계
x = np.arange(3,1001,3) #3의 배수
y = np.arange(5, 1001, 5) #5의 배수
z = np.arange(15,1001,15) #3과 5의 공배수인 15의 배수
total = sum(x) + sum(y) - sum(z)
print(total)
```

#500원 동전 2개, 100원 동전 3개, 50원 동전 4개, 10원 동전 9개 있을 때
이 조합으로 만들 수 있는 금액의 경우의 수를 구하시오.(0원 제외)

```
import numpy as np

a500 = np.arange(0,1100,500)
a100 = np.arange(0,310,100)
a50 = np.arange(0,210,50)
a10 = np.arange(0,91,10)
```

```
result=[]
for a in a10:
    for b in a50:
        for c in a100:
            for d in a500:
                if a+b+c+d>0:
                    result.append(a+b+c+d)


r=list(set(result))
r.sort()
print(len(r))
print(r)
```

#크기가 다른 주사위 2개를 동시에 던져 나온 숫자를 각각 x, y라고 할 때 다음 물음에 답하시오.

1) x+y <= 6 이 될 확률

2) x-y = 3 이 될 확률

3) y = x-2, y > -x + 7을 동시에 만족하는 확률

#크기가 다르므로 순서가 중요하고 중복이 가능하므로 product를 사용함

#모든 경우의 수는 36

```
from fractions import Fraction

event1 = 0
```

```
event2 = 0
event3 = 0
for x in range(1,7):
    for y in range(1,7):
        if x+y <= 6:
            event1 += 1
        if x-y == 3:
            event2 += 1
        if (y == x-2) and (y > -x + 7):
            event3 += 1

a = Fraction(event1,36)
b = Fraction(event2,36)
c = Fraction(event3,36)
print(a)
print(b)
print(c)
```

```
#평균
x = [2,3,4,5,6,6,7,8,8,10]
def average(x):
    return sum(x)/len(x)

print(average(x))

average2 = lambda a:sum(a)/len(a)
print(average2(x))
```

```
print(np.average(x))
```

```
#중앙값(median)
x = [9,3,5,2,7,2,6,6,7,7,8,8,10]
x.sort()
print(x)

n = len(x)
middle_number = n//2

print ('원소개수 : {}'.format(n))
print ('median index: {0}번'.format(middle_number))
print ('중앙값 : {0}'.format(x[middle_number]))

print(np.median(x))
```

```
x = np.array([9,3,5,2,7,2,6,6,7,7,8,8,10])
print(x.argmax()) #최대값의 인덱스
print(x.argmin()) #최소값의 인덱스
print(x[x.argmax()])
print(x[x.argmin()])
```

```
#최빈값
x = [9,3,5,2,7,2,6,6,7,7,8,8,10]

# 0이 0개, 1이 0개, 2가 2개, 7이 3개...
```

```
bins=np.bincount(x)
print(bins)
#가장 큰 수는 3, 3의 인덱스는 7번
idx=bins.argmax()
print(idx)
```

```
x = [9,3,5,2,7,2,6,6,7,7,8,8,10]
frequency = {}
max_n = 0
for i in x:
    frequency[i] = x.count(i)  # 숫자별로 count

for j in frequency:
    print (j,frequency[j])
    if frequency[j] > max_n:
        max_n = frequency[j]
        max_key = j

print ('최빈수 : ',max_key)
```

```
print(np.bincount(x))

#argument 중에서 가장 큰 값
print(np.bincount(x).argmax())
```

```
#분산과 표준편차
#편차 = 변량 - 평균
```

```
#분산 = 편차를 제곱한 값들의 평균
#표준편차 = 분산의 제곱근
x = [0, 1, 3, 6, 12, 13, 10, 7, 5, 1]

mean = np.average(x)
print(mean)
print(np.var(x))
print(np.std(x))
```

```
variance = 0
for i in range(len(x)):
    variance += (x[i]-mean)**2 #(변량 - 평균)의 제곱

variance /= len(x)
std= np.sqrt(variance)

print ('분산 : ', variance)
print ('표준편차 : ',std)
```

#주사위를 두번 던져서 처음 나온 숫자를 x, 두번째 나온 숫자를 y라고 할 때 x-y의 분산은?

```
x = np.arange(1,7)
y = np.arange(1,7)
#print(x)
#print(y)
```

```
x_minus_y = []
for i in x:
    for j in y:
        x_minus_y.append(i-j)
        #print(i,j,i-j)

#print(x_minus_y)
print(np.mean(x_minus_y))
print(np.var(x_minus_y))
```

#아래와 같은 2개의 히스토그램을 보고 평균과 표준편차가 큰 쪽을 고르시오.

```
bin = np.arange(-120, 120, 10)
#normal(평균,표준편차,샘플갯수) 정규분포 난수
hist1 = np.random.normal(0, 30, 1000)
hist2 = np.random.normal(10, 20, 1000)
plt.hist(hist1, bin, alpha = 0.5,rwidth=0.8, color="blue",label='hist1')
plt.hist(hist2, bin,alpha = 0.5,rwidth=0.8, color="yellow",label='hist2')
plt.legend()
plt.grid()

print(np.mean(hist1))
print(np.mean(hist2)) #평균값이 더 크다
print(np.std(hist1)) #표준편차가 더 크다
```

```
print(np.std(hist2))

#평균은 yellow가 더 크고
#표준편차는 blue가 더 크다
```

6. 2차방정식과 2차함수

```
#2차 방정식

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-1, 7) # -1~6
# y=5x - x제곱
y = 5*x - x**2
print(x)
print(y)
plt.plot(x,y)
plt.grid()
plt.axhline(y=0, c='r') #수평선
plt.axvline(x=0, c='r') #수직선
# y가 0이 되는 x는 0과 5
```

```
#x: -2,-1,0,1,2

x = np.arange(-2,3,1)
# x제곱 + x - 2 = 0의 해를 구하시오.
```

```
for i in x:
    if i**2 + i - 2 == 0:
        print (i)
```

```
# x제곱 - 3x - 7 = 0
# x : -5 ~ 5

x = np.arange(-5,6,1)
print(x)
y = x**2 - 3*x - 7
print(y)
for i in x:
    print ('x = {}, y = {}'.format(i, i**2 - 3*i - 7))
plt.plot(x, y)
plt.grid()
# 0이 되는 해가 존재하지 않음
```

```
x = np.arange(-1.6,-1.5,0.01)
y = x**2 - 3*x - 7
#for i in x:
#    print ('x = {}, y = {}'.format(i, i**2 - 3*i - 7))
plt.plot(x, y)
plt.grid()

plt.figure()
x = np.arange(4.4,4.7,0.01)
y = x**2 - 3*x - 7
```

```
#for i in x:
#      print ('x = {}, y = {}'.format(i, i**2 - 3*i - 7))
plt.plot(x, y)
plt.grid()
# -1.54와 4.55에 가까움, arange를 더 촘촘히 하는 방법을 사용함, 근사화 과정
```

```
#2차함수
y=x^2
%matplotlib inline
import matplotlib.pyplot as plt

# y=x제곱

x = np.arange(-5,6) # -5~5

y = x**2
print(x,y)
plt.plot(x,y)
plt.grid()

#기울기가 0보다 크면 아래쪽으로 휘는 그래프
x = np.arange(-5,6) # -5~5
y = x**2
y2 = 2*x**2 # 기울기 2, x의 제곱
y3 = 0.5*x**2 # 기울기 0.5, x의 제곱
plt.plot(x,y3, label = 'a = 0.5')
```

```
plt.plot(x,y, label = 'a = 1')
plt.plot(x,y2, label = 'a = 2')
plt.legend(fontsize = 14)
plt.grid()
```

```
#기울기가 0보다 작으면 위쪽으로 휘는 그래프

y2 = 2*x**2
y3 = 0.5*x**2
plt.plot(x,-y3, label = 'a = -0.5')
plt.plot(x,-y, label = 'a = -1')
plt.plot(x,-y2, label = 'a = -2')
plt.legend(fontsize = 14)
plt.grid()
```

```
#이차함수 y=a(x-p)^2 + q 그래프
#y=a * x^2 그래프를 x축으로 p만큼, y축으로 q만큼 이동한 그래프
#(x,y)=(p,q)의 좌표를 지날 때 최대 혹은 최소가 됨
x = np.arange(6) #0~5
y = 2*(x-3)**2 + 4
print(x,y)
plt.plot(x,y)
plt.scatter(3,4, s = 100, color = 'r')
plt.grid()
```

#y=-3x^2 을 x축으로 1만큼, y축으로 -2만큼 이동한 그래프

```
#y=-3(x-1)^2 - 2
x = np.arange(-1,5) # -1~4
y = -3*(x-1)**2 - 2
print(x,y)
plt.plot(x,y)
plt.scatter(1,-2, s = 100, color = 'r')
plt.grid()
```

```
#이차함수 y=a*x^2 + b*x + c 그래프
#-5x^2 + 9x + 2
x = np.arange(-5,6) # -5 ~ 5
y = -5*x**2 + 9*x + 2
print(x,y)
plt.plot(x,y)
plt.grid()
```

```
x = np.arange(2,11) # 2~10
y = x + np.sqrt(x-2)
print(x,y)
plt.plot(x,y)
```

1. 집합
2. 자연수와 정수, 유리수와 무리수
3. 방정식과 부등식
4. 일차함수
5. 확률과 통계
6. 2차방정식과 2차함수