2.variables

() Parentheses
[] Brackets
{} Braces
shell=console=실행창

type() 타입 알려주는거
int 정수형
float 소수형
소수 j (허수 i를 의미)
complex 복소수형

0o 8진수
0x 16진수

// 몫
% 나머지
** 승

A=B : B를 A에 지정한다
26.0만 있는 건 object 아니고 value임
degrees_celcius/id1->26.0
-Value 26.0 has the memory address id1.
-The object at the memory address id1 has type float and the value 26.0
-Variable degree_celcius contains the memory address id1.

3.Function(1)

abs 절댓값
pow(x,y) 승 x**y
pow(x,y,z)는 x**(y%z)
※ pow(2,4,3)은 2**(4%3)=1
round 반올림
※ round(4.3)은 4
※ round(3.1415,2)는 3.14

typecast 형변환
int 반(버림)
int(34.6)은 34
float(21)은 21.0

argument 입력값
return 반환한다. 출력한다
display, print 화면 출력
output 함수 출력
return value 함수값

id: 컴퓨터 저장된 메모리 주소값
id(-9)하면 어쩌고저쩌고
값 같은 (평범한) a,b 있으면 주소값 같음. ; 재사용
값 같은 (안평범) a,b 있으면 주소값 다름.

```
>>> def <<함수이름>> :
        return <<~~>>
```

def 함수 정의 시작 -> : -> 엔터 ->
(들여쓰기;네칸 space bar) -> return -> (엔터 두번)

local variables 지역 변수
function header / function name / (function parameters (=arguments) 입력변수)
function body
function call
지역변수: a,b,c,x,first,second,third...

```
>>>def say():
        print('Hello')
>>>word=say()
Hello
>>>print(word)
None
```

4.Functions(2)

docstring=documentation string 메뉴얼
""" (start) ~ """ (end) 메모장 같은 셈

>>> <<function header>>
"""
<<Types of (Parameters) -> Return value>>
<<Fuction description>>
<<Example calls>>
"""
<<function body >>

5.String

string 문자열 타입 str() '~'(quotes)(quotation mark) or "~"
character 문자(알파벳, 한글, 음악기호, 숫자, 심볼) 한 글자
''empty string도 가능 (길이 0)

operations(계산) on strings
▷ len() 길이(띄어쓰기도 하나로 침)
▷ +/* 9=(string랑 숫자 못더해!)(*는 이어붙이기 횟수)
▷ int('0') 하면 0
▷ float('-324') 하면 -324.0

escape sequence는 캐릭터 하나로 봄.
▷ \' \"
▷ \\ backslash
▷ \t tab
▷ \n newline
▷ \r carriage return

(''' ''' 할 때 엔터 같은거 가능)
print : 사람이 보는 것처럼 나타내라.
tab은 알아서 잘 정렬해줌.
print는 , 이나 ' 빼고 보여줌
print는 기존 sep=' ', end='\n'임

int랑 float 합쳐서 number

>>>s=input()
d
>>>s
'd'

input은 무조건 string으로 받음
int(input())이렇게 코드 짜면 됨
s=input("질문어쩌고") 엔터치면, 질문어쩌고, 쓸말,,,
print(s) 하면 쓸말

slicing string: 0~ 그리고 -1~ 지정됨. a[2] a[-1] 캐릭터 하나 볼 수 있음.
c=a[9:12]는 9이상 12미만. 정방향.
거꾸로 가려면 a[7:1:-1]
a[2:5:2] 두칸째씩 나타내라

6.Boolean

bool : Boolean / True or False 두 가지 값밖에 없음
Boolean operators:
▷and, or, not
not: unary operator 한 개에 쓰임.
and, or: binary operators 두 개

and 는 False 하나만 있어도 False
or 는 True 하나만 있어도 True

(XOR exclusive or 둘 중 하나만 true여야 true.)

Relational operators
▷ >,<.>=,<=,==,!=(not equal to)

(iff : if and only if 역도 참)

0,None만 False 나머지는 모두 True
True or 혹은 False and 나오면 뒤에 계산도 안하고 그냥 답 도출.

ASCII code 모든 게 다 지정되어 있음
'대문자'<'소문자'
'A'<'Z'<'a'<'z'

in 앞의 문자열이 뒤 문자열에 있는가 (boolean. true 아니면 false 나옴). case sensitive
"" in '???' 항상 true

return print(blabla) 말이 안됨!!!

7.Conditional Statement

if statement. 조건문.
>>> if <<condition>:
        <<block>>
condition : 보통 boolean expression (true 혹은 false)이다.

>>>ph=float(input('Enter the pH level: '))
지정하고,
>>>if ph < 7.0:
        print(ph, "is acidic.")
        print("Becareful with that!")

elif : if에 종속되어 있음. if에서 False 나올 때만. 한 번 true로 판정되면 elif는 절대 안봄.
elif 여러 개 쓸 수 있음.

else: 는 맨뒤, 하나밖에 못씀. if에 종속되어 있음.
else: -> 엔터탭.

if ((elif elif elif elif) (else))

nested if statements : 조건문들끼리 종속되어 있을 수 있음.

조건문에 relational operations 종종 들어감.
결과값이 boolean이어야 함.

boolean variable을 지정할 수 있음.

8.Module

(from scratch 처음부터)

Module. 일반화된 흔한 코드들. object. functions, variables,,,의 집합.
종류: math, random , ...

>>>import math 불러들이는 거!
그러면 type, help, ... 가능해짐

<<module이름>>.<<함수or변수>>
>>>math.sqrt(9) / 3.0
>>>math.pi / 3.141592....

module에 정의되어 있는 함수나 변수 덮어쓰기 가능하긴 함!

>>>from math import pi, sqrt
-> math. 을 하지 않아도 pi나 sqrt만 써도 됨
>>>from math import *
-> math module의 모든 함수, 변수 그냥 됨 (서로 겹치는 거 주의)

shell창에서 새로 만들고 저장(모듈이름.py)하면 module형태 사용 가능 (import 'module이름' 해줘야 사용 가능)

>>>math.sin(math.radians(30))
약 0.5
>>>math.degrees(math.asin(0.5))
약 30

★math module
▷inf 무한대 (data이다)
▷exp(5) 는 e^5 (function이다)
▷log(4) 는 log e 4 (function이다)
▷log10(3) 는 log 10 3 (function이다)

★imp module
▷reload(<<모듈네임>>) 은 모듈네임이라는 모듈을 reload 해주는 함수이다; 중복실행하는 유일방법!(experiment.py를 만들고 shell창에서 import experiment해줘도 되고 import imp 후 imp.reload(experiment)해도 됨)

★random module (변수에 지정해줌!)
(import random 쓰고 시작.)
▷random() ;소수 생성 ;[0,1) 중에
▷uniform(a,b) ;소수 생성 ;[a,b) 중에
▷randrange(stop) ;정수 생성 ;[0,stop) 중
▷randrange(start,stop) ;정수 생성 ;[start,stop) 중
▷randrange(start,stop,step) ;정수 생성 ;[start,start+step,start+step*2,...,stop)중
▷randint(start,stop) ;정수 생성 ;[start,stop]중
※>>>x1=random.random()
※>>>x2=random.uniform(3,6)
※>>>x3=random.randrange(5)
※>>>x4=random.radient(1,9)

★turtle module
▷Pen(forward(4))
▷Pen(backward(5))
▷Pen(up())
▷Pen(up(3))
▷Pen(down())
▷Pen(down(7))
▷Pen(left(8))
▷Pen(right(2))
▷Pen(rest())
▷Pen(clear())
※>>>turtle.Pen(forward(4))
※>>>turtle.Pen(down())

★__name__
>>>__name__ /'__현재 shell의 상태__' 알려줌

>>><<모듈네임>>.__name__ /imported된 모듈에 대해서만 '<<모듈네임>>'반환.

※>>>__name__
'__main__'
※>>>experiment.__name__
'experiment'

8.Class,Method

class :another kind of object :how Python represents a type :has methods

※>>>type(17)
<class 'int'>
※>>>type(17.0)
<class 'float'>
※>>>type('hello')
<class 'str'>

Method 일종의 함수. (class 안에 정의된, 있는 function)
str methods / int methods / bool methods / ... 모든 type은 고유 set of methods 가짐. 모든 type들이 class로 분류됨.

class 를 먼저 만들어 놓고, 함수나 변수 채워놓음. / class: 붕어빵 틀 / 객체: 붕어빵

method : 첫번째 argument(입력 변수)가 스트링이어야 함!
str method 가 일반적!

Functions in Module
<->
Methods in Class

>>> <<module이름>>.<<function or variable 이름>>
<->
>>>
<<class이름>>.<<method이름('argument값',a)>>
>>>
<<'argument값';'expression'>>.<<method이름(a)>>
(expression.method_name(arguments) 순!)

※>>>import math
※>>>math.sqrt(9.0)
3.0
※>>>str.upper('trump')
'TRUMP'
※>>>'trump'.upper()
'TRUMP'

★ str이라는 class의 methods
▷capitalize('~') 첫 글자를 대문자로
▷upper('~') 모두 대문자로
▷center('string',26) 길이 맞게 중앙정렬
▷count('string','s') 해당 글자가 몇 개 있는지
▷startswith(str,beginning) 해당 str이
▷beginning으로 시작되는지 판단 (Boolean임)
▷endswith(str,end) 해당 str이 end로 끝나는지 판단 (Boolean임)
▷find(str,s) 해당 str에 s가 어느 위치에? 없으면 -1 반환.
▷find(str,s,beg) 해당 str에 s가 beg포함 뒤 중 어느 위치에?
▷find(str,s,beg,end) 해당 str에 s가 [beg,end] 중 어느 위치에?
▷format({}포함하는str,<<expression>>) 해당 str의 {}에다가 expression을 넣어줌.
▷islower(str) 해당 str가 전부 소문자일 때만 True 반환
▷isupper(str) 해당 str가 전부 전부 대문자일 때만 True 반환
▷lower(str) 해당 str를 전부 소문자로 낮춰 반환
▷upper(str) 해당 str를 전부 대문자로 높여 반환
▷lstrip(str) 해당 str의 왼쪽 공백 모두 없애 반환
▷lstrip(str,s) 해당 str의 맨 왼쪽에 s 있을 경우 s 없애 반환
▷rstrip(str) 해당 str의 오른쪽 공백 모두 없애 반환
▷rstrip(str,s) 해당 str의 맨 오른쪽에 s 있을 경우 s 없애 반환
▷strip(str) 해당 str의 양옆 공백 없애 반환
▷strip(str,s) 해당 str에서 양옆 끝에 s 있을 경우 s 없애 반환
▷replace(str,old,new) 해당 str에 있던 old를 new로 바꿔 반환
▷split(str) 해당 str에 있던 공백 아닌 것들 같이 있는 것들끼리 str으로 따로 묶어 반환
▷swapcase(str) 해당 str의 대문자는 소문자로, 소문자는 대문자로 바꿔 반환

※>>>str.capitalize('trump')
※>>>str.upper('trump')
※>>>str.center('trump',30)
(class이름이 앞에 있다!)

-객체지향적인 Class
※>>>'trump'.capitalize()
'Trump'
class는 객체 앞으로 빼서 method 사용 가능
빈괄호라도 꼭 있어야 함
※>>>'Center'.center(26)
' Center '
※>>>str.upper('leegyurin')
'LEEGYURIN'

▷str.format()
{0} {1} {2},,,로 argument(입력변수)지정.
※>>>'{0} ate {1}'.format('I','apple')
'I ate apple'
※>>>'{} ate {}'.format('I','apple')
'I ate apple'
※>>>'{1} ate {0}'.format('apple','I')
'I ate apple'
>>>'{0:.2f}'.format(~~) : 0번 스트링에서 소수점 아래 셋째 자리에서 반올림
변수로 해도 된다
※>>>'Pi is {:2f}'.format(3.141592)
'Pi is 3.14'

nesting : 여러개 해도 된다
※>>>'Computer Science'. swapcase(). endswith ('ENCE')
True

Imperative programming

9 List

List는 type이다. 그리고 collection of data다.
List [, ,] 대괄호와 ,로 구별
List는 하나의 object로 취급

List 안의 object 혹은 item 혹은 element 는 variable인 셈
item(element)들이 index 0부터 차례로 저장됨
List가 length가 14라면 indicies는 0부터 13까지 범위된다.
indexname[0]~ 혹은 indexname[-1]~ 불러오기 가능
list 안에 item들 type 섞어서 쓸 수는 있다. ; integers, strings, float, Boolean, other List...

덮어쓰기, modifying(특정 item만) 가능

mutable하다.
>>>x=L[i] : x를 새롭게 지정하는 거
>>>L[i]=x : modifying (string(immutable_은 이게 불가). L[i]를 새롭게 지정하는 것. -> mutable하다. L[i]는 variable같다.
str은 수정하는 게 아니다..(str은 immutable하다)

opterations of lists
▷len(L), max(L), min(L), sum(L), sorted(L)(정렬된)
▷sorted(L) -> 작은 순으로 정렬
▷sorted(L,reverse=True) -> 역순
(sorted(L) 한다고 해서 L의 순서 바뀌는 게 아니다!!!)
▷+, *, del
▷+ 는 list+list만 concatenate연결 가능
▷*는 list*3
▷del (mutable하다) 은 실제로 list의 element를 지운다!!! 조심해서 쓰기

※>>>nobles=['h','n','a']
※>>>gas=input('가스입력하세요:')
※>>>if gas in nobles:
        print('{} is nobel.'.format(gas))
    else:
        print('{} is not nobel.'.format(gas))

※>>>1 in [0,1,2,3]
True
※>>>[1] in [0,1,2,3]
False
※>>>'ar' in 'Bargain'
True
※>>>'ar' in ['Bar','Bargain']
False

slicing 가능
A=list_name[0:4]

Copying할 때에는 list_copying=list_name[:]
list_name이 바뀐다고 해서 list_copying이 바뀌진 않음. 주소값이 새로 지정되는 셈.

Aliasing할 때에는 list_alias=list_name
list_name이 바뀌면 list_alias도 같이 바뀜. 주소값 자체가 복사됨.

```
※>>>def remove_last(L):
        del L[-1]
        return L
※>>>a=[1,2,3,4,5]
※>>>b=remove_last(a)
```
하면 원래 list인 a도 변형이 온다.

```
※>>>def remove_last(L):
        L2=L[:]
        del L2[-1]
        return L2
※>>>a=[1,2,3,4,5]
※>>>b=remove_last(a)
```
하면 원래 list인 a는 안바뀜.

list of list : list 안에 item을 list로 할 수 있다.
=>list_name[0] 혹은 list_name[0][0] 이런 식으로 보기 가능

리스트의 주요 Methods (class는 list다)
▷append(list,v) 해당 list에 v를 마지막에 추가함
※>>>list.append(L,'ooo')
※>>>L.append('oo')
※>>>L.append('oo',center(5))
▷extend(L,V) 해당 리스트 L에 주어진 리스트V의 아이템들을 마지막에 추가함.
▷insert(L,i,x) 해당 L의 주어진 위치 i에 주어진 아이템 v를 삽입하고, 삽입된 위치의 기존 아이템은 뒤에 위치하도록 함
▷remove(L,x) 해당 L 내에서 주어진 아이템 v와 동일한 아이템 중 맨 첫번재꺼 제거함
▷count(L,x) 해당 L에서 주어진 아이템 x와 동일한 아이템의 갯수를 카운트함
▷sort(L) 리스트 내의 아이템을 순서대로 정렬함. 아예.
▷sort(L,reverse=True) 리스트 내의 아이템들을 순서대로 정렬 후 거꾸로.
▷reverse(L) 리스트 내의 아이템을 그냥 거꾸러만 정렬함.
▷pop(L) 해당 리스트 L의 마지막 아이템을 돌려주면서, 그 마지막 아이를 리스트에서 제거함!
※>>>a1=L.pop()
▷index(L,x) 해당 리스트 L에서 x가 몇 번째에 있는가 알려줌!
▷index(L,x,beg) 해당 리스트 L에서 x가 [beg,) 중 몇 번째에 있는가 알려줌
▷index(L,x,beg,end) 해당 리스트 L에서 x가 [beg,end] 중 몇 번째에 있는가 알려줌
▷clear(L) 해당 리스트의 아이템 전부 지움

10 Loop(1)

Loop 반복문
for-loop : list 안에 있는 element 모두 반복해야 끝남.
while-loop : 어떤 condition이 만족할 때만 돎.

>>> for <<variable>> in <<str>>: 혹은
>>> for <<variable>> in <<list>>: 혹은
>>> for <<variable>> in <<range>>:
<<block>> (:must be indented)
variable: loop variable ;덮여씌워짐

※>>>for a in 'str1':
    print(a*3)
sss / ttt / rrr / 111
a: 'str1' 안에 있는 글자 하나하나

※>>>for b in 'str1','str2':
    print(b*3)
str1str1str1 / str2str2str2

※>>>for c in 'str1'+'str2':
    print(c*3)
sss/ttt/rrr/111/sss/ttt/rrr/222

※>>>velocities=[0,9,19,29]
※>>>for v in velocities:
    print('~',v,'~')
v: velocities라는 list 안에 있는 값들

interation: 한 번 실행하는 거
loop이 끝나면 variable은 마지막 값임.

for loop 안에 if 조건절 가능.

range(n) : 하나의 type. 숫자의 collection. (숫자 list인 셈). 0부터 n 전까지의 정수들.
range(n,m) : n부터 m 전까지의 정수들.
range(n,m,p) : n부터 m 전까지의 p번째들 정수들.

※>>>a=range(10)
※>>>a
range(0,10)
※>>>list(a)
[0,1,2,3,4,5,6,7,8,9]

※>>>for num in range(10):
    print(num)
0/1/2/3/4/5/6/7/8/9

※>>>list(range(0,10,2))
[0,2,4,6,8]

※>>>list(range(0,10,-1))
[]

※>>>list(range(10,0,-1))
[10,9,8,7,6,5,4,3,2,1]

※>>>total=0
※>>>for i in range(1,11):
    total=total+i
※>>>total
55

※>>>values=[4,10,3,8,-6]
※>>>for i in range(len(values)):
    print(i,values[i])
0 4 / 1 10 / 2 3 / 3 8 / 4 -6

※>>>values=[4,10,3,8,-6]
※>>>for i in range(len(values)):
    values[i]=values[i]*2
※>>>print(values)
[8,20,6,16,-12]

※>>>metals=['Li','Na','K']
※>>>weights=[6.9, 22.9, 39.0]
※>>>for i in range(len(metals)):
    print(metals[i],weights[i])
Li 6.9 / Na 22.9 / K 39.0

※>>>positive=['Li','Na','K']
※>>>negative=['F','Cl','Br']
※>>>for metal in positive: outerloop임
        for halogen in negative: innerloop임
            print(metal+halogen)
LiF / LiCl / LiBr / NaF / ...

loop 중첩 많이 하지 않는 걸 권장.

구구단테이블만들어보기
※>>>def print_table(n):
#numbers 지정
numbers=list(range(1,n+1))

#첫쨋줄 작성
for i in numbers:
print('\t'+str(i),end='')

#줄바꿈용
print()

#구구단 표
for i in numbers:
    print(i,end='')
    for j in numbers:
        print('\t'+str(i*j),end='')

#outloop 끝날 때마다 줄바꿈
    print()

※>>>elements=[['Li','Na','K'],['F','Cl','Br']]
※>>>for inner_list in elements:
    print(inner_list)
['Li','Na','K'] / ,['F','Cl','Br']

※>>>elements=[['Li','Na','K'],['F','Cl','Br']]
※>>>for inner_list in elements:
    for item in inner_list:
        print(item)
Li / Na / K / F / Cl / Br

ragged list: list 안에 있는 list들의 길이가 제각각

---
while loop : 어떤 condition이 만족할 때만 돎.
>>> while <<expression>>:
<<block>>
expression: boolean(true or false)으로 되는 문장. (if <<conditionn>> : <<block>>와 형식 유사)

※>>>rabbits=3
※>>>while rabbits>0:
    print(rabbits)
    rabbits=rabbits-1
3 / 2 / 1

※>>>time=0
※>>>population=1000
※>>>growth_rate=0.21
※>>>while population<2000:
    population =population +growth_rate *population
    print(round(population))
    time=time+1
※>>>print("It took",time,"minutes for the bacteria to double.")
※>>>print("The final population was",round(population),"bacteria.")
1210 / 1464 / 1772 / 2144 /
It took 4 minutes for the bacteria to double.
The final population was 2144 bacteria.

Ctrl+C 누르면 중단 가능

11 Loop(2)

user input
input 함수는 한 번 입력 받고 그냥 끝남.
반복문 사용하면 interactive 한 프로그램 가능
같은 질문을 반복적으로 물어볼 수 있다.

```
※>>>while text != "quit":
        text=input(" 화학물 입력하세요 (or 'quit' to exit)")
        if text == "quit":
                print("...exiting program")
        elif text == "H20":
                print("Water")
        elif text == "NH3":
                print("Ammonia")
        else:
                print("Unknown compound")
```

while문 안에 if문 있고 그 안에 break
break: break가 포함된 loop에서 즉시 탈출

```
>>>while(for) <<expression>>
        ...
        if <<condition>>:
        break
        ...
```

```
※>>>text=""
※>>>while True:
        text=input("입력하세요(or 'quit' to exit):")
        if text == "quit":
                 print("~~~"
                 break
```

숫자가 몇 번째 index인지 찾고 싶다,

```
※>>>s='C3H7'
※>>>digit_index = -1
※>>>for i in range(len(s)):
        if s[i].isdigit():
                digit_index=i
                print(digit_index)
```

1 / 3

숫자가 시작되는 첫 번째 index만 찾겠다,

```
※>>>s='C3H7'
※>>>digit_index=-1
※>>>for i in range(len(s)):
        if s[i].isdigit():
                digit_index=i
                print(digit_index)
                break
```

1

continue
for문 돌다가 중간에 continue가 있으면 for문 포함되어있고 continue 밑에 있는 거 안함. 그 다음 번 loop으로 넘어감.
continue문은 false가 나오더라도 어쨌든 다 반복하기는 함.

```
※>>>s='C3H7'
※>>>total=0
※>>>count=0
※>>>for i in range(len(s)):
        if s[i].isalpha():
                continue
        total=total+int(s[i])
        count=count+1
※>>>print ('total:',total)
※>>>print('count:',count)
```

total: 10 / count: 2

```
※>>>s='C3H7'
※>>>total=0
※>>>count=0
※>>>for i in range(len(s)):
        if s[i].isalpha():
                print('alphabet!')
        #total=total+int(s[i])
        count=count+1
※>>>print('total:',total)
※>>>print('count:'count)
```

alphabet! / alphabet! / total: 0 / count: 4

l∟total=total+int(s[i]) 쓰면 runtime 에러뜸

break, continue 의 단점:
이해하기 힘듦. break 경우 그냥 interactive program (!=쓰는 등) 으로 하는 게 더 보기 쉬움.

12. File

파일 열기 방법1
>>>file=open('<<filename>>','r')
>>>contents=file.read()
>>>print(contents)
>>>file.close()
꼭 close 해주기

파일 열기 방법2; 저절로 닫힘
>>>with open('<<filename>>','r') as file:
    contents=file.read()
>>>print(contents)

파일 한줄씩 읽기; List in list로; 방법1
>>>fileMatrix=[]
>>>with open('<<filename>>','r') as file:
    for lineContent in file:
        fileMatirx. append (lineContent. strip ('\n'). split(','))
>>>print(fileMatrix)

파일 한줄씩 읽기; List in list로; 방법2
>>>fileMatrix=[]
>>>file=open('<<filename>>','r')
>>>lineContent=file.readline()
>>>while lineContent!= '' :
    fileMatrix. append (lineContent.strip('\n').split(','))
    lineContent=file.readline()
>>>print(fileMatrix)
>>>file.close()

기존 화일에 추가하는 방법
>>>with open('<<filename>>','a') as file:
    file.write("~~~")

파일 모드
'r' : read-only. 수정은 불가능
'a' : append. 기존 화일의 끝에 추가.
'w' : write. 출력 용도. 기존 화일 있다면 그걸 제거하고 새 빈 화일 만듦.

.csv 파일은 텍스트 형태로 표 저장

file path 경로
window버튼+E -> 파일 탐색기 열기
내문서: 보통 C:\Users\jiyoung\Documents
파이썬 안 스트링에 넣을 때에는 \를 \\로 혹은 /로 쓰기
>>>path="C:\\Users~" 혹은 "C:/Users~"
>>>print(path) 하면 C:\Users~ 로 나옴.

파이썬 안에 쓸거면 폴더명에다가 파일명까지 정확하게 명시해주기 (절대경로)
그냥 파일 이름만 적어줄라면 파이썬 파일(현재 working directory)이 코드랑 같은 파일 안에 있어야만 가능 (상대경로:짧다)

Os module; 파일 위치
>>>import os
>>>os.getcwd() 하면 현재 파이썬 파일의 working directory 알려줌
>>>os.chdir('F:\\~') 하면 현재 파이썬 파일의 working directory를 F:\\~로 바꿔줌

상대경로일 때
>>>file=open('data/data1.txt','r')
하면 하위 폴더인 data 파일 안의 data1.txt 읽음
>>>file=open('../data/data1.txt','r')
하면 상위 폴더인 data 파일 안의 data1.txt 읽음
>>>file=open('../../../data/data1.txt','r')

read 함수
read 함수 안에 파라메타 넣을 수 있다.
file.read(10) 딱 열 개만 읽고 커서 멈춰있음
또 file.read() 하면 11째부터 커서가 움직임
파라메타 값은 현재 커서로부터 몇 개 읽을건지 결정하는 숫자. 돌아갈 수 절대 없다.
파일은 무조건 한 번 읽고 지나감.

readlines 함수
:처음부터 끝까지 다 읽음.
:라인별로 짤라서 리스트안의 요소들과 \n으로 반환.
>>>file=open('file_example.txt','r')
>>>contents=file.readlines()
>>>file.close()
>>>print(contents)
['~', '~', ~]

```
>>>file=open('planets.txt.','r')
>>>planets=file.readlines()
>>>file.close()
>>>planets 하면
['Mercury\n', 'Venus\n', ~] 처럼 리스트로 반환.
```

p.strip()은 앞 뒤 잘라주는 거
(줄 바꿔주는 것도 여백에 포함 됨.)

```
>>>for p in planets:
        print(p.strip()) 하면
Mercury / Venus / Earth / Mars

>>>for p in reversed(planets):
        print(p.strip()) 하면
Mars / Earth / Venus / Mercury

>>>for p in sorted(planets):
        print(p.strip()) 하면
Earth / Mars / Mercury / Venus
```

for line in file함수랑 readline 함수가 제일 많이 쓰임
len(line) 하면 그 라인에 있는 글자에다가 줄바꿈까지 포함한 갯수로 나옴
len(line.strip()) 하면 글자 수만 나옴!

```
>>>file=open('planets.txt','r')
>>>for line in file:
        print(len(line))
>>>file.close() 하면
8/6/6/5

>>>file=open('planets.txt','r')
>>>for line in file:
        print(len(line.strip()))
>>>file.close() 하면
7/5/5/4
```

텍스트 자료 파일의 전형적인 예)
제목 한 줄 (;header)
뒤에 설명
실제 데이터들 쭉

```
>>>hopedale_file=open('hopedale.txt','r')
>>>hopedale_file.readline()
```

는 한 줄만 읽는 거, 커서가 한 줄(header) 읽고 난 다음에 멈춰있음,

```
>>>data=hopedale_file.readline().strip()
```

하면 줄바꿈 없앰. #으로 시작하는지 따져보기 위해

```
>>>while data.startswith("#"):
        data=hopedale_file.readline().strip()
```

여러 줄이 될 수도 있으니까 if 말고 while 쓰기
data가 이제 22(숫자) 가진 채로 while문 빠져나옴.
!!!

~~>>>for data in hopedale_file:~~
~~print(data)~~
~~하면 29부터 print가 됨. 왜냐하면 while문 빠져나올 때 22를 print하지 않았기 때문.~~
~~!!! 사이에~~

```
>>>total_pelts=int(data)
```

써줘서 22를 지정해주기

```
>>>for data in hopedale_file:
        total_pelts=total_pelts+int(data.strip())
>>>hopedale_file.close()
>>>print('Total number of pelts:', total_pelts)
```

Local file

```
>>>hopedale_file=open('hopedale.txt','r')
>>>line=hopedale_file.readline() #header
>>>type(hopedale_file)
<class '_io.TextIOWrapper'>
>>>type(line)
<class 'str'>이다.
```

(시험x)
인터넷에 있는 파일 직접읽고 싶을 때
urllib이라는 module

```
>>>import urllib.request
>>>url='http://~'
>>>webpage=urllib.request.urlopen(url)
>>>line=webpage.readline()
>>>line=line.strip()
>>>line=line.decode('utf-8')
>>>type(webpage)
<class 'http.client.HTTPResponse'>
>>>type(line)
<class 'bytes'>
```

혹은

```
import urllib.request
url="http://~"
webpage=urllib.request.urlopen(url)
for line in webpage:
```

```
        line=line.strip()
        line=line.decode('utf-8')
        print(line)
webpage.close
```

writing files하는 법
```
>>>ourfile=open('topics.txt','w')
>>>outfile.write('Computer Science')
>>>outfile.close 하면
```
topics.txt를 'Computer Science' 로만 덮어씀.
만약에 topics.txt 파일이 없으면 topics.txt파일을 새로 만듦.

몇 글자 썼는지 알아야 되면
```
>>>outfile=open('topics.txt','w')
>>>word=outfile.write('Computer Science')
>>>outfile.close()
```
하면 word에 16이란 숫자 지정됨

'a' 모드: append. 뒤에 추가하기
```
>>>outfile=open('topics.txt','a')
>>>outfile.write('Software Engineering')
>>>outfile.close()
```
하면 Computer ScienceSofrware Engineering 처럼 그냥 딱 붙임

신규 화일에 출력하는 방법, csv 모듈
```
>>>import csv
>>>fileMatrix=[]
>>>with open('<<filename>>','r') as fileRead:
        for lineContent in fileRead:
                fileMatrix. append (lineContent.
strip('\n'). split(','))
>>>print(fileMatrix)

>>>fileMatrix[0].extend(['~','~'])
>>>lenFileMatrix=len(fileMatrix)
>>>for i in range(lenFileMatrix-1):
        i=i+1
        fileMatrix[i].extend(['_','_'])
>>>print(fileMatrix)

>>>with open('<<filename>>','w') as fileWrite:
        myWriter=csv.writer(fileWrite)
        for i in range(lenFileMatrix):
                myWriter.writerow(fileMatrix[i])
```

csv 모듈: 행렬 형태 화일 다룸

예시) 두 숫자 합치는 거
number_pairs.txt 가
1.3 3.4 / 2 4.2 / -1 1 일 때,
total.py 안에를
```
>>>def sum_number_pairs(input_file,
output_filename):
        output_file=open(output_filename,'w')
        for number_pair in input_file:
                number_pair =number_pair.strip()
                operands =number_pair.split()
                total =float(operands[0])
+float(operands[1])
                new_line ='{0}
{1}\n'.format(number_pair, total)
                output_file.write(new_line)
        output_file.close() 처럼 쓴다.
>>> import total
>>> total.sum_number_pairs (open
('number_pairs.txt','r'), 'out.txt) 하면
```
out.txt 가
1.3 3.4 4.7 / 2 4.2 6.2 / -1 1 0 으로 생성됨.

13. Data collection type

set
{}중괄호로 묶음.
들어온 순서와 관계 없이 가장 효율적 순서로 저장
id 값들의 우위 관계 없고 동등하다.
중복되는 거 없애고 정리.
empty set 은 {}말고 set()라고 해야함 꼭!
{}는 empty dictionary이다.

set([list])하면 리스트 안에 있는 것들이 세트로 저장.
```
>>>set{[2,3,2,5])
```
{2,3,4}
set 함수 안에는 list, set, range, tuple만 argument로서 집어넣을 수 있음.
set 함수 안에는 오로지 하나 이하의 파라메터만!

set method
▷S.add(v)

▷S.clear() 하면 set()됨.
▷S.difference(other) 차집합. S에서 other 뺌
▷S.intersection(other) 교집합.
▷S.issubset(other) :true or false로 나옴. S가 other의 부분집합이냐
▷S.issuperset(other) :true or false로 나옴. S가 other의 상위 집합이냐.
▷S.remove(v)
▷S.symmetric_difference(other) 대칭 차집합. 두 집합에서 교집합만 딱 뺀 것
▷S.union(other) 합집합.

set operators
▷set1.difference(set2) / set1-set2
▷set1.intersection(set2) / set1&set2
▷set1.issubset(set2) / set1<=set2
▷set1.issuperset(set2) / set1>=set2
▷set1.union(set2) / set1|set2
▷set1.symmetric_difference(set2) set1^set2

예제
```
>>>observations_file =open('observations.txt')
>>>birds_observed =set()
>>>for line in observation_files:
        bird =line.strip()
        birds_observed.add(bird)
>>>print(birds_observed)
```

tuple은 소괄호로 묶음.
empty tuple은 ()
한 개만 들어가는 tuple 만들려면 (x,)
콤마를 꼭 찍어놔야 함
리스트에서처럼 A[0], A[1:3] 등 가능
tuble은 immutable하다. tuple 안에 있는 걸 못바꾼다. (list는 mutable해서 바꿀 수 있음. string도 immutable)

multiple assignment
```
>>>10,20
```
(10,20) 바로 tuple이 됨.
```
>>>a=10,20
>>>a
(10,20)
>>>x,y=10,20
>>>x
10
>>>y
```
20이다.

multiple assignment: swap
swap : 바꾼다
>>>s1,s2=s2,s1 하면 둘이 바뀌어짐.

dictionary는 dict1 ={ 'keyname' : 4, 'keyname2' : 6} 이런 식으로. (a.k.a. map)
type은 <class 'dict'>
>>>dict1['keyname']하면/4 즉, value 반환!
unordered mutable collection of key/value pairs
empty dictionary는 {}이다.
key는 dictionary 내에 딱 한 번만 나타날 수 있다.
key는 immutable
values는 mutable

>>>A['keyname']=5 하면 A라는 dictionary에 {'keyname' : 5}가 생김
>>>del A['keyname'] 하면 그 key값과 value값도 쌍으로 같이 없어짐
>>>'keyname' in A 하면 boolean으로 확인 가능

```
>>>bird_to_observations ={'goose': 183,
'jaeger': 71}
>>>for bird in bird_to_obervations:
       print(bird, bird_to_observations[bird])
```
하면 bird는 bird_to_observation이라는 dictionary 안에 key들을 뜻하게 된다.

key는 항상 distinct 하다. 고유. 중복되지 않음. 그래서 숫자로 하지 않고 주로 str으로 key를 지정함.

dictionary operations
dictionary method
▷D.keys() 하면 key만 가지고옴.
dict_keys(['key1','key2'])처럼 반환
▷D.values() 하면 value만 가지고옴.
dict_items([v1,v2])처럼 반환
keys함수 썼을 때 가져오는 순서와 values 함수 썼을 때 가져오는 순서는 같음! 항상 pair로 섞이기 때문.
▷D.items()
dict_itmes([('key1',v1),('key2',v2)])처럼 반환

▷D.get(k) 중요! 해당 key의 value 값 반환
▷D.get(k,v) 있으면 값을 반환하고 없으면 v 반환
▷D.update(other) D 안에 other이라는 다른 dictionary를 추가함.
▷D.clear() 하면 D는 {} 됨.
▷D.pop(k) D에서 k라는 key를 없애고 그 해당 key의 value 반환. k가 D에 없다면 에러.
▷D.pop(k,v) D에서 k라는 key 없애고 그 해당 key의 value 반환. k가 D에 없다면, v 반환.
▷D.setdefault(k) D의 k의 해당 value를 반환.
▷D.setdefault(k,v) D의 k의 해당 value를 반환. k가 D에 없다면, k와 v를 D에 추가하고 v 반환.

for bird, operations in bird_to_observations.item() 이런 식으로 활용

페이지 28쪽 시험 잘!! abc 순서대로 정렬... value와 key가 뒤섞이지 않게

```
>>>observations_file =open('obervations.txt')
>>>bird_to_observations={}
>>>for line in observations_file:
        bird =line.strip()
        bird_to_observations[bird]
=bird_to_observations.get(bird,0) +1
>>>observations_file.close()
>>>for bird, observations in
bird_to_observations.items():
        print(bird, observations)

>>>sorted_birds
=sorted(bird_to_observations.keys())
>>>for bird in sorted_birds:
        print(bird, bird_to_observations[bird])
goose 5 / jaeger 2 / ~
>>>type(sorted_birds)
<class 'list'>
>>>type(bird_to_observations.keys())
<class 'dict_keys'>

inverting a dictionary
keys-> Values, values->keys
{'a':1,'b':1,'c':1} -> {1:['a','b','c']}

>>>bird_to_observations
{'goose':1, 'fulmar':1, 'jaeger':2}
>>>observations_to_birds_list ={}
>>>for bird, observations in
bird_to_observations.items():
        if observations in
observations_to_bird_list:
                observations_to_birds_list
[observations]. append(bird)
        else:
                observations_to_birds_list
[observations] =[bird]
>>>observations_to_birds_list
{1: ['goose', 'fulmar'] , 2: ['jaeger']}
>>>observations_sorted
=sorted(observations_to_birds_list.keys())
>>>for observations in observations_sorted:
        print(observations, ":" , end="")
        for bird in
observations_to_birds_list[observations]:
                print(" ", bird, end="")
        print()
1: goose fulmar / 2: jaeger
```

in 써서 있는가 없는가 확인할 때에는 key만 확인 가능. value는 무조건 false나옴.

정리!
str. immutable. ordered. 중복 가능
“string”
s= ‘’

list. mutable. ordered. 중복 가능
[1,2,5,2,3]
l=[]

tuple. immutable. ordered. 중복 가능
(1,2,5,2,3) ([‘item1’,12],[‘item2’,22])
t=()

set. mutable. non-ordered. 중복 불가
{1,2,3,5}
S=set()

dictionary. mutable. non-ordered. 중복 불가
{‘one’:1, ‘two’:2}
d={}

14, 15. Search

algorithms : set of steps
top-down design

Searching for the smallest values
마구잡이 list에서 가장 작은 숫자 두 개 찾기

```
>>>count=[809,834,~,96,102,~,476]
>>>min(counts)
96
>>>low=min(counts)
>>>min_index=counts.index(low)
>>>print(min_index)
6
counts.index.(min(counts))
```

방법1. find remove, find

```
>>>def find_two_smallest(L):
        """(list of float) -> tuple of (int,int) ~"""
        v1=min(L)
        i1=L.index(v1)
        L.remove(v1)
        v2=min(L)
        L.insert(i1,v1)
        if i2>+i1:
                i2=i2+1
        return i1,i2
        혹은
        i2=L.index(v2)
        return i1, i2
>>>x=find_two_smallest(counts)
>>>x
(6,7)
>>>a,b=find_two_smallest(counts)
>>>a
6
```

방법2. sort, identify minimums, get indices

```
>>>def find_two_smallest2(L):
        L2= L[:]
        L2.sort()
        혹은
        L2=sorted(L)
        v1=L2[0]
        v2=L2[1]
        i1=L.index(v1)
        i2=L.index(v2)
        return i1, i2
```

방법3. walk through the list

```
>>>def find_two_smallest3(L):
        if L[0] < L[1]:
                i1=0
                i2=1
                v1=L[0]
                v2=L[1]
        else:
                i1=1
                i2=0
                v1=L[1]
                v2=L[0]
        for j in range(2,len(L)):
                if L[j] < v1:
                        v2 = v1
                        v1 = L[j]
                elif L[j] < v2:
                        v2 = L[j]
         i1=L.index(v1)
         i2=L.index(v2)
         return i1, i2

>>>def find_two_smallest3(L):
        if L[0] < L[1]:
                i1=0
                i2=1
        else:
                i1=1
                i2=0
        for j in range(2,len(L)):
                if L[j] < L[i1]:
                        i2=i1
                        i1=j
                elif L[j] < L[i2]:
                        i2=j
         return i1, i2
```

timing the function
time module
perf_counter function 사용

```
>>>import time
>>>t1=time.perf_counter()
>>>#some code runs here
>>>t2=time.perf_counter()
```

Searching a list
list.index() 가 가장 간단한 search 방법
pseudo code: 가짜코드

방법1. while loop version of linear search

```
>>>def linear_while(lst, value):
    i=0
    while i != len(lst) and lst[i] != value:
        i = i+1
    if i == len(lst):
        return -1
    else:
        return i
```

방법2. for loop version of linear search

```
>>>def linear_for(lst, value):
    for i in range(len(lst)):
        if lst[i] == value:
            return i
     return -1
```

방법3. sentinel search

```
>>>def linear_sentinel(lst, value):
    lst.append(value)
    i=0
    while lst[i] != value:
        i = i+1
    lst.pop()
    if i ==len(lst):
        return -1
    else:
        return i
```

이를 바탕으로 시간 확인해보기

```
>>>import time
>>>import linear_while
>>>import linear_for
>>>import linear_sentinel
>>>def time_it(search,lst,value):
    t1=time.perf_counter()
    search(lst,value)
    t2=time.perf_counter()
    return (t2-t1)*1000
>>>L=list(range(100000001))
>>>t1_while
=time_it(linear_while.linear_search,L,10)
```

Binary search
N values can be serached in roughly log2N steps

```
>>>def binary_search(L,v):
    i=0
    j=len(L)-1
    while i != j+1:
        m =(i+j) //2
        if L[m] <v:
            i =m+1
        else:
            j =m-1
    if 0 <= i < len(L) and L[i] == v:
        return i
    else:
        return -1
```

How to test?
-value is the first item.
-value occurs twice. we want the index of the first one.
-value is in the middle of the list.
-value is the last item.
-value is the smallest one in the list.
-value is the largest one in the list.
-value isn't in the list, but larger than some and smaller than others.
-list has no items.
-list has one item.

16. Class

모든 type은 class로 구현됨.
(class가 type인 것은 아님)

instance: 어떤 class에 속해 있다는 게 더 강조된 object(객체)와 비슷한 개념
isinstance 함수: 어떤 object가 어떤 class에 속하느냐 안되어 있느냐

```
>>>isinstance('abc',str)
True
#'abc'는 str의 instance이다.
>>>isinstance(55.2,str)
False
>>>isinstance(55.2,float)
True
#55.2는 float의 instance이다.
>>>isinstance(~~, object) 하면
무조건 True
```

새로운 type을 만드는 법
보통 class의 이름은 대문자로 시작함.
보통 그 파일을 class 이름과 똑같은, 소문자로 파일 이름 저장. student.py

```
>>>class Student:
        name = ""
        id = 0
        gender = "female"
#name, id gender는 instance variable이다.
>>>a = Student()
>>>b = Student()
>>>c = Student()
#a,b,c,는 class Student의 instance이다.
>>>a.name="Harry Potter"
>>>a.id=2017103701
>>>a.gender="male"
>>>b.name="Hermione Granger"
>>>b.id=2019103722
>>>b.birthyear=1999
#하면 birthyear라는 instance variable이 쉽게 추가됨. b만 instance variable이 4개인 거고 a는 여전히 instance variable이 3개이다.
>>>c.name="Ron Weasley"
```

method 만드는 법

```
>>>class Student:
        name=""
        id=0
        gender="female"
        course=[]
        def num_courses(self):
                return len(self.course)
>>>a=Student()
>>>b=Student()
>>>c=Student()
>>>a.name="Harry Potter"
~...~
>>>c.name="Ron Weasley"
>>>a.course=["English","Programming"]
>>>b.course=["Writing","Physics","Programming"]
>>>c.course=["Programming"]
```

어떤 class에 소속되어 있는 method는 구현 시 반드시 첫번째 파라메타를 self라고 해주어야 한다!!
그리고 함수 정의할 때 반드시 한 개 이상의 파라메터가 있어야 한다!
f5 눌러서 실행 시키고 shell창에서,

```
>>>a.num_courses()
2
#객체지향적, 밖으로 빼낸 a가 자동으로 self가 됨.
self.method이름()
>>>Student.num_courses(a)
2
#객체oriented.class이름.method이름(self)
```

Inheritance 상속
super class/ sub class
parent class/ child class
모든 class의 super class, 근원 class는 object class이다.

```
>>>isinstance(~, object)
```

항상 True ; 55.2, 'abc', str, max 등 모든.

```
>>>dir(object)
['__class__','__delattr__','__dir__',~~~]
```

dir은 attributes의 list를 보여준다.
attributes는 methods, functions, variables, or other classes를 가르키는 class 안의 variables이다.

```
>>>class Book:
        """Information about a book"""
>>>type(str)
<class 'type'>
>>>type(Book)
<class 'type'>
>>>dir(Book)
['__class__','__delattr__','__dict__','__dir'__',~~~
]
#'__dict__','__module__','__qualname__','__wea
kref__' 가 추가됨. 어쨋든 object class로부터 다
상속됨.

>>>class Book:
    """Information about a book"""
    ruby_book = Book()
    ruby_book.title = "Programming Ruby"
    ruby_book.authors = ['Thomas', 'Fowler',
'Hunt']
>>>ruby_book.title
'Programming Ruby'
>>>ruby_book.authors
['Thomas', 'Fowler', 'Hunt']

>>>help(Book)
Help on class Book in module __main__:
class Book(bultins.object)
| Information about a book
| Data descriptors defined here:
| __dict__
|     dictionary for instance variables )if
defined)
| _weakref_
|     list of weak references to the object (if
defined)

Book class - method
>>>class Book:
    """Information about a book"""
    def num_authors(self):
        """(Book) -> int
        Return the number of autors of
this book. """
        return len(self.authors)
        #method 추가
>>>import book
#book은 imported module이다.
>>>ruby_book=book.Book()
#book이라는 module 안에 Book이라는 class
있다.
>>>ruby_book.title ="Programming Ruby"
>>>ruby_book.authors
=['Thomas','Fowler','Hunt']
>>>book.Book.num_authors(ruby_book)
3
#Book이라는 class 안에 num_authors라는
method 있다.
>>>ruby_book.num_authors()
3

__init__ method
>>>class Book:
    def __init__(self, title, authors, publisher,
isbn, price):
        self.title =title
        self.authors =authors[:]
        self.pubisher =publisher
        self.ISBN =isbn
        self.price =price
    def num_authors(self):
        return len(self.authors)
>>>import book
>>>ruby_book =book.Book()
Error
>>>python_book =book.Book('Practical
Programming', ['Campbell', 'Gries', 'Montojo'],
'Pragmatic Bookshelf', '078-1-93778-~', 25.0)
#차례차례 앞에서부터 지정함
>>>python_book.title
'Practical Progaamming'
>>>python_book.authors
['Campbell','Gries','Montojo']
>>>python_book.price
25.0
>>>python_book.num_authors()
3
```

Constructor
book이라는 module은 single statement를 포함한다.
__init__이라는 method는 Book이라는 object가 생성될 때 새로운 object를 initialize하기 위해 불러진다. ->constructor

__init__ method -refined
```
>>>class Book:
        def __init__(self, title="", authors=[], publisher="", isbn="0", price=10.0):
                self.title =title
                self.authors =authors[:]
                self.publisher =publisher
                self.ISBN =isbn
                self.pice =price
        def num_authors(self):
                return len(self.authors)
>>>import imp
>>>imp.reload(book)
>>>ruby_book =book.Book()
>>>ruby_book.title
''
>>>python_book =book.Book('Practical Programming', ~~)
>>>python_book.title
'Practical Programming'
```

__str__ method
```
>>>def __str__(self):
        rep = "Title: {0}\n Authors: {1}\n Publisher: {2}\n ISBN: {3}\n Price: {4}" .format(self.title, self.authors, self.publisher, self.ISBN, self.price)
        return rep
>>>print(python_book)
<book.Book object at 0x00000230A~>
>>>imp.reload(book)
>>>python_book =book.Book('Practical Programming', ~)
>>>print(python_book)
Title: Practical Programming
Authors: ['Campbell','Gries','Montojo']
Publisher: Pragmatic Bookshelf
ISBN: 978-1-93~~
Price: 25.0
```

__eq__ method
```
>>>python_book_1 =book.Book('P~)
>>>python_book_2 =book.Book('P~)
>>>python_book_1 == python_book_2
False
>>>python_book_1 == python_book_1
True
>>>python_book_2 == python_book_2
True
>>>def __eq__(self,other):
        return self.ISBN == other.ISBN
>>>python_book_1 =book.Book('P~)
>>>python_book_2 =book.Book('P~)
>>>survival_book =book.Book('New~)
>>>python_book_1 ==python_book_2
True
>>>python_book_1 == survival_book
False
```

Override vs. overload
상속 받은 걸 고쳐쓰는 거, oberride. 자식 class 수정하는 것. subclass 안에 새 버전을 defining 해줌으로써 inherited method를 override 할 수 있다. 이것은 inherited method를 더이상 못쓰게 대체하는 것이다.
overloading은 한 class 안에 2개 이상의 method가 같은 method 이름을, 그러나 다른 parameters를 가질 때 일어난다. 함수가 overloading 되어 있다: 파라메터 개수가 달라도 다른 대로 동작 가능하다. (상속이랑 전혀 관련 없)

Lookup rules for a method call
현재 object의 class를 보라. 옳은 이름을 가진 method를 찾으면, 써도 됨. 없으면, superclass를 찾아라.

Inheritance: example
```
>>>class Member:
        """A member of a university"""
        def __init__(self, name, address, email):
                """(Member, str, str, str) -> NoneType
                Create a new member named name, with home address and email address"""
                self.name =name
```

```
            self.address =address
            self.email =email

>>>class Faculty(Member):
      """A faculty member at a university"""
      def __init__(self, name, address, email,
faculty_num):
            super().__init__(name, address,
email)
            self.faculty_number =faculty_num
            self.courses_teaching =[]

>>>class Student(Member):
      def __init__(self, name, address, email,
student_num):
            super().__init__(name, address,
email)
            self.student_number =student_num
            self.courses_taken =[]
            self.courses_taking =[]

>>>snape =Faculty('Severus Snape', 'Seoul',
'snape@khu.ac.kr', '1234')
>>>snape.name
'Severus Snape'
>>>snape.email
'snape@khu.ac.kr'
>>>snape.faculty_number
'1234'

>>>harry =Student('Harry Potter', 'London',
'hpotter@khu.ac.kr', '4321')
>>>harry.name
'Harry Potter'
>>>harry.email
'hpotter@khu.ac.kr'
>>>harry.student_number
'4321'

Add features to the superclass
>>>class Member:
      def __init__(self, name, address, email):
            self.name =name
            self.address =address
            self.email =email
      def __str__(self):
            rep ="{}\n{}\n{}" .format(self.name,
self.address, self.email)
            return rep
>>>snape =Faculty('Severus Snape', 'Seoul',
'snape@khu.ac.kr', '1234')
>>>harry =Student('Harry Potter', 'London',
'hpotter@khu.ac.kr', '4321')
>>>print(snape)
Severus Snape
Seoul
snape@khu.ac.kr
>>>print(harry)
Harry Potter
London
hpotter@khu.ac.kr
>>>str(harry)
'Harry Potter\nLondon\nhpotter@khu.ac.kr'

Add features to the subclass
>>>class Faculty(Member):
      def __init__(self, name, address, email,
faculty_num):
            super().__init__(name, address,
email)
            self.faculty_number =faculty_num
            self.courses_teaching =[]
      def __str__(self):
            member_string =super().__str__()
            rep ="{}\n{}\nCourses:{}"
.format(member_string, self.faculty_number,
self.courses_teaching)
            return rep
>>>snape =Faculty('Severus Snape', 'Seoul',
'snape@khu.ac.kr', '1234')
>>>print(snape)
Severus Snape
Seoul
snape@khu.ac.kr
1234
Courses:[]
```

실습자료에 있는 거 정리

#1 class 이해하기

```
>>>class Student:
    id = 0
    name = ''
    def setId(self, givenID):
        self.id = givenID
    def getId(self):
        return self.id
    def setName(self, givenName):
        self.name = givenName
    def getName(self):
        return self.name
>>>student1 = Student()
>>>student2 = Student()

>>>student1.setId(20190001)
>>>student1.setName("Harry Potter")
>>>print(student1.getId())
20190001
>>>print(student1.getName())
Harry Potter
>>>student2.setId(20190002)
>>>student2.setName("Hermione Granger")
>>>print(student2.getId())
20190002
>>>print(student2.getName())
Hermione Granger
```

#2 생성자(constructor, __init__()) 이해하기

```
>>>class Student:
    def __init__(self, givenID, givenName):
        self.id = givenID
        self.name = givenName
    def setId(self, givenID):
        self.id = givenID
    def getId(self):
        return self.id
    def setName(self, givenName):
        self.name = givenName
    def getName(self):
        return self.name

>>>student1 = Student(20190001, "Harry Potter")
>>>student2 = Student(20190002, "Hermione Granger")
>>>print(student1.getId())
20190001
>>>print(student1.getName())
Harry Potter
>>>print(student2.getId())
20190002
>>>print(student2.getName())
Hermione Granger
```

#3 __str__ 적용 이해하기

```
>>>class Student:
    def __init__(self, givenID, givenName):
        self.id = givenID
        self.name = givenName
    def setId(self, givenID):
        self.id = givenID
    def getId(self):
        return self.id
    def setName(self, givenName):
        self.name = givenName
    def getName(self):
        return self.name
    def __str__(self):
        msg = "id:{}, name:{}".format(self.id, self.name)
        return msg
>>>student1 = Student(20190001, "Harry Potter")
>>>student2 = Student(20190002, "Hermione Granger")
>>>print(student1.getId())
20190001
>>>print(student1.getName())
Harry Potter
>>>print(student2.getId())
20190002
>>>print(student2.getName())
Hermione Granger
```

#4 class 변수와 객체(인스턴스) 변수 이해하기

```
>>>class Student:
    # Class variables
    countStudent = 0
    def __init__(self, givenID, givenName):
        # Instance (or Object) variables
        self.id = givenID
        self.name = givenName
        Student.countStudent =
Student.countStudent + 1
    def setId(self, givenID):
        self.id = givenID
    def getId(self):
        return self.id
    def setName(self, givenName):
        self.name = givenName
    def getName(self):
        return self.name
    def __str__(self):
        msg = "id:{}, name:{}".format(self.id,
self.name)
        return msg
    def getNumOfStudent():
        return Student.countStudent

>>>print(Student.getNumOfStudent())
0
>>>student1 = Student(20190001, "Harry
Potter")
>>>student2 = Student(20190002, "Hermione
Granger")
>>>print(student1)
id:20190001, name:Harry Potter
>>>print(student2)
id:20190002, name:Hermione Granger
>>>print(Student.getNumOfStudent())
2
```

#5 Private Attribute 이해하기

```
>>>class Student:
    # Class variables
    __countStudent = 0
    def __init__(self, givenID, givenName):
        # Instance (or Object) variables
        self.__id = givenID
        self.__name = givenName
        Student.__countStudent =
Student.__countStudent + 1
    def setId(self, givenID):
        self.__id = givenID
    def getId(self):
        return self.__id
    def setName(self, givenName):
        self.__name = givenName
    def getName(self):
        return self.__name
    def __str__(self):
        msg = "id:{}, name:{}".format(self.__id,
self.__name)
        return msg
    def getNumOfStudent():
        return Student.__countStudent
>>>print(Student.__countStudent)
ERROR
AttributeError: type object 'Student' has no
attribute '__countStudent'
```

#6 상속(Inheritance) 이해하기

```
>>>class GraduatedStudent(Student):
    def __init__(self, givenId, givenName,
givenYear):
        self.__graduatedYear = givenYear
        super().__init__(givenId, givenName)
    def __str__(self):
        msg = super().__str__() + ",
graduation:{}".format(self.__graduatedYear)
        return msg
>>>student1 = GraduatedStudent(20190001,
"Harry Potter", 2023)
>>>print(student1)
id:20190001, name:Harry Potter,
graduation:2023
```

#7 포함관계 (has-a relationship) 이해하기

```
>>>class Department:
    def __init__(self):
        memberStudent = Student()
```