MIT 코스웨어

C3 | 스트링 편집과 검색 알고리즘

스트링은 대소문자를 구별하는 char의 seq이다.
compare가 가능하며, len(s)를 통해 string의 길이 파악

스트링을 편집하는 방법은 **인덱싱**과 **슬라이싱**이 있다.
단, 스트링은 edit이 불가능한 immutable한 데이터 타입!
컴퓨터에서는 모든 순서는 0에서 시작하며,
s[0], s[1] 형식으로 특정 char를 추출할 수 있다.
s[-1], s[-2] 형식으로 string의 역순을 파악할 수 있다.

스트링과 루프를 같이 활용하는 경우,
"for char in s" 같은 형식으로 loop가 가능!
이중 for문을 통해서 common letter를 찾을 수 있다.

```
s1 = "mit u rock"
s2 = "i rule mit"
if len(s1) == len(s2):
    for char1 in s1:
        for char2 in s2:
            if char1 == char2:
                print("common letter")
                break
```

이중 for문은 1-1, 1-2 2-1, 2-2

검색에는 다음의 **3가지 방법**이 존재한다.

1) guess-and-check

하나씩 전부 체크해보는 형식으로 진행
만약 일정 범위를 넘어서버리면 break하는 형식으로!

```
cube = 8
for guess in range(abs(cube)+1):
    if guess**3 >= abs(cube):
        break
if guess**3 != abs(cube):
    print(cube, 'is not a perfect cube')
else:
    if cube < 0:
        guess = -guess
    print('Cube root of '+str(cube)+' is '+str(guess))
```

2) approximate solution

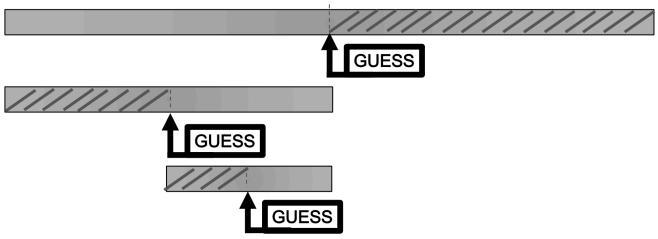
하나씩 체크해서 추측을 하되, 그 범위를 정한다.
범위가 좁아질 수록 값의 정확도가 더 높아진다.

≒ increment

```
cube = 27
epsilon = 0.01
guess = 0.0
increment = 0.0001
num_guesses = 0
while abs(guess**3 - cube) >= epsilon  and guess <= cube :
    guess += increment
    num_guesses += 1
print('num_guesses =', num_guesses)
if abs(guess**3 - cube) >= epsilon:
    print('Failed on cube root of', cube)
else:
    print(guess, 'is close to the cube root of', cube)
```

3) bisection search

가장 효율적인 방법으로, 시간 복잡도가 N(logn)이다.
recursive process/iteration으로 검색 범위 1/2씩!




```
cube = 27
epsilon = 0.01
num_guesses = 0
low = 0
high = cube
guess = (high + low)/2.0
while abs(guess**3 - cube) >= epsilon:
    if guess**3 < cube :
        low = guess
    else:
        high = guess
    guess = (high + low)/2.0
    num_guesses += 1
print 'num_guesses =', num_guesses
print guess, 'is close to the cube root of', cube
```

loop문을 시작하기 전에 low, high, guess를 assign 시켜야