파이썬을 파이썬스럽게

a = 7

b = 5

print( \*divmod(a, b) )

이건 print(a//b, a%b)와 같은 문법 -> 큰 숫자일땐 divmod가 작은 숫자일땐 후자가 더 좋은 방법

num = '3212'

base = 5

answer = int(num, base)

기본적으로 int는 base= 10이 default값 base값을 바꿔주면 해당 스트링을 원하는 진법으로 바꿔줌

s = 'abc'

n = 7

s.ljust(n) *# 좌측 정렬*

s.center(n) *# 가운데 정렬*

s.rjust(n) *# 우측 정렬*

길이가 n인 문자열에서 s를 원하는 걸로 정렬

파이썬은 이런 데이터를 상수(constants)로 정의해놓았습니다.

import string

string.ascii\_lowercase *# 소문자 abcdefghijklmnopqrstuvwxyz*

string.ascii\_uppercase *# 대문자 ABCDEFGHIJKLMNOPQRSTUVWXYZ*

string.ascii\_letters *#대소문자 모두 abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ*

string.digits *# 숫자 0123456789*

파이썬의 sorted를 사용해보세요. 반복문이나, deepcopy 함수를 사용하지 않아도 새로운 정렬된 리스트를 구할 수 있습니다.

list1 = [3, 2, 1]

list2 = sorted(list1)

mylist = [ 1,2,3 ]

new\_list = [ 40, 50, 60 ]

for i in zip(mylist, new\_list):

print (i)

(1, 40)

(2, 50)

(3, 60)

zip은 원래 이런 식으로 2개의 리스트에 대해 같이 순서대로 보여줄 때 쓰는데

mylist = [ [1,2,3], [4,5,6], [7,8,9] ]

new\_list = list(map(list, zip(\*mylist)))

하나의 2차원 배열에 대해서도 이렇게 쓸 수 있음

파이썬의 zip 함수와 dict 생성자를 이용하면 **코드 단 한 줄**로, 두 리스트를 합쳐 딕셔너리로 만들 수 있습니다.

animals = ['cat', 'dog', 'lion']

sounds = ['meow', 'woof', 'roar']

answer = dict(zip(animals, sounds)) *# {'cat': 'meow', 'dog': 'woof', 'lion': 'roar'}*

어메이징

알고리즘 문제를 풀다보면, 시퀀스의 멤버들을 하나의 string으로 이어붙여야 할 때가 있습니다

예시)

* 문자열 배열 ['1', '100', '33']을 이어 붙여 문자열 '110033' 만들기
* 정수형 튜플 (3, 22, 91)을 이어붙여 문자열 '32291' 만들기
* my\_list = ['1', '100', '33']
* answer = ''.join(my\_list)
* [itertools.product](https://docs.python.org/3/library/itertools.html#itertools.product)를 이용하면, for 문을 사용하지 않고도 곱집합을 구할 수 있습니다.
* import itertools
* iterable1 = 'ABCD'
* iterable2 = 'xy'
* iterable3 = '1234'
* itertools.product(iterable1, iterable2, iterable3)
* **2차원리스트를 1차원 리스트로 만들기**
* my\_list = [[1, 2], [3, 4], [5, 6]]
* *# 방법 1 - sum 함수*
* answer = sum(my\_list, [])
* *# 방법 5 - reduce 함수 이용1*
* from functools import reduce
* list(reduce(lambda x, y: x+y, my\_list))

[itertools.permutation](https://docs.python.org/3/library/itertools.html#itertools.permutations)를 이용하면, for문을 사용하지 않고도 순열을 구할 수 있습니다.

import itertools

pool = ['A', 'B', 'C']

print(list(map(''.join, itertools.permutations(pool)))) *# 3개의 원소로 수열 만들기*

print(list(map(''.join, itertools.permutations(pool, 2)))) *# 2개의 원소로 수열 만들기*

※ 조합은 [itertools.combinations](https://docs.python.org/3/library/itertools.html" \l "itertools.combinations" \t "_blank)를 사용해서 구할 수 있습니다. 사용법은 permutations와 비슷해요!

파이썬의 list comprehension을 사용하면 한 줄 안에 for 문과 if 문을 한 번에 처리할 수 있습니다.

mylist = [3, 2, 6, 7]

answer = [ i\*\*2 for i in mylist if i %2 == 0]

각 원소가 몇 번 등장했는지

import collections

my\_list = [1, 2, 3, 4, 5, 6, 7, 8, 7, 9, 1, 2, 3, 3, 5, 2, 6, 8, 9, 0, 1, 1, 4, 7, 0]

answer = collections.Counter(my\_list)

파이썬의 [bisect.bisect](https://docs.python.org/3.6/library/bisect.html" \l "bisect.bisect" \t "_blank) 메소드를 사용하면 이 코드를 간략하게 만들 수 있습니다.

import bisect

mylist = [1, 2, 3, 7, 9, 11, 33]

print(bisect.bisect(mylist, 3))

1. 이진 탐색 - 오름차순으로 정렬된 리스트에서 특정한 값의 위치를 찾는 알고리즘. 검색 속도가 아주 빠르다. [위키 백과에서 발췌](https://ko.wikipedia.org/wiki/%EC%9D%B4%EC%A7%84_%EA%B2%80%EC%83%89_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98) [↩](https://programmers.co.kr/learn/courses/4008/lessons/13173#fnref1)

\*\*\* 정렬된 리스트에 대해서 할 것!