반올림 함수 round(실수,표시될 자리수)

파이썬에서 실수끼리의 합 등은 정상적으로 표기 안될 수 있으므로 반올림 할것.

슬라이싱

리스트에서 일부분만 가져올때

a[1:4] 1번 인덱스~3번인덱스(마지막은 +1 되어있음)

리스트 컴프리헨션

a = [i for i in range(10)]

하면 0~9까지 값이 입력된다.

array = [0 for i in range(10)]

print(array)

#0부터 19까지 홀수만 포함하는 리스트

array = [i for i in range(20) if i%2==1]

print(array)

n=4

m=3

#2차원 리스트 초기화

array = [[0]\*m for \_ in range(n)]

print(array)

#array[[0]\*m]\*n 하면 큰일남!

#반복문에서 변수값을 무시하고자 할 때 \_사용

#단순 5번 반복

for \_ in range(5):

print("hello")

리스트 메소드



문자열은 변경 불가능한 자료형

집합 자료형은 중복x, 순서x

데이터가 존재하는지 안하는지 여부만을 확인할 때 유용

리스트나 문자열을 통해 초기화 가능 Set() 함수 사용

중괄호 안에 콤마를 기준으로 구분삽입하여 초기화 가능

({ , , ,})

초기화 예시 중복이 자동으로 제거된다

Data = set([1,2,3,4,4,5])

Data = {1,2,3,4,4,5}

집합 연산을 제공한다. 집합 a,b에 대해

합집합

A|b

교집합

A&b

차집합

a-b

집합에 새 원소 추가 .add(n), 여러 값 추가 .update(1,2,3,4), 특정값 제거 .remove(n)

표준 입력 방식

Input() 한 줄의 문자열을 입력받는다. (정수형을 원하는 경우 int(input()) 으로 받는다)

Map()함수는 리스트의 모든 원소에 각각 특정한 함수를 적용할 때 사용

예시

공백을 기준으로 구분된 데이터를 입력받을 때

List(map(int,input.split()))

개수가 적으면

A,b,c = map(int,input.split())

Split은 ()안의 문자를 기준으로 문자열을 나눈다(공백인 경우 공백)

여기서 map은 input.split으로 공백을 기준으로나누어진 문자들을 int형으로 바꾸는 역할을 한다.

반대로 리스트를 문자열로 만드는 ‘’,join(list)이 있다. 리스트 원소간 ‘’문자를 추가하여 더한다

문자열 빠르게 입력 받기

Import sys

Data = sys.stdin().readline().rstrip()

출력

Print는 ,를 통해 띄어쓰기로 구분하여 출력 가능

기본적으로 줄바꿈 출력인데 end를 수정하여 변경가능

Print(1,2, end=” “)

Print(1,2) 하면 1,2, 1,2 가 된다(print 끝나고 나서 출력이 공백으로 바뀜)

Fstring

f스트링은 3.6이후부터 가능하며

answer = 7

print(“정답은”+str(answer)+”입니다”) 를

print(f“정답은 {answer}입니다”) 로 간단하게 할 수 있다.

//앞에 f와 변수 사용시 {}

논리 연산자

X and Y

X or Y

not X

in 과 not in 연산자

리스트,튜플,문자열,딕셔너리 등 다수의 자료를 갖는 유형에 모두 사용 가능

X in 자료형

X not in 자료형

X가 자료형에 있는지 없는지 반환(T/F)

Pass 키워드

아무것도 처리하고 싶지 않을 때 사용 (예: 조건문 중 조건 통과 이후 미구현 등에서)

파이썬에서 x<20 and x>0 을 0<x<20 이 가능하다.

파이썬에서 함수 밖의 변수를 전역처럼 가져와서 쓰려면

함수 내에서 global 키워드를 붙여야 한다

Global a 등

단. 값을 수정하는 것이 아닌 단순 참조의 경우에는 그냥 a를 써도 된다.

또한 리스트 등의 내장함수 등은 global 키워드 없이 쓸 수 있다(수바 뭔차이여)

파이썬은 함수가 여러 개의 반환값을 가질 수 잇다

Return a,b,c,d

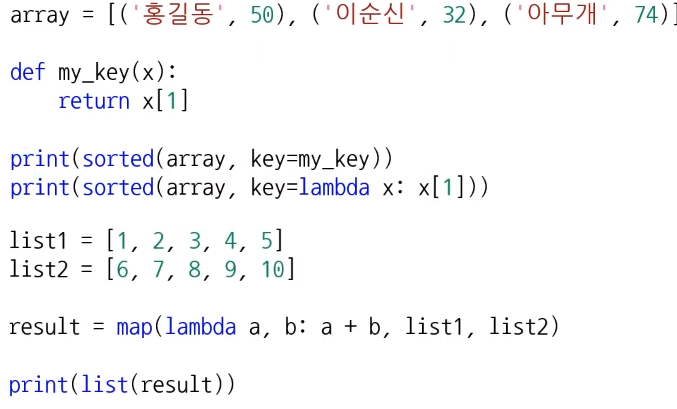
A,b,c,d = 함수()

람다표현식

함수를 입력으로 받는 함수를 사용할 때, 함수가 간단하거나 한번만 쓸 때 사용.

표현식

Print((lambda a,b:a+b)(3,7))



람다식 사용 예시

유용한 라이브러리

Intertools

반복되는 형태의 데이터 처리에 유용한 기능기능

순열과 조합 라이브러리가 포함되어있고, 모든 경우의 수(브루트포스?)에 유용

Heapq

힙 자료구조를 제공

우선순위 큐 기능을 구현하기 위해 사용

Bisect

이진탐색 기능 제공

Collections

덱, 카운터 등의 유용한 자료구조 포함

Math

필수적인 수학적 기능 제공

팩토리얼,제곱근,최대공약수,삼각함수부터 파이 같은 상수 제공

자주 사용하는 내장함수

Sum, min,max, eval

Eval 은 문자열로 된 식을 실제 숫자로 계산되어 반환해줌

순열

서로다른 n개에서 서로다른 r개를 선택하여 일렬로 나열하는 것

A,B,C 의 경우 ABC,ACB,CBA,CAB,BCA,BAC

조합

서로 다른 n개에서 순서에 상관없이 서로다른 r개를 선택 하는 것

A,B,C의 경우 AB,BC,AC

순열의 수 nPr = n\*(n-1)…(n-r+1)

조합의 수 nCr = n\*(n-1)…(n-r+1)/r! = nPr/r!

모든 순열 구하기

From intertools import permutations

Data = [‘A’,’B’,’C’,’D’]

Result = list(permutations(data,3)) //4P3

From intertools import combinations

Data = [‘A’,’B’,’C’]

Result = list(combinations(data,2)) //3C2

중복 순열과 중복 조합

From intertools import product

Data = [‘A’,’B’,’C’’]

Result = list(product(data,repeat=2))//2개뽑는 모든 순열, 중복허용

From intertools import combinations\_with\_replacement

Data = [‘A’,’B’,’C’]

Result = list(combinations\_with\_replacement(data,2)) //2개뽑는 모든 조합, 중복허용

Counter 라이브러리

From collections import Counter

Counter = Counter([‘red’,’blue’,’red’,’green’,’blue’,’blue’])

Print(counter[‘blue’])//blue가 등장한 횟수

Print(counter[‘green’])

Print(dict(counter)) //사전형으로 반환

{‘red’:2, ‘blue’:3,’green’:1}

최대 공약수

Import math

Print(math.gcd(21,14))//최대공약수 gcd계산

최소공배수

Print(21\*14//math.gcd(21,14))

기타 내장함수

.isalpha, .isdigit

둘다 문자열을 검사하는 내장함수로 true,false를 반환하며

Isapha는 해당 문자열이 영어인지, isdigit은 해당 문자열이 숫자로만 되어있는지를 반환한다.

스택과 큐

스택은 리스트를 그대로 사용하고 .append와 .pop을 사용하면 된다.

큐의 경우 리스트로 구현이 가능하지만 pop이후 순서 조정 등 시간 복잡도 때문에

Deque를 import하여 사용하는게 좋다,

From collections import deque

Q = deque()

q.append(2)

q.leftpop

q.reverse() 등이 가능하다.

2차원 리스트입력(숫자)

G=[]

G.append(list(map(int,input())))

Bisect 라이브러리

From bisect import bisect\_left, bisect\_right

bisect\_left(arr,x) 배열 arr 에서 x가 들어갈수 있는 왼쪽인덱스 반환

bisect\_right(arr,x) 배열 arr에서 x가 들어갈 수 있는 오른쪽 인덱스 반환

(왼쪽 오른쪽인 이유는 중복때문)

이를 응용하여 특정 범위의 값들의 수를 구할 수 있고, 중복수의 개수를 구할 수 도 있다.