자료형

type()으로 알 수 있음.

문자 string: “Hello”

number:

bool: True, False

“안녕하세요” 표현시

‘ “안녕하세요” ‘

“\”안녕하세요\”” 둘 다 가능

문자열 연결시

“ “+” “

숫자

abs(), pow, max, min, round(반올림)

from math import\*

print(floor(4.99)) #올림, 4

print(ceil(3.14)) #내림, 4

print(sqrt(16)) #4.0

from random import\*

print(random()) #[0.0 1.0) , 0.8914755733658388

print(int(random()\*45)) #[0 45) \*괄호 위치 유의

print(int(random()\*45)+1) #[1 46) ,32

print(randint(1,45)) #[1 45]

슬라이딩

a= '980623'

print(a[0:2]) #98 \*0~1까지

print(a[2:]) #0623 \*2~끝까지

print(a[-4:]) #0623 \*끝에서 4만큼 자름

문자열 처리 (공백포함이 디폴트)

a= 'I am happy'

print(a.lower()) #소문자

print(a.upper()) #대문자

print(a[3].isupper()) #한글자 대문자

print(a[1].isupper()) #\*공백부분, False출력

print(len(a)) #공백포함, 10

print(a.replace("happy","good")) #바꿔 출력

a= 'I am happy'

print(a.index("a")) # a위치, 2

print(a.index("a",2))

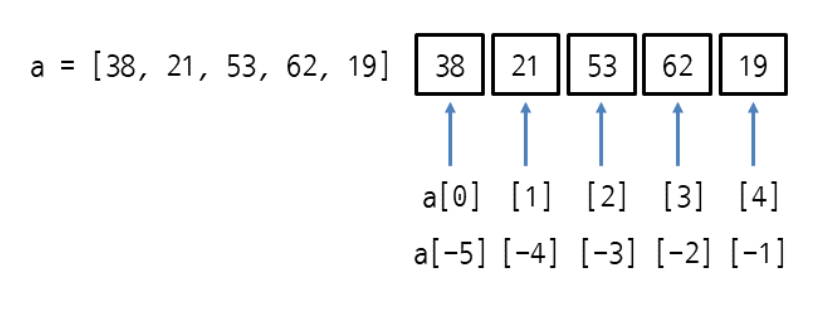
#find와 index의 차이

print(a.find("sad")) #-1 반환

print(a.index("sad")) #오류내며 종류

print(a.count("a")) #등장 횟수

인덱스



* IndexError인 경우, 범위에 없는 문자나 범위를 넘는 인덱스를 지시하는지 확인.

문자열 연산자

#문자열 연산자

print("I am"+" happy") #문자열끼리 연산자만 허용

print("happy"\*3) #문자열 반복

I am happy

happyhappyhappy

arr=[1,2,3,4,5]

arr.sort() #내림차순 정렬

arr.reverse() #순서 뒤집기

자료형

print(type(3))

print(type(3.14))

<class 'int'>

<class 'float'>

c++과 달리 파이썬은 자료형을 미리 지정하지 않아 유동적으로 바꾸기 가능

a=문자

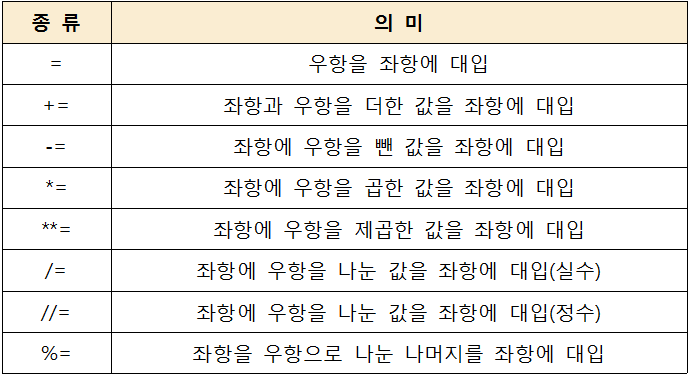
a=True

a=10

사칙연산

제곱은 \*\*

복합 대입 연산자



* //: 정수 나누기 연산자. (소수점 떼버림)
* print(3/2) #1.5
* print(3//2) #1
* TypeError : 서로 다른 자료를 연산하면. 문자열+숫자 이런 식으로,

.

입출력

입력

-사용자가 int, float, T/F를 입력해도 무조건 string(문자열)로 받아들임.

printf("이름:%s",a)

a=input("이름") #파이썬

cast(캐스트): 문자열을 숫자로 바꾸기

* int(“52”): 문자열을 int로
* float(“52.5”): 문자열을 float로
* str(52) : int, float를 문자열로

VlaueError: 숫자가 아닌걸 숫자로 변환하려 할 때. int(“a”)

맞는 함수를 쓰지 않을 때. int(“52.5”)

format함수

숫자를 문자열로

* “{}”.format(52) : 숫자를 문자열로 (여러 개도 가능)
* a="{} {} {}".format(3000, 4000, 5000)
* print(a)

3000 4000 5000

print("{:05d}".format(52)) #0붙여 출력, 00052

print("{:+d}".format(52)) #+- 출력, +52 -52

print("{: d}".format(52)) #+는 공백, -만 출력.



\*IndexError: {}개수와 매개변수 개수가 안맞을 때

문자열 다루기

print("{:g}".format(52.0)) #의미없는 소수점 제거, 52

#문자열 공백 제거

a=" 안녕 "

print(a.strip()) #'안녕'

print(a.lstrip()) #'안녕 '

print(a.rstrip()) #' 안녕'

#문자열 구성 파악하기: isOO()

b="hello"

print(b.isalnum()) #T / 알파벳or숫자로만 구성?

print(b.islower()) #T /소문자로만 구성?

print(b.isspace()) #F /공백으로만 구성?

#문자열 찾기 (시작부분 기준)

c="안녕안녕하세요"

print(c.find("안녕")) #0

print(c.rfind("안녕")) #2 /오른쪽 찾기

#문자열에 포함하는지 T/F

print(c in("혜진")) #F

#문자열 자르기 split()

aaa="10 20 30 40 50"

print(aaa.split(" ")) #공백을 기준으로 자른다

#['10', '20', '30', '40', '50']

날짜 시간

import datetime

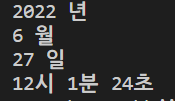
a = datetime.datetime.now()

print(a.year,"년")

print(a.month,"월")

print(a.day,"일")

print("{}시 {}분 {}초".format(a.hour,a.minute,a.second))



if 조건문

if 조건:

elif 조건:

else :

#오전오후 구분

import datetime

a = datetime.datetime.now()

if a.hour<12:

    print("오전")

if a.hour>=12:

    print("오후")

#홀짝구분

a=input("정수: ")

a=int(a) #정수로 변환해야 에러 안뜸

if a%2==0:

    print("짝")

else:

     print("홀")

빈 컨테이너 (None, 숫자0, 0.0)은 false로 처리한다.

pass는 미구현 상태라고 선언하기 위해 #오전오후 구분

import datetime

a = datetime.datetime.now()

if a.hour<12:

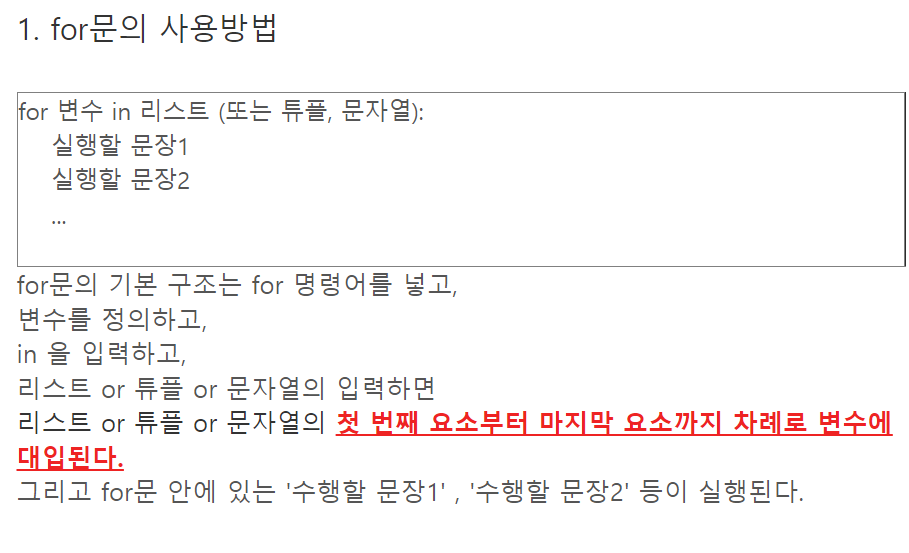
    print("오전")

if a.hour>=12:

    pass #아직 미구현 상태

반복문

for



#20점 넘기면 통과

arr=[10, 15, 30, 40]

i=0

for ck in arr:

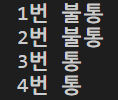
    i+=1

    if ck>20:

        print("%d번 통" %i)

    else:

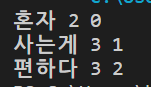
        print("%d번 불통" %i)



arr=["혼자", "사는게", "편하다"]

for a in arr:

    print(a, len(a), arr.index(a))



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

for i in arr:

    print("{}".format(i))



while문

customer="혜진"

i=5

while i>=1:

    print("{},남은시간은{}초 입니다".format(customer,i))

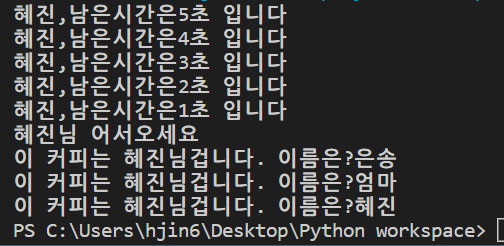
    i-=1

print("{}님 어서오세요".format(customer))

p="?"

while p!=customer:

    p=input("이 커피는 혜진님겁니다. 이름은?")



#모든 배열에 100 더하기

arr =[1,2,3,4,5]

arr=[i+100 for i in arr]

print(arr) #[101, 102, 103, 104, 105]

#이름을 길이로 변환

arr=["Jack","May"]

arr=[len(i) for i in arr]

print(arr) #[4, 3]

#이름을 대문자로 변환

arr=["Jack","May"]

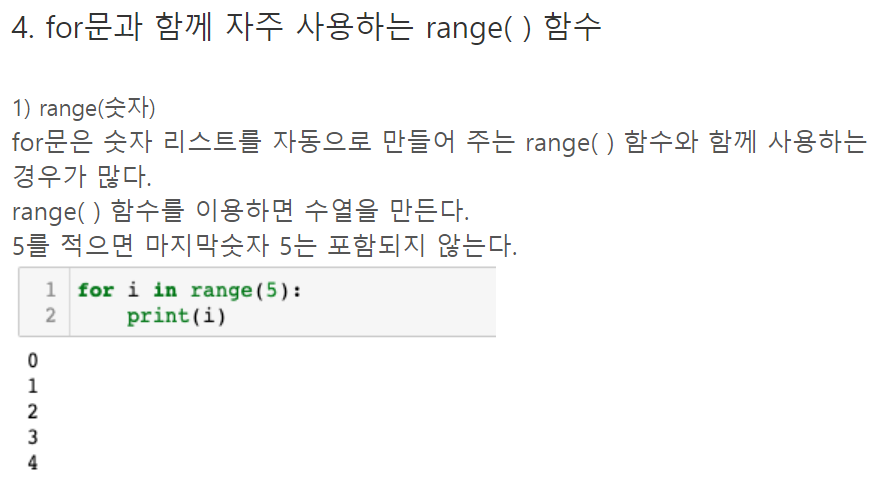
arr=[i.upper() for i in arr]

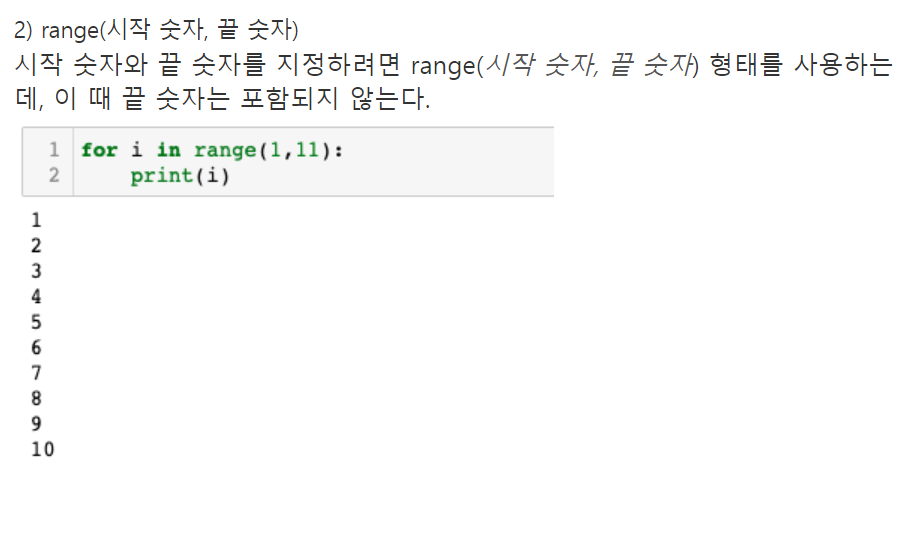
print(arr) #['JACK', 'MAY']

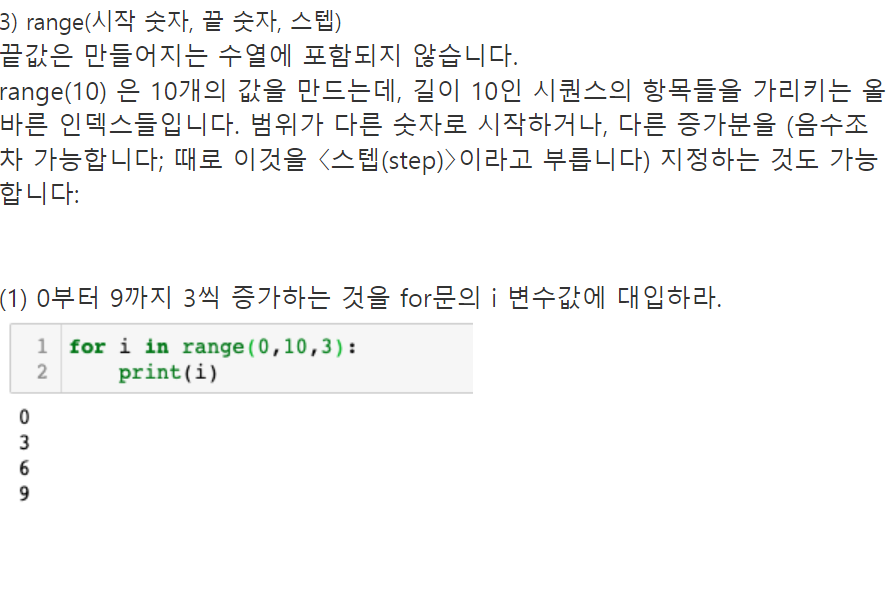
continue 넘어가기

break반복문 빠져나가기

range 함수 (C++이랑 비슷한 반복문 표현방법이네)







리스트에 요소 추가하기

마지막에 요소 추가

리스트.append(요소)

중간에 요소 추가

리스트.insert(위치, 요소)

중간에 여러 요소 추가

리스트.extend(리스트)

arr=[1, 2, 3]

arr.append(4)

print (arr) #[1, 2, 3, 4]

arr=[1, 2, 3]

arr.append("4")

print (arr) #[1, 2, 3, ‘4’]

arr=[1, 2, 3]

arr.insert(2,0)

print (arr)

# [1, 2, 0, 3]

arr=[1, 2, 3]

arr.extend([4,5])

print (arr)

요소를 추가하는 방법은 2가지, extend와 연산자

요소를 빼는 방법은 리스트.pop()

#1. 파괴적(원본을 바꿈)

arr=[1, 2, 3]

arr.extend([4,5])

print (arr)

#[1, 2, 3, 4, 5]

#2. 비파괴적(원본 유지)

arr=[1, 2, 3]

brr=[4,5,6]

print(arr+brr)

#[1, 2, 3, 4, 5, 6]

리스트 비우기

arr.clear() #[]

리스트 내부에 있는지/없는지 확인

arr=[1,2,3]

print(2 in arr) #True

print(4 not in arr) #True

출력

#방법1

print("나이:%d,이름:%s,%c로 시작" %(20,"이나연","이"))

#방법2

print("나이:{}, 이름{}, {}로 시작".format(20,"이나연","이"))

#방법3

age=20

name="이나연"

print(f"나이:{age}, 이름:{name}")

딕셔너리

키는 3, 이은송은 변수

딕셔너리={키: 이름}

키는 정수, string도 가능

cabinet = {3:"이은송", 100: "엄마"}

print(cabinet[3])

cabinet = {3:"이은송", 100: "엄마"}

print(cabinet[3])

print(cabinet.get(3))

#없는 키를 호출할 때 차이점

print(cabinet[5]) #오류남

print(cabinet.get(5)) #None 출력

print(cabinet.get(5,"0")) #값이 없으면 0 출력

print(3 in cabinet) #T

cabinet = {3:"이은송", 100: "엄마"}

#추가 (뒤쪽에 추가됨)

cabinet[1]="이혜진"

#삭제

del cabinet[3]

#key만/value만 출력

print(cabinet.keys())

print(cabinet.values())

#('1','이혜진') 이런 식으로 출력

print(cabinet.items())

#dict\_items([(100, '엄마'), (1, '이혜진')])

#비우기

cabinet.clear

튜플 (리스트와 다르게 내용 변경/추가 불가능, 속도 빠름)

변경되지 않는 것을 사용할 때 유용

menu=("돈까스","치즈까스")

print(menu[0])

#추가/삭제가 안됨

#튜플X

name="이혜진"

age=20

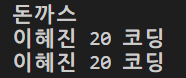
hobby="코딩"

print(name, age, hobby)

#튜플사용시 (짧아짐)

name, age, hobby = "이혜진",20,"코딩"

print(name, age, hobby)



집합(set)

#집합(set) 중복안됨, 순서없음

#표현방법 2개

arr={1,2,3,3} #{1,2,3}출력

brr=set([3,5])

print(arr&brr) #교집합 {3}

print(arr|brr) #합집합 {1,2,3,5}

print(arr-brr) #차집합 {1,2}

#추가/제거

arr.add(4)

arr.remove(4)

자료구조 변경은

list(), tuple(), set()로 가능

치킨커피 뽑기

from random import\*

arr=list(range(1,21)) #1~20까지

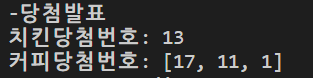
shuffle(arr)

win=sample(arr,4) #4명 뽑기

print("-당첨발표")

print("치킨당첨번호: {}".format(win[0]))

print("커피당첨번호: {}".format(win[1:]))



친구 10명, 90점 이상만 내 친구

from random import\*

cnt=0

for i in range(1,11): #1~10

    score=randrange(1,101)

    if score>=90:

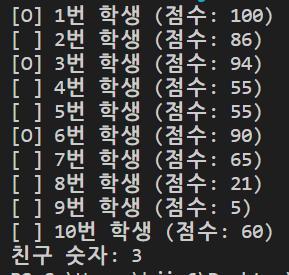
        print("[O] {}번 학생 (점수: {})".format(i,score))

        cnt+=1

    else:

        print("[ ] {}번 학생 (점수: {})".format(i,score))

print("친구 숫자: {}".format(cnt))



파이썬

2798 곱셈

-int()를 꼭 씌워주기. 아니면 문자열로 해석함.

num=int(input())

for i in range(1,10):

    print("{}\*{}={}".format(num,i,num\*i))

2884

-and

h,m=map(int,input().split()) #띄어쓰기로 입력받음

if m>44:

    print(h,m-45)

elif m<45 and h>0:

    print(h-1,m+15)

else:

    print(23,m+15)

2480 주사위

a,b,c=map(int,input().split()) #띄어쓰기로 입력받음

re=0

if a==b==c:

    re=10000+a\*1000

elif a==b:

    re=1000+a\*100

elif b==c:

    re=1000+b\*100

elif a==c:

    re=1000+c\*100

else:

    re=max(a,b,c)\*100

print(re)

본격적으로 for문 문제를 풀기 전에 주의해야 할 점이 있다. 입출력 방식이 느리면 여러 줄을 입력받거나 출력할 때 시간초과가 날 수 있다는 점이다.

C++을 사용하고 있고 cin/cout을 사용하고자 한다면, cin.tie(NULL)과 sync\_with\_stdio(false)를 둘 다 적용해 주고, endl 대신 개행문자(\n)를 쓰자. 단, 이렇게 하면 더 이상 scanf/printf/puts/getchar/putchar 등 C의 입출력 방식을 사용하면 안 된다.

Java를 사용하고 있다면, Scanner와 System.out.println 대신 BufferedReader와 BufferedWriter를 사용할 수 있다. BufferedWriter.flush는 맨 마지막에 한 번만 하면 된다.

Python을 사용하고 있다면, input 대신 sys.stdin.readline을 사용할 수 있다. 단, 이때는 맨 끝의 개행문자까지 같이 입력받기 때문에 문자열을 저장하고 싶을 경우 .rstrip()을 추가로 해 주는 것이 좋다.

또한 입력과 출력 스트림은 별개이므로, 테스트케이스를 전부 입력받아서 저장한 뒤 전부 출력할 필요는 없다. 테스트케이스를 하나 받은 뒤 하나 출력해도 된다.

15552

t=int(input())

for i in range(t):

    a,b=map(int,input().split())

    print(a+b)

맞지만 시간 초과.

import sys

t=int(input())

for i in range(t):

    a,b=map(int,sys.stdin.readline().split())

    print(a+b)

import sys 를 선언하고

input()대신 sys.stdin.readlin()

import sys

n=int(input())

a\_list=list(map(int,input().split())) #선언한 만큼 리스트에 넣기

Max=max(a,b,c) <<<여기서 틀림!

re\_list=[]

for i in a\_list:

    re\_list.append(i/Max\*100) #a리스트의 값 하나하나 계산해서 b리스트에 넣기

re=sum(re\_list)/n

print(re)

배열중 가장 큰 값

Max=max(a\_list)

IndentationError: expected an indented block after 'if' statement on line 8

들여쓰기 오류

    for i in a\_list[1:]:

        if i > avg :

            cnt +=1

    re=cnt/a\_list[0]\*100

    print (f'{rate:.3f}%')

a\_list리스트 안에 있는 수 하나하나 비교하기

for \_ in range(a\_list[0])이라 적었었는데 이것보다 더 쉬운 방법이..

-sum(list)/list[0]인데 잘못썼었당~

배열을 공백 기준으로 한번에 입력받고싶을 때 arr=list(map(int,input().split()))

소수출력시 f 꼭 붙이기!

t=int(input())

for \_ in range(t):

    a\_list=list(map(int, input().split())) #배열에 다 넣어버리고 인덱스 1부터 계산

    avg=sum(a\_list[1:])/a\_list[0]

    cnt=0

    for i in a\_list[1:]:

        if i > avg :

            cnt +=1

    re=cnt/a\_list[0]\*100

    print ("{:.3f}%".format(re)), # 소수점 세 자리까지만 표현

3052

-실시간으로 입력받아 배열 저장하는 법~

-서로 다른 나머지의 개수를 구하는 것인데, 집합은 중복을 줄인다.

따라서 set()을 이용하면 수월

arr=[]

for \_ in range(10):

   n=int(input())

   arr.append(n%42)

arr=set(arr) #집합(set)은 중복을 줄여버림

print(len(arr))

8958 문자열에서 O몇 개인지 세기

t=int(input())

for \_ in range(t):

    arr=list(map(str,input())) <<input 기본형 str이어서 없어도 ㄱㅊ

    cnt=0

    sum=0

    for i in arr:

        if i=='O':

            cnt+=1

            sum+=cnt

        else:

            cnt=0

    print(sum)

for i in arr: arr[0]~arr[끝]의 요소들을 흝겠다

for \_ in range(9): 9번 반복

for i in range(9): 9번 반복, i는 점점 커질 것

print(cnt)

동빈나 그리디

문제1)

1번과 2번 조건 중 2번을 우선시해야함.

\*그리디인 이유: 최대한 많이 나누기. (2번이 수를 빠르게 줄일 수 있음)

n,k=map(int,input().split())

cnt=0

while True:

    if n%k==0:

        n=n//k

        cnt+=1

    else :

        n-=1

        cnt+=1

    if n==1:

        break

print(cnt)

구현

구현 유형의 문제:풀이를 떠올리는 건 쉽지만 코드로 옮기기 어려운 문제

1. 알고리즘은 간단한데 코드가 넘 길어짐
2. 실수 연산을 다루고, 특정 소수점 자리까지 출력해야 함.
3. 문자열 슬라이싱 (lv1,2로 출제 많이 됨)
4. 적절한 라이브러리를 찾아서 사용해야 하는 문제

파이썬에선 행렬>>리스트라고 부르고

2차원 리스트는

for I in range(5)

for j in

시뮬레이션 및 완전 탐색 문제는 방향 벡터가 자주 활용됨.

상하좌우:

-int ()잊지말기~타입에러남

-x가 위아래, y가 좌우임

-(1,1)부터 시작한다고 하면 방법 2개 1)(0.0)은 사용안하기 2)(1.1)을 (0.0)처럼 해석

1번이 훨 쉬운것같음.

띠용)자꾸 원하는 답이 안나옴 ㅅㅂ 와이라노

n=int(input())

x,y=1,1

plans=input().split()

dx=[0,0,-1,1]

dy=[-1,1,0,0]

move\_types=['L','R','U','D']

nx,ny=0,0

for p in plans:

    for i in range(len(move\_types)):

        if p==move\_types[i]:

            nx=x+dx[i]

            ny=y+dy[i]

        #공간을 벗어나는 경우

        if x<1 or y<1 or x>n or y>n:

            continue

        x,y=nx,ny

print(nx,ny)

완전탐색 시계

\*문자열로 바꿔서 찾기.

#완전탐색: 가능한 모든 경우의 수를 탐색

n=int(input())

cnt=0

for i in range(n+1): #범위에 +1해주는것!

    for j in range(60):

        for k in range(60):

            if '3' in str(i)+str(j)+str(k):

                cnt+=1

print(cnt)

아스키코드 반환 함수

1. 문자->정수

ord(‘a’)넣으면 정수 97반환

1. 정수->문자

인자(정수)의 유효 범위는 0 ~ 1,114,111 (16진수 0x10 FFFF)까지 입니다.

1. 출처: <https://blockdmask.tistory.com/544> [개발자 지망생:티스토리]

chr(97)넣으면 문자 ‘a’반환

#나이트 위치 입력 'a1'

data= input()

row=int(data[1])

column=int(ord(data[0]))-int(ord('a')) +1

#나이트 이동 가능 8방향

steps=[(-2,-1),(-1,-2),(1,-2),(2,-1),(2,1),(1,2),(-1,2),(-2,1)]

re=0

for i in steps :

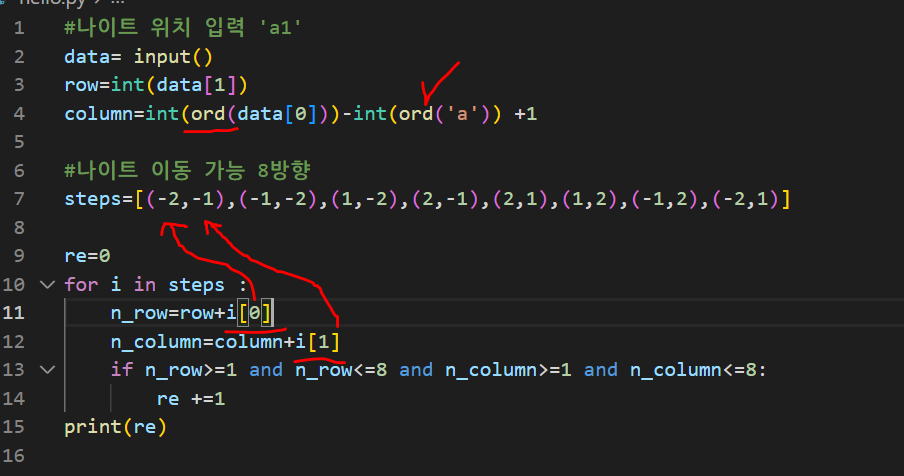
    n\_row=row+i[0]

    n\_column=column+i[1]

    if n\_row>=1 and n\_row<=8 and n\_column>=1 and n\_column<=8:

        re +=1

print(re)



일케 바꿔도 되는듯

 if 1<=n\_row<=8  and 1<= n\_column<=8:

코테

N=int(input())

cnt=0

for i in range(N+1):

    if '1' in str(i):

        continue

    if'2'in str(i) or '5'in str(i):

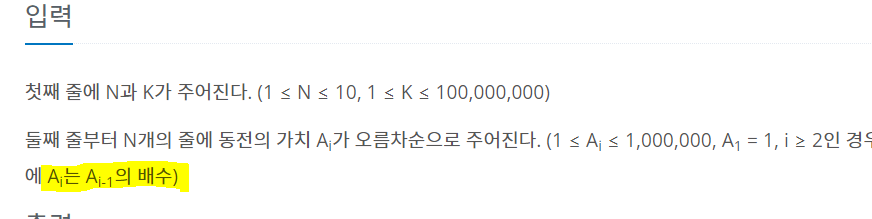
        cnt+=1

print(cnt)

데이터 분석에 json 필수네.. ㅅㅂ

백준11047 그리디

최소 동전 개수인데, 왜 그리디 문제냐면

아래 조건 때문이다. 큰 동전이 전부 작은 동전의 배수가 되기 때문에 그리디 알고리즘이 사용 가능하다. 

n,k = map(int,input().split())

c\_list=list()

for i in range(n):

    c\_list.append(int(input())) #input옆에 괄호 빼먹지 말자..

cnt=0

for i in reversed(range(n)): #역순으로 돌리는 함수

    cnt+=k//c\_list[i]

    k=k%c\_list[i]

print(cnt)

-input() 빼먹지 말자.

-i 끝부터 역순으로 돌리기~

배열 n개수만 주어지고, 줄바꿈하며 입력할 때

c\_list=list()

for i in range(n):

    c\_list.append(int(input()))

15596번: n개가 주어졌을 때 n개의 합을 구하는 함수

1)sum 함수

def solve(a):

    return sum(a)

2)for 이용한 함수(오래걸림)

def solve(a):

    re=0

    for i in a:

        re+=i

    return re

**for [변수] in [문자열, 리스트, 튜플]:**

**[수행부분]**

이런식으로 구성이 되어있습니다.

간단히 설명하자면,

**1. for 반복문**은 in 뒤에 존재하는 **문자열, 리스트, 튜플을 순서대로 순회**하면서**하나씩 하나씩 [변수]에 넣어줍니다**.

예를들어 [1,2,3,4,5] 라는 리스트가 왔다고 했을때

순서대로 [변수]에 1이 들어간 상태로 [수행부분]들을 진행하고.

모든 [수행부분]이 끝나면 다시 for 반복문으로 돌아와서 [변수]에 2를 넣어주고, 아래 [수행부분]을 수행하고

...

이런식으로 진행을 3, 4, 5하다가 5번을 마지막으로 "아 이제 더이상 순회할게 없네. 리스트가 끝났네"하면

for 반복문을 빠져나오게 됩니다.

출처: <https://blockdmask.tistory.com/379> [개발자 지망생:티스토리]

1931 회의실 배정

빨리 **끝나는 회의 순**으로 배정해야 최대한 많이 고려해볼 수 있음

-끝나는 시간이 같다면 빨리 시작하는 순으로 정렬해야 함

2 2

1 2

일대 1 2를 먼저 택해야 함.

1순위) 끝나는 시간 오름차순 2순위)시작 오름차순

key에 튜플로 여러 인자를 주면 해당 인자의 순서대로 정렬해준다

아니면 []리스트 안에 [[1,2] ,[2,3], [5,7] ]

람다(lambda)

-인공지능 분야나 오토캐드 설계 프로그램에서 쓰이는 Lisp언어에서 물려받음.

함수를 한 줄로 편하게 만들어주는 녀석

**lambda** 매개변수 : 표현식

다음은 두 수를 더하는 함수입니다.

>>> **def** **hap**(x, y):

... **return** x + y

...

>>> hap(10, 20)

30

이것을 람다 형식으로는 어떻게 표현할까요?

>>> (**lambda** x,y: x + y)(10, 20)

30

**map()**

먼저 map 함수를 볼까요?

map(함수, 리스트)

이 함수는 함수와 리스트를 인자로 받습니다. 그렇죠? 그리고, 리스트로부터 원소를 하나씩 꺼내서 함수를 적용시킨 다음, 그 결과를 새로운 리스트에 담아준답니다. 말이 좀 복잡하죠? 그럴 때 예제를 보는 게 최고죠.

>>> map(**lambda** x: x \*\* 2, range(5)) # 파이썬 2

[0, 1, 4, 9, 16]

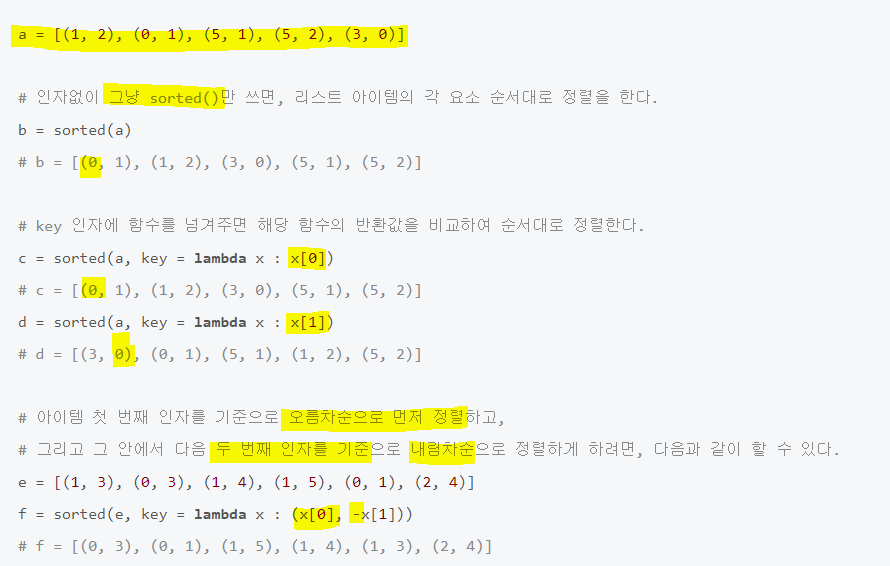
>>> list(map(**lambda** x: x \*\* 2, range(5))) # 파이썬 2 및 파이썬 3

[0, 1, 4, 9, 16]

위의 map 함수가 매개변수로 받은 함수는 lambda x: x \*\* 2구요, 리스트로는 range(5)를 받았습니다. range 함수는 알고계시죠? range(5) 라고 써주면 [0, 1, 2, 3, 4]라는 리스트를 돌려줍니다. 그리고 x \*\* 2 라는 것은 x 값을 제곱하라는 연산자죠.

map 함수는 리스트에서 원소를 하나씩 꺼내서 함수를 적용시킨 결과를 새로운 리스트에 담아주니까, 위의 예제는 0을 제곱하고, 1을 제곱하고, 2, 3, 4를 제곱한 것을 새로운 리스트에 넣어주는 것입니다.

<https://wikidocs.net/64>



sorted(a,key=lambda x:x[0])

리스트 안 fist부분을 기준으로 정렬한다.

\*끝나는 시간 비교는 제일 끝에 넣어주어야 한다

2차원 리스트에서 반복문 사용하기 1)

먼저 for 반복문을 한 번만 사용하는 방식입니다.

>>> a = [[10, 20], [30, 40], [50, 60]]

>>> for x, y in a:    # 리스트의 가로 한 줄(안쪽 리스트)에서 요소 두 개를 꺼냄

...     print(x, y)

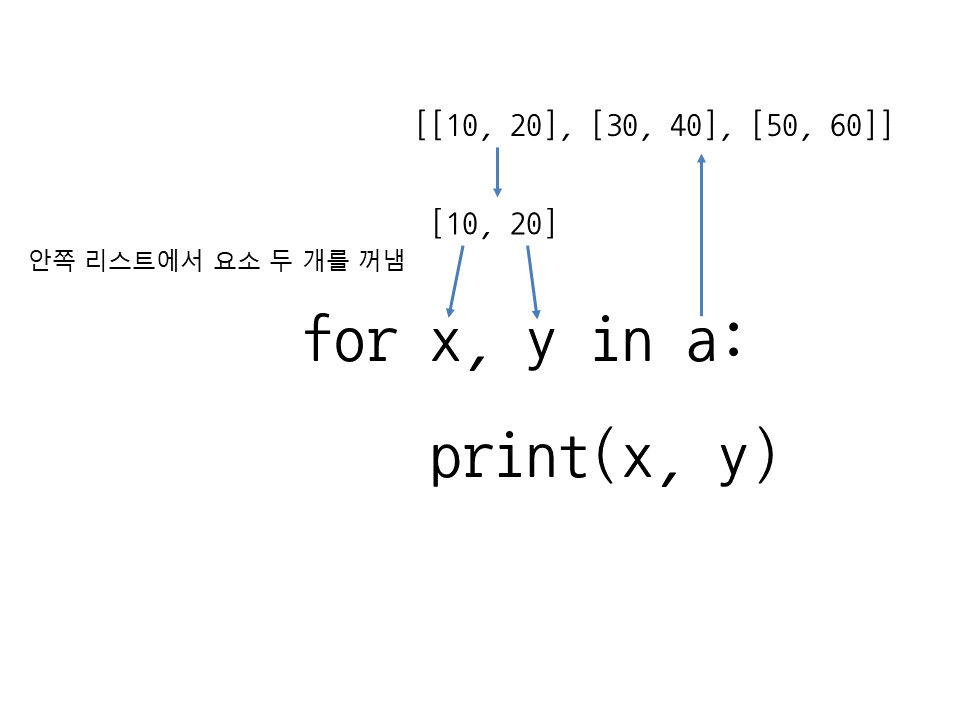
...

10 20

30 40

50 60

for x, y in a:와 같이 in 앞에 변수를 두 개 지정해주면 가로 한 줄(안쪽 리스트)에서 요소 두 개를 꺼내 옵니다.

▼ **그림 23-3** 2차원 리스트에서 for 반복문을 한 번만 사용

당연히 in 앞에 변수의 개수는 2차원 리스트에서 가로 크기(안쪽 리스트의 요소 개수)와 일치해야 합니다. 특히 for 반복문을 한 번만 사용하는 방식은 2차원 리스트의 가로 크기가 크지 않을 때 유용합니다

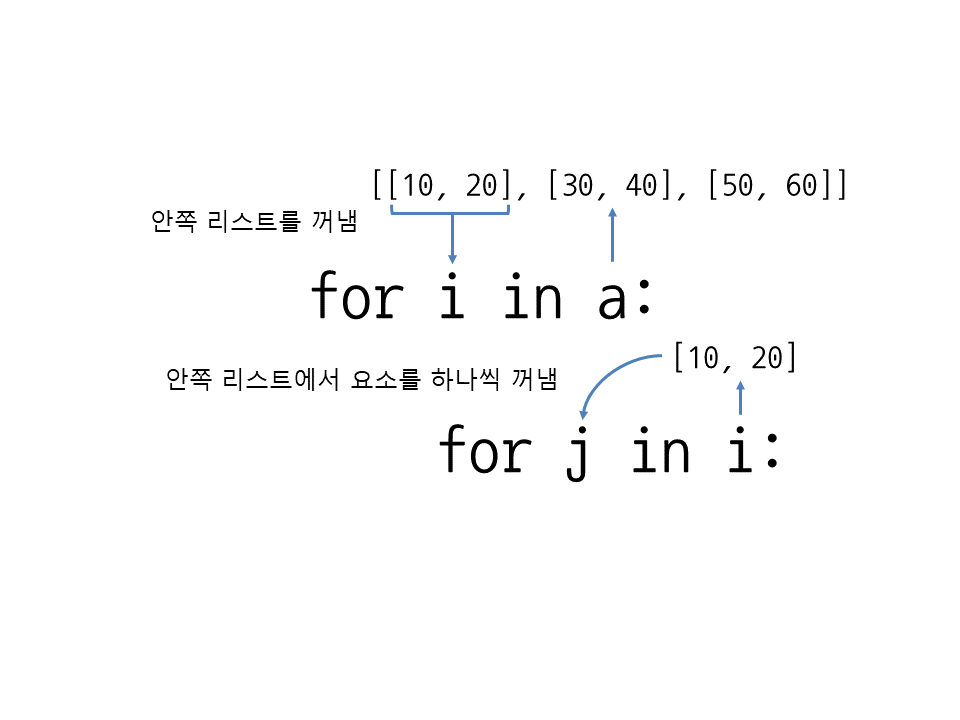
2)

for i in a:        # a에서 안쪽 리스트를 꺼냄

    for j in i:    # 안쪽 리스트에서 요소를 하나씩 꺼냄

        print(j, end=' ')

    print()

▼ **그림 23-4** 2차원 리스트에서 for 반복문을 두 번 사용

a = [[10, 20], [30, 40], [50, 60]]

for i in range(len(a)):            # 세로 크기

    for j in range(len(a[i])):     # 가로 크기

        print(a[i][j], end=' ')

    print()

그 밖의 출력방법은.. <https://dojang.io/mod/page/view.php?id=2292>

#1931

from platform import java\_ver

n=int(input())

s=[]

for i in range(n):

    first, second=map(int,input().split()) <<공백 기준으로 변수2개 대입

    s.append([first,second]) <<2차원 리스트로 넣기

s=sorted(s,key=lambda x:x[0])

s=sorted(s,key=lambda x:x[1]) #중요한 마지막시간 비교는 제일 끝에

cnt=0

last=0

for i,j in s:

    if i>=last: #2차원 배열에서 요소[1]의 last와 요소[2]의 first를 비교

        cnt+=1

        last=j

print(cnt)

2차원 리스트를 정렬하고, for문 두 개 쓰는 법을 알았다

lambda x:x[0]이라는 것은 x가 인자고 x[0]이 함수 내용인 것이다

즉 x(=x[0])를 기준으로 정렬한다는 뜻이다 (key=lambda x:x[0])

## 1. 파이썬 정렬 sorted 함수 정리

**sorted(정렬할 데이터)**

**sorted(정렬할 데이터, reverse 파라미터)**

**sorted(정렬할 데이터, key 파라미터)**

**sorted(정렬할 데이터, key 파라미터, reverse 파라미터)**

출처: <https://blockdmask.tistory.com/466> [개발자 지망생:티스토리]

**key 옵션 (key 파라미터)**  
sorted 함수의 key 파라미터는 어떤 것을 기준으로 정렬할 것인가? 에 대한 기준입니다.  
즉, key 값을 기준으로 비교를 하여 정렬을 하겠다는 것인데, 이것을 정해 줄 수 있는 파라미터입니다.  
sorted( ~~ , key=뭐뭐)로 입력하게 되면 해당 키를 기준으로 정렬하여 반환합니다.

출처: <https://blockdmask.tistory.com/466> [개발자 지망생:티스토리]

11399

인출하는데 오래걸리는 사람은 뒤로 가야 시간합이 최소가 된다

즉 오름차순으로 정렬한뒤 계산하면 됨.

n=int(input())

s=list(map(int,input().split()))

s=sorted(s) #s.sort() 가능

cnt=0

sum=0

for i in s:

    cnt+=i

    sum+=cnt

print(sum)

리스트 정렬 함수

오름차순

s.sort() : 숫자뿐만 아니라 대소문자(ABCD다음 abcd, 아스키코드 순)

내림차순

s.sort(reverse=True)

1541: 잃어버린 괄호

입력은 식. 식이 주어지고 슬라이싱 하는 문제.

-를 기준으로 나눠 리스트a를 만든다.(**split()쓰면 자동으로 리스트형으로 바뀐다.즉, a는 리스트형이다.** join()으로 문자열 리스트를 다시 합칠 수 있다.)

sum

a=input().split('-')

sum=0

for i in a[0].split('+'):

    sum+=int(i)

for i in a[1:]:

    for j in i.split('+'):

        sum-=int(j)

print(sum)

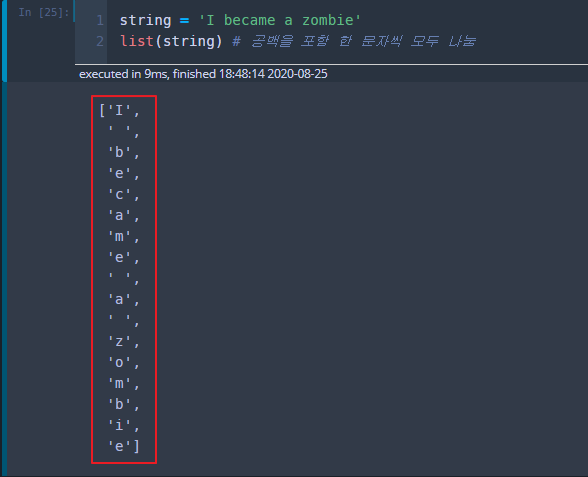
## 1. 문자열을 리스트로 바꾸는 2가지 방법

### 1) list() 함수 이용

|  |  |
| --- | --- |
|  | string = 'I became a zombie' |
|  | list(string) *# 공백을 포함 한 문자씩 모두 나눔* |

list() 함수에 문자열을 넣으면 한 문자씩 다 나누어 리스트를 생성합니다. (공백도 한 문자로 취급)

**\* 출력**

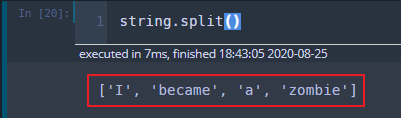
list(string) 출력

### 2) split() 함수 이용

string.split()

문자열.split() 을 사용하면, 문자열에 대해 공백을 구분자로 나누어 리스트를 생성합니다.

**\* 출력**

string.split()

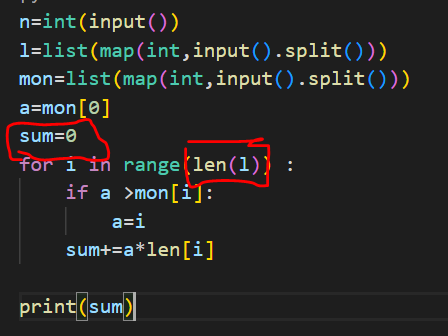
split은 최대 몇 번 쪼갤지, 무엇을 기준으로 슬라이싱할지 등..

13305 주유소

-n개의 도시라고 n! 을 넣어주었다.

위처럼 두 배열의 길이가 서로 다르면 for문을 0부터 넣는 방법보단

sum=a\*l[0]으로 초기값을 넣은 뒤 for문 범위를range(1,n-1)로 하는 것이 낫다. (그럼 range(1,3)이므로 범위는 1, 2까지이고 l[3]은 비교할 필요가 없다. 없기도 하고.



n=int(input())

l=list(map(int,input().split()))

mon=list(map(int,input().split()))

a=mon[0]

sum=a\*l[0]

for i in range(1,n-1) :

    if a >mon[i]:

        a=mon[i]

    sum+=a\*l[i]

print(sum)

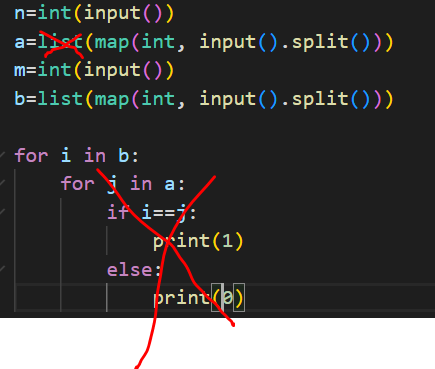
1920 수 찾기 (이분탐색)

**a를 집합(set)로 선언하여 탐색**하면 더 빠르다

if i in a: 이딴 식으로 써도 되는구나..파이썬 댑악이네

#삽질

-저렇게 i,j로 써버리면 결과값도 i\*j만큼 나온다 ㅠ



n=int(input())

a=set(map(int, input().split()))

m=int(input())

b=list(map(int, input().split()))

for i in b:

    if i in a: <<이렇게 써도 되군아…조건 만족시 1 반환

        print(1)

    else:

        print(0)

10816 숫자카드2

해쉬테이블

#삽질

n=int(input())

a=set(map(int, input().split()))

m=int(input())

b=list(map(int, input().split()))

re=[]

for i in b:

    cnt=0

    for j in a:

        if b[i]==a[j]:

            cnt+=1

    re.append(cnt)

print(re)

답: 에러나옴 ㅠ 나중에 고칠래 (띵)

해쉬구조: key, value쌍으로 이루어진 데이터 구조. key를 이용하여 데이터를 찾아 속도가 빨라짐. (ex.딕셔너리 in 파이썬) 공간은 많이 쓰나 시간은 빠르다.

사용: 검색이 많이 필요한 경우, 저장, 삭제, 읽기 많은 경우, 캐쉬 구현시

 **장점**

* **데이터 저장/검색 속도가 빠릅니다.**
* **해쉬는 키에 대한 데이터가 있는지(중복) 확인이 쉽습니다.**

 **단점**

* **일반적으로 저장공간이 좀더 많이 필요합니다.**
* **여러 키에 해당하는 주소가 동일할 경우 충돌을 해결하기 위한 별도 자료구조가 필요합니다. (충돌 해결 알고리즘)**

 **시간 복잡도**

* **일반적인 경우(충돌이 없는 경우): O(1)**
* **최악의 경우(모든 경우에 충돌이 발생하는 경우): O(n)**

출처: <https://davinci-ai.tistory.com/19> [DAVINCI - AI:티스토리]

아래가 정답인데 넘 어려브서 나중에 다시 리뷰해야겟다~~~~~

from sys import stdin

n = stdin.readline().rstrip()

card = list(map(int,stdin.readline().split()))

m = stdin.readline().rstrip()

test = list(map(int,stdin.readline().split()))

hash = {}

for i in card:

    if i in hash:

        hash[i] += 1

    else:

        hash[i] = 1

for i in test:

    if i in hash:

        print(hash[i], end=' ')

    else:

        print(0, end=' ')

함수 선언하기

def 함수() :

불러오는건 함수()하면 됨

def a():

    print("go")

a()

#함수에 디폴트값 넣기

def a(name,age=20,hobby="book"):

    print("{} {} {}".format(name,age,hobby))

a("이혜진") #이혜진 20 book

a("이혜진",25,"run") #이혜진 25 run

#인자를 바꿔도 출력 가능

a(age=25,hobby="run",name="이혜진") #이혜진 25 run

**줄바꿈 없이 출력** 방법: print인자 **끝에 end=” ”를 붙여준다. 얼마든지 바꿀 수 있음**

print()인자 끝에 end=”?”넣어주면, 끝에 ? 출력하고 바로 다음 print가 붙어서 나온다

for i in range(5)

print(i,end=” “)

함수에 가변인자 넣기

#함수에 가변인자(인자 여러개 넣기, \*인자)

def a(name,age, \*hobby):

    print(name,age,end=" ")

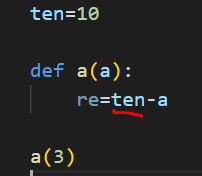
    for i in hobby:

        print(i,end=" ")

a("이혜진",20, "게임") #이혜진 20 게임

a("이혜진",25,"운동","독서","코딩") #이혜진 25 운동 독서 코딩

함수 안에 지역변수를 써야 함.



ten은 전역변수이다. 함수 안에서 호출하면 오류 남.

* 1. a()함수 안에 ten을 정의해준다
  2. global키워드를 쓴다. (global ten)

반환은 return

표중체중 함수:

def w(h,g):

    if g=="여":

        re=h\*h\*21

    if g=="남":

        re=h\*h\*22

    print("키{}m {}의 표준체중: {:.2f}kg".format(h,g,re))

w(170/100,"여")

키1.7m 여의 표준체중: 60.69kg

sys

import sys

print("야","너",file=sys.stdout)#표준 출력

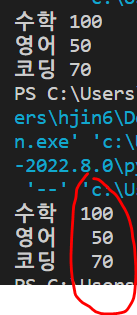
print("야","너",file=sys.stderr)#표준 에러 처리

야 너

야 너

오른쪽 정렬시

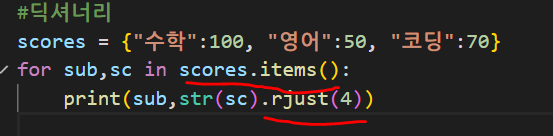
문자.rjust(4) :4칸을 비운다는 뜻



딕셔너리 {아이템, 아이템, 아이템 }

아이템=키,밸류

아이템 값들을 차례대로 출력시



<0으로 채워 출력하는 방법>

print( **“{:03}”.format(정수)** ) : 칸이3개 나머지 0으로 채우기.

print( **문자.zfill(3)** ) : 나머지 0으로 채우기

표준입출력- 사용자 입력시 항상 str형식으로 받음.

#빈 공간, 오른쪽 정렬, 10자리

print("{0: >10}".format(500))

#+-표시

print("{0: >+10}".format(500))

#빈칸\_, 왼쪽정렬

print("{0:\_<10}".format(500))

#3자리마다 ,찍기

print("{0:,}".format(1000000000000000))

#3자리마다 ,찍기 ,+-

print("{0:+,}".format(1000000000000000))

#{0: <+30}

500

+500

500\_\_\_\_\_\_\_

1,000,000,000,000,000

+1,000,000,000,000,000

파일

변수=open(“이름.txt”,”w”,encoding=”utf8”)

encoding=”utf8” 쓰는 이유: 한글 깨지지 말라고

“w” : write

“a”: append(추가)

“이름.txt”는 open()함수 인자때만 등장하지(첫 줄), 나머지는 파일 변수인 me\_f가 함수랑 같이 사용됨.

1)파일 생성

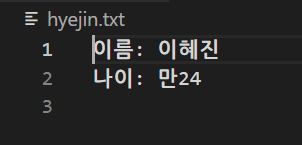
me\_f=open("hyejin.txt","w",encoding="utf8")

print("이름: 이혜진",file=me\_f)

print("나이: 만24",file=me\_f)

me\_f.close()

실행시



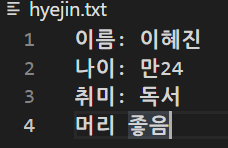
2)내용 추가

me\_f=open("hyejin.txt","a",encoding="utf8")

me\_f.write("취미: 독서")

me\_f.write("\n머리 좋음") <<.write는 줄바꿈 없어서 \n추가

me\_f.close()



3) 파일 읽기

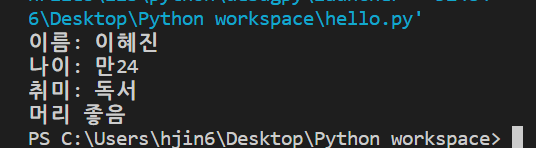
3-1)한 번에 읽기1

me\_f=open("hyejin.txt","r",encoding="utf8")

print(me\_f.read())

me\_f.close()

터미널 창에 나타남



3-1) 한 번에 읽기2 : readline() 반복문

값이 없으면 반복문 탈출

me\_f=open("hyejin.txt","r",encoding="utf8")

while True:

    a=me\_f.readline()

    if not a:

        break

    print(a,end="") <<줄바꿈 없이 위와 동일하게 나옴

me\_f.close()

3-1)한 번에 읽기3: readlines() 리스트 활용

me\_f=open("hyejin.txt","r",encoding="utf8")

a\_list=me\_f.readlines() #readline's', list형태로 저장

for i in a\_list:

    print(i,end=" ")

me\_f.close()

3-2) 한 줄씩 읽기

me\_f=open("hyejin.txt","r",encoding="utf8")

print(me\_f.readline()) #한 줄 읽고 커서는 다음 줄로 이동

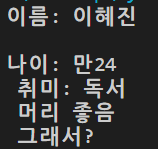
print(me\_f.readline(),end=" ") #줄바꿈 안하고 싶으면 end=" "인자 추가

print(me\_f.readline(),end=" ")

print(me\_f.readline(),end=" ")

print(me\_f.readline(),end=" ")

me\_f.close()



import 모듈

피클

1920백준

n=int(input())

a=map(int,input().split())

m=int(input())

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

for i in b: #b안에 있는 걸 흝으며

    if i in a: #그 b안에 있는 i하나하나가 a에 있으면

        print(1)

    else:

        print(0)

-b가 a안에 있는지 확인, a안에 있으면 1, 없으면 0 출력

10816

1) 삽질 : 왜 틀렷는지 머를..답 오류남(띵)

n=int(input())

a=map(int,input().split())

m=int(input())

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

for i in b: #b안에 있는 걸 흝으며

    cnt=0

    for i in range(n) :

        if i in a: #그 b안에 있는 i하나하나가 a에 있으면

            cnt+=1

    print(cnt,end=" ")

>>이진탐색을 더 배운 다음 풀어보아야겠다

**Lower Bound와 Upper Bound (bisect라이브러리에 있음)**

**from 라이브러리 import 쓸것들 <<이렇게 선언하면 됨**

**이 문제에 적합한 알고리즘은 Lower Bound와 Upper Bound일 것이다. 각각은 경계값을 찾는것. 즉, 찾고자 하는 수가 처음 나온 위치와 마지막으로 나온 위치를 찾아내므로 이 둘의 차이를 계산하면 그것이 카드의 개수가 될 것이다.**

**그렇다면 정렬을 한 뒤에 upper bound-lower bound+1하면 되지 않을까**

**-10 -10 0 2 3 3 3 10 이면 3을 찾는 다 했을 때 6-4+1 =3번!**

from bisect import bisect\_left, bisect\_right

n=int(input())

a=list(map(int,input().split()))

a=sorted(a)

m=int(input())

b=list(map(int,input().split()))

for i in b:

    if i in a: <<<<이거 빼라..

     print(bisect\_right(a,i)-bisect\_left(a,i)+1, end=" ")

    else :

        print(0,end=" ")

이렇게 적었더니 시간초과가 났다.. 개빡쵸

답)

from bisect import bisect\_left, bisect\_right

n=int(input())

a=list(map(int,input().split()))

a=sorted(a)

m=int(input())

b=list(map(int,input().split()))

result=[]

for i in b:

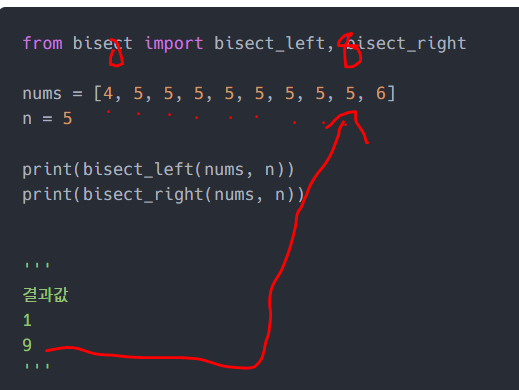
        re=bisect\_right(a,i)-bisect\_left(a,i)

        result.append(re)

for i in result:

    print(i,end=" ")

-bisect\_right(대상, 찾을 것) (이미+1해서 나오기 때문에 식에서 해줄 필요가!!!!!업가다ㅏ!!!!!!!!!!)



pickle 파일 (유용한 라이브러리)

1) pro\_file.pickle 만들기, 쓰기

import pickle

pro\_file = open("profile.pickle","wb")

p = {"이름: 박명수", "나이: 30", "취미: 놀기"}

print(p)

pickle.dump(p,pro\_file) #p 정보를 pro\_file에 저장

pro\_file.close()

2)피클 파일 불러오기

import pickle

pro\_file = open("profile.pickle","rb")

p=pickle.load(pro\_file) #파일에 있는 정보를 p에 불러오기

print(p)

pro\_file.close()

파일을 읽고 쓸 때, with함수를 이용하자.: close()로 매번 닫아주지 않아도 됨,

1)파일 쓰기

**with open(“파일.txt”,”w”,encoding=”utf8”) as 변수:**

변수.write(“쓸 내용~~~~~~~”)

with open("study.txt","w",encoding="utf8") as study\_file:

    study\_file.write("파일내용어쩌구저쩌구")

2)파일 불러오기

**with open(“파일.txt”,”r”,encoding=”utf8”) as 변수:**

변수.read(“쓸 내용~~~~~~~”)

with open("study.txt","r",encoding="utf8") as study\_file:

    print(study\_file.read())

2 추가)pickle파일 with를 통해 불러오기

import pickle

with open("profile.pickle","rb") as pro\_file:

    print(pickle.load(pro\_file))

퀴즈) “-주차 보고서.txt”를 50개 만들기

for i in range(50):

    with open("{}주차.txt".format(i+1),"w",encoding="utf8") as b\_file:

        b\_file.write("-{}주차- \n부서:\n이름:\n업무: ".format(i+1))

클래스 (붕어빵 틀)

struct랑 비슷한듯

사용방법:

**class 이름:**

**def \_\_init\_\_(self, 인자)**

**어쩌고~~~~~~~**

ex)계산기



딜러1 2 3 탱커1 2 3 … 일일히 만들기 힘들어

(전)

#딜러

name="딜러"

hp=30

damage=500

#탱커

name="탱커"

hp=3000

damage=1

(class사용 후)

class Unit:

    def \_\_init\_\_(self,name,hp,damage):

        self.name=name

        self.hp=hp

        self.damage=damage

        print("이름:{}\n체력:{}\n데미지:{}".format(self.name,self.hp,self.damage))

딜1=Unit("딜러1",30,500)

딜2=Unit("딜러2",30,500)

탱1=Unit("탱커1",200,30)

객체(클래스의 인스턴스): 딜1, 딜2, 탱1

self인자를 빼고 넘겨주기.

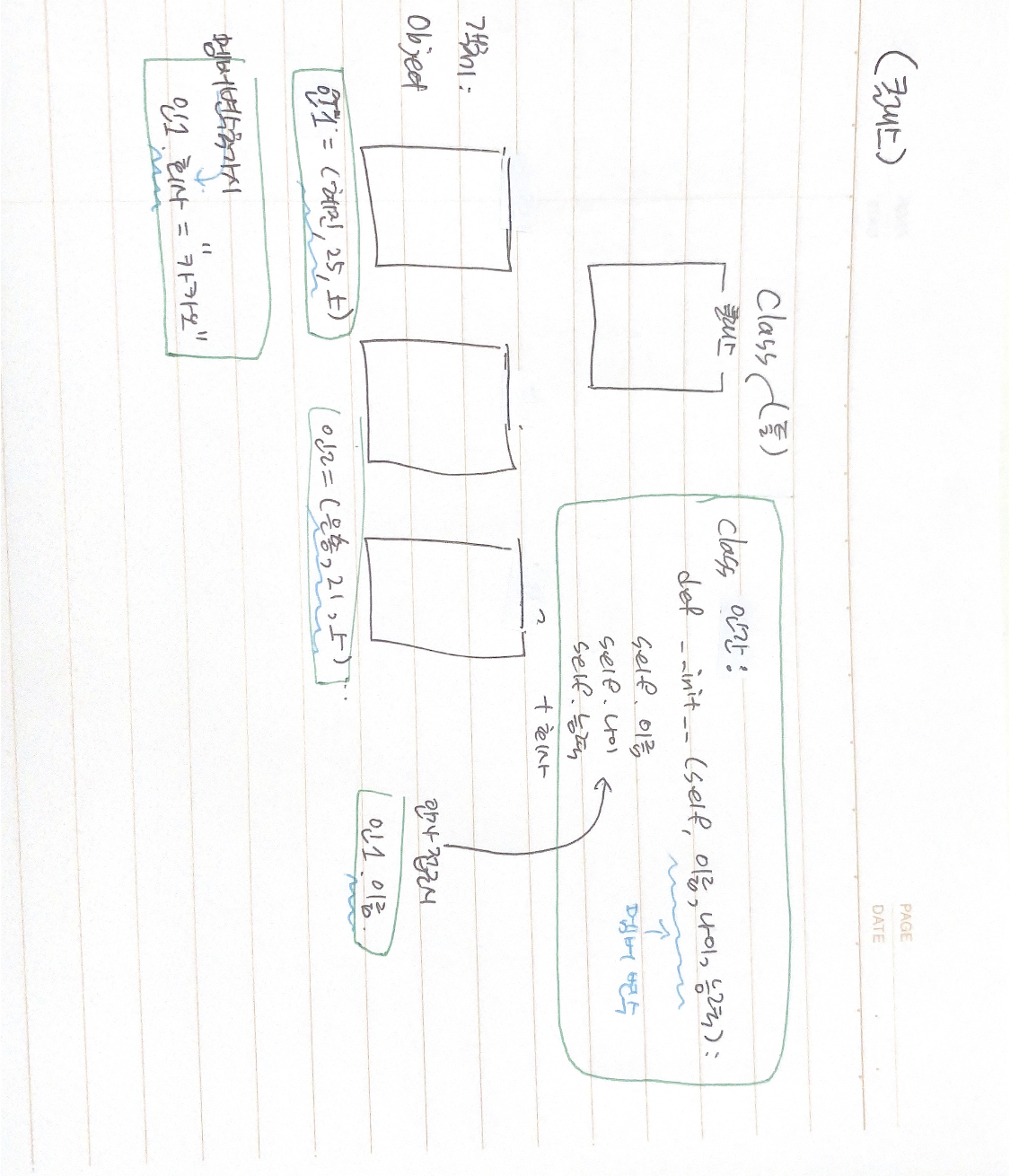
멤버 변수

**멤버변수.인자** 로 접근 가능

print ("딜1의 정보:{} {}".format(딜1.damage,딜1.hp))

새로운 멤버 변수를 선언할 수도 있다

딜1.skill="죽이기"



클래스를 사용하지 않고 함수 작성한다면 보통 다음과 같이 작성한다.

def some\_function(something):

print(something)

그러나 클래스를 이용한다면 기본적으로 다음과 같이 작성하게 된다.

class SomeClass:

def \_\_init\_\_(self,something):

self.something = something

def some\_function(self):

print(self.something)

클래스 구성을 사용하는 메리트는 다음과 같다고 할 수 있다.

* 글로벌 변수를 없애고, 모든 변수를 어떠한 스코프에 소속시킨다.
* 몇 번이고 재사용할 수 있다.
* 코드의 수정을 최소화한다.
* 함수 실행중에, 함수 자신을 다시 호출하는 처리 등이 가능하게 하다

출처: <https://engineer-mole.tistory.com/190> [매일 꾸준히, 더 깊이:티스토리]

## self의 역할

 지금까지 봤던 클래스 안에 있던 self에 대해 찾아보니  "인스턴스 자신", "그 시점의 자신", "메소드의 임의의 인수"등 다양하게 부르고 있었다. 그러나 클래스의 구성을 취득할 때에 정형의 구문으로써 기억해두면 괜찮은 것 같다.

## \_\_init\_\_의 이해

  self와 나란히 클래스 내에 등장하는 \_\_init\_\_에 대해서는 알아보자. \_\_init\_\_에 대해 간단히 정리하자면, 다음과 같다.

* 컨스트럭터라고 불리는 초기화를 위한 함수(메소드)
* 인스턴스화를 실시할 때 반드시 처음에 호출되는 특수한 함수
* 오브젝트 생성(인스턴스를 생성)과 관련하여 데이터의 초기를 실시하는 함수

출처: <https://engineer-mole.tistory.com/190> [매일 꾸준히, 더 깊이:티스토리]

class Person:

    def \_\_init\_\_(self,name,age,height,weight):

        self.name=name

        self.age=age

        self.height=height

        self.weight=weight

        print("이름:{},나이:{}".format(self.name,self.age))

    def f1(self,school): #메소드1

        print("고등학교:{}".format(school))

    def f2(self,Univ): #메소드2

        print("대학교:{}".format(Univ))

#HJ 유닛 만들기

HJ=Person("혜진","22",170,58)

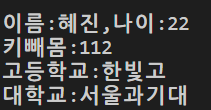
#HJ인자에 접근하기

print("키빼몸:{}".format(HJ.height-HJ.weight))

#메소드 접근

HJ.f1("한빛고")

HJ.f2("서울과기대")



상속(inheritance)

부모클래스(Super Class)의 내용을 자식클래스(Sub Class)가 물려받는 것.

* class 에 속한 함수 (method)는 첫 번째 인자로 self를 입력합니다. (사실 self가 아니어도 상관없지만, 일반적으로 self를 많이 사용합니다.)

### 인스턴스(instance) 생성과 객체(object)

객체는 바로 밑의 예시에서 p1, p2, p3를 일컫습니다.

정의된 클래스(class)로부터 생성된 녀석을 **인스턴스(instance) 혹은 객체(object)**라고 합니다.

하지만, 용어의 온도차(?)는 존재합니다.

p1 = Person()

p2 = Person()

p3 = Person()

p1.introduce()

그렇다면 객체와 인스턴스의 차이는 무엇일까?

이렇게 생각해 보자. p1 = Person() 이렇게 만든 p1는 **객체**이다.

그리고 p1 객체는 Person의 **인스턴스**이다.

즉 **인스턴스**라는 말은 특정 객체(p1)가 어떤 클래스(Person)의 객체인지를 관계 위주로 설명할 때 사용한다.

**"p1는 인스턴스"**보다는 **"p1은 객체"**라는 표현이 어울리며 **"p1는 Person의 객체"**보다는 **"p1은 Person의 인스턴스"**라는 표현이 훨씬 잘 어울린다.

상속https://jacoblee.io/class-inheritance/

class Dog:

    species = "Canis familiaris"

    def \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

    def \_\_str\_\_(self):

        return f"{self.name} is {self.age} years old"

class Bulldog(Dog):

    pass

class Pome(Dog):

    pass

class Retriever(Dog):

    pass

구찌=Bulldog("구찌",3)

당당이=Bulldog("당당이",5)

뽀삐=Pome("뽀삐",2)

소녀=Retriever("소녀",4)

print(소녀)

#객체(소녀)가 속한 클래스 확인하기

print(type(소녀)) #<class '\_\_main\_\_.Retriever'>

print(isinstance(소녀,Dog)) #부모클래스에 속해있는지 확인 True

오버라이딩

퀴즈 8

class House:

    def \_\_init\_\_(self,location,h\_type,d\_type,price,year):

        self.location =location

        self.h\_type =h\_type

        self.d\_type =d\_type

        self.price =price

        self.year =year

    def show(self):

        print(self.location, self.h\_type,self.d\_type,\

            self.price,self.year)

house=[]

house1=House("강남","아파트","매매","10억","2010년")

house2=House("마포","오피스텔","전세","5억","2007년")

house3=House("송파","빌라","월세","500/50","2000년")

house.append(house1)

house.append(house2)

house.append(house3)

print("총 매물: {}".format(len(house)))

for h in house:

    h.show()

--

예외처리

:에러 발생시 처리해주는 것

숫자 입력 받아야하는데 문자 받는 경우, 서버 접속 잘 안될 때..

### try, except문

다음은 오류 처리를 위한 try, except문의 기본 구조이다.

**try**:

...

**except** [발생 오류[**as** 오류 메시지 변수]]:

...

try 블록 수행 중 오류가 발생하면 except 블록이 수행된다. 하지만 try 블록에서 오류가 발생하지 않는다면 except 블록은 수행되지 않는다.

except 구문은 다음 3가지 방법으로 사용할 수 있다.

**1. try, except만 쓰는 방법**

**try**:

...

**except**:

...

이 경우는 오류 종류에 상관없이 오류가 발생하면 except 블록을 수행한다.

**2. 발생 오류만 포함한 except문**

**try**:

...

**except** 발생 오류:

...

이 경우는 오류가 발생했을 때 except문에 미리 정해 놓은 오류 이름과 일치할 때만 except 블록을 수행한다는 뜻이다.

**3. 발생 오류와 오류 메시지 변수까지 포함한 except문**

**try**:

...

**except** 발생 오류 **as** 오류 메시지 변수:

...

나눗셈 계산기

def f(num1, num2):

    try:

        re=num1/num2

        print(re)

    except ZeroDivisionError as e:

        print(e)

f(4,0)

결과

그럼 3의 배수를 입력받은 뒤 숫자가 3의 배수가 아니면 예외를 발생시켜보겠습니다.

**try\_except\_raise.py**

try:

    x = int(input('3의 배수를 입력하세요: '))

    if x % 3 != 0:                                 # x가 3의 배수가 아니면

        raise Exception('3의 배수가 아닙니다.')    # 예외를 발생시킴

    print(x)

except Exception as e:                             # 예외가 발생했을 때 실행됨

    print('예외가 발생했습니다.', e)

**실행 결과**

3의 배수를 입력하세요: 5 (입력)

예외가 발생했습니다. 3의 배수가 아닙니다.

5는 3의 배수가 아니므로 raise Exception('3의 배수가 아닙니다.')로 예외를 발생시켰습니다. 이때 Exception에 넣은 에러 메시지는 except Exception as e:의 e에 들어갑니다.

그리고 raise로 예외를 발생시키면 raise 아래에 있는 코드는 실행되지 않고 바로 except로 넘어갑니다. 따라서 try의 print(x)는 실행되지 않습니다.

class BigNumError(Exception):

    def \_\_init\_\_(self, msg):

        self.msg=msg

    def \_\_str\_\_(self):

        return self.msg

try:

    print("한 자리 숫자 나누기 계산기")

    num1,num2=map(int, input().split())

    if num1 >=10 or num2 >=10:

        raise BigNumError("입력값: {} {}".format(num1,num2))

    print("{}/{}={}".format(num1,num2,int(num1/num2)))

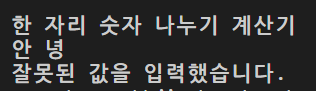
except ValueError:

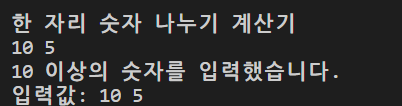
    print("잘못된 값을 입력했습니다.")

except BigNumError as err:

    print("10 이상의 숫자를 입력했습니다.")

    print(err)





finally 구문

finally:

    print("계산기를 이용해 주셔서 감사합니다.")

를 뒤에 추가하면 에러가 발생하던 안하던 항상 무조건!!finally 안 구문은 처리됨.

에러가 있을 때 중지하지 않고 계속 하도록 해 프로그래밍 완성도를 높임.

치킨

class SoldOutError(Exception):

    pass

chicken =10

wating =1

while(True):

    try:

        print("[남은 치킨: {}]".format(chicken))

        order = int(input("치킨 몇 마리 주문?"))

        if order>chicken:

            print("재료가 부족합니다")

        elif order <=0:

            raise ValueError

        else:

            print("[대기번호 {}] {}마리 주문이 완료되었습니다.".format(wating,order))

            wating+=1

            chicken-=order

        if chicken==0:

            raise SoldOutError

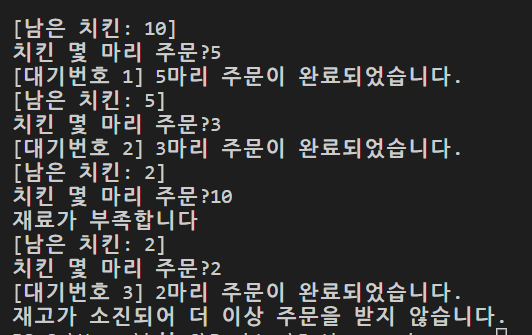
    except ValueError:

        print("잘못된 값을 입력하였습니다.")

    except SoldOutError:

        print("재고가 소진되어 더 이상 주문을 받지 않습니다.")

        break



모듈: 필요한 부품끼리 만들어진 파일.

엔진에도 수리작업을 해야 할 경우 비용도 많이 드는데

자동차 타이어가 고장나면 타이어만 고치는 것처럼..

확장자는 py

theater\_module.py

#일반 가격

def price(people):

    print("{}명 가격은 {}원 입니다.".format(people,people\*10000))

#조조할인 가격

def price\_morning(people):

    print("{}명 가격은 {}원 입니다.".format(people,people\*10000\*0.6))

#군인 할인

def price\_sold(people):

    print("{}명 가격은 {}원 입니다.".format(people,people\*10000\*0.4))

함수호출방법

1)

import theater\_module

theater\_module.price(3) #일반 3명 가격

theater\_module.price\_morning(3) #조조 3명 가격

theater\_module.price\_sold(3) #군인 3명 가격

2)

import theater\_module as mv #모듈을 mv로 호출

mv.price(3)

mv.price\_morning(3)

mv.price\_sold(3)

3)젤 간단

from theater\_module import \*

price(3)

price\_morning(3)

price\_sold(3)

3-1) 거기 안에 있는 필요한 함수만 호출 가능

from theater\_module import price,price\_morning

price(3)

price\_morning(3)

함수 price에 별명을 붙일 수도 있음

from theater\_module import price as f1

f1(3)

모듈

.py로 만든 파이썬 파일은 모두 모듈이다.

#mod1.py

def add(a,b):

    return a+b

def sub(a,b):

    return a-b

모듈(mod1.py)를 만든 모습

같은 디렉터리 안에 다른 .py에서 위 모듈을 불러와 실행한 모습

1) mod1.add() 쓰는 경우

import mod1

print (mod1.add(3,4))

print(mod1.sub(4,2))

 실수로 import mod1.py로 입력하지 않도록 주의하자.

모듈 안에 있는 add함수를 사용하고 싶으면 mod1.add()를 쓰면 된다.

2) add()만 쓰고 싶은 경우(좀 더 짧게)

2-1)

from mod1 import add,sub

print(add(3,4))

print(sub(4,2))

2-2)

from mod1 import \* #'\*'는 '모든 것'이라는 뜻

print(add(3,4))

print(sub(4,2))

패키지는 도트(.)를 사용하여 파이썬 모듈을 계층적으로 관리할 수 있게 해준다. 예를 들어 모듈 이름이 A,B인 경우 A는 패키지 이름, B는 A패키지의 B모듈이 된다.

장점: 복잡한 파이썬 프로그램일 경우 공동작업, 유지 보수 등에 유리함.

다른 모듈과 이름이 겹치더라도 더 안전.

*가상의 game 패키지 예*

game/

\_\_init\_\_.py

sound/

\_\_init\_\_.py

echo.py

wav.py

graphic/

\_\_init\_\_.py

screen.py

render.py

play/

\_\_init\_\_.py

run.py

test.py

\_\_all\_\_: https://nadocoding.tistory.com/79

<if \_\_name\_\_==”\_\_main\_\_”의 의미>

C언어, 자바는 main()함수로 프로그램을 실행시킨다.

그러나 파이썬은 main함수가 존재하지 않는다.

파이썬의 특징

1. 들여쓰기를 통해 코드 실행의 레벨을 결정한다

2. main이 존재하지 않는다.

**def** **sum**(num1, num2)

**return** **num1** + **num2**

**print**(sum(1, 1))

파이썬 코드에서 들여쓰기를 한 return문은 들은 sum 이라는 함수에 속해있는 코드로 볼 수 있고, print함수는 sum함수 밖에 존재하는 것이다.

결국 파이썬은 main문이 없는 대신에, 들여쓰기가 되지 않은 Level0의 코드를 가장 먼저 실행시킨다.

파이썬은 다양한 정보를 담고 있는 내장변수가 존재한다.

\_\_name\_\_은 현재 모듈의 이름을 담고 있는 내장 변수이다.

이 변수는

직접 실행된 모듈의 경우 : \_\_main\_\_

직접 실행되지 않은 import된 모듈의 경우: 모듈의 이름(파일명)

을 갖는다.

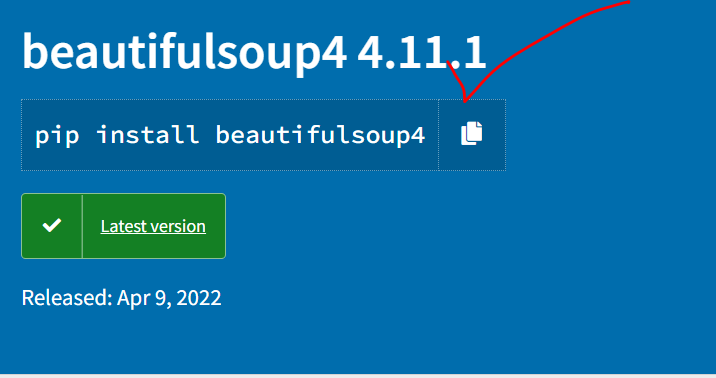
모듈에 if \_\_name\_\_=="\_\_main\_\_"이라는 조건문을 넣어주고 그 아래는 직접 실행시켰을 때만 실행되길 원하는 코드들을 넣어주는 것

예를 들어 모듈내에서만 사용하는 테스트나 로그같은 출력부분들을 넣어주게 되면 실제 import될 때는 출력되지 않으면서 모듈에 포함된 기능만을 포함시킬 수 있기 때문에 상당히 효율적인 프로그래밍이 가능하게 된다.

다른사람이 만든 패키지

<https://pypi.org/>

<https://pypi.org/project/beautifulsoup4/>

여기서복사, 터미널 창에 명령

정보 보기 pip show beautifulsoup4

삭제시 pip uninstall beautifulsoup4

dir(객체)

그 객체가 어떤 변수, 함수를 가지고있는지 표시

ex) dir(random)

cls터미널창 모두 지우기

내장 함수 (디폴트)

<https://docs.python.org/3/library/functions.html>

외장 함수(import사용필수)

<https://docs.python.org/3/py-modindex.html>

ex) #glob: 경로 내 폴더/파일 목록 조회 (윈도우에서 dir명령과 같음)

import glob

print(glob.glob("\*.py")) #확장자가 py인 모든 파일

#OS:운영체제에서 제공하는 기본 기능

import os

print(os.getcwd()) #현재 디렉토리

folder = "sample\_dir"

if os.path.exists(folder):

    print("이미 존재하는 폴더")

else:

    os.makedirs(folder)

    print(folder,"폴더를 생성하였습니다")

그 외 폴더 삭제 등등 가능

import time

print(time.localtime())

print(time.strftime("%Y %m %d %H:%M:%S"))

time.struct\_time(tm\_year=2022, tm\_mon=7, tm\_mday=3, tm\_hour=10, tm\_min=35, tm\_sec=30, tm\_wday=6, tm\_yday=184, tm\_isdst=0)

2022 07 03 10:35:07

import datetime

print("오늘 날짜는",datetime.date.today())

오늘 날짜는 2022-07-03

import datetime

today=datetime.date.today()

td=datetime.timedelta(days=100) #100일 저장

print("오늘로부터 100일 후: ",today+td)

오늘로부터 100일 후: 2022-10-11

퀴즈10

#byme.py

def sign():

    print("이 프로그램은 나도코딩에 의해 만들어짐")

#main.py

import byme

byme.sign()

이 프로그램은 나도코딩에 의해 만들어짐