

“

파이썬 심화

모듈



“

모듈 기초

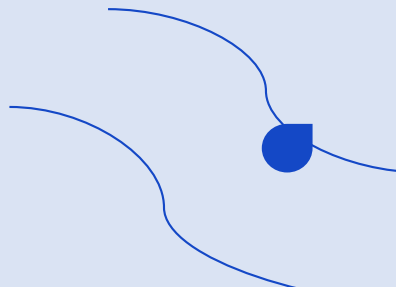
”





모듈이란?

- 모듈은 함수, 변수, 클래스들을 모아놓은 파일
- 프로그램에서 모듈 파일에 있는 내용을 가져와 사용
- 파이썬은 접착언어로 많은 모듈이 제공됨





모듈 가져오기

- 모듈을 가져오기 (import)
 - <모듈명>: 사용할 모듈의 이름 (random, os, re, ...)
 - <함수명>: 사용할 함수의 이름

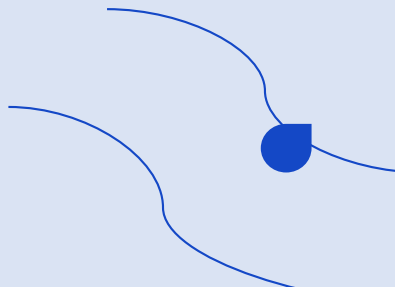
```
import <모듈명1>
from <모듈명2> import <함수명1>, <함수명2>, ...
from <모듈명3> import *      # <함수명3>이 정의된 모듈일 경우
...

<모듈명1>.<함수>(...)      # <모듈명1>을 기재 필수
<함수명1>(...)            # <모듈명2>를 기재할 필요가 없음
<함수명3>(...)            # <모듈명3>을 기재할 필요가 없음
```



모듈 만들기

- 모듈 생성 과정
 - 일반적인 파이썬 파일 생성 (예 : abc.py라고 저장)
 - 해당 파일을 불러올 파이썬 파일에서 모듈 불러오기 (예 : import abc)
 - 해당 파일에서 사용 (예 : abc.foo(...))
- 사용자 모듈 사용시 주의점
 - 모듈 파일에 실행되는 명령문이 없도록 작성
 - 또는 if __name__ == "__main__": 을 이용해서 해당 명령문들을 격리





“

itertools

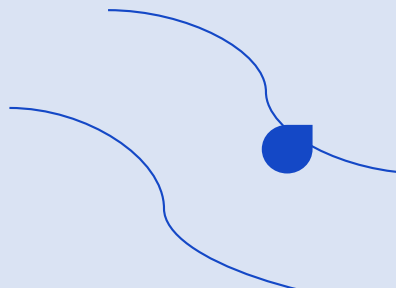
”





itertools

- 반복되는 수행에 대한 것을 반복함수로 처리
- 경우의 수를 계산하는 것 외에 모든 경우를 처리할 때 유용
 - `product(p, q)` : p에 있는 아이템과 q에 있는 아이템들을 카테시안 곱 수행
 - `accumulate(p)` : p에 있는 아이템을 차례대로 더함
 - `permutations(p[, r])` : p에 있는 아이템을 r 길이만큼 순열
 - `combinations(p, r)` : p에 있는 아이템을 r만큼 선택하는 조합





Itertools 예제

- 친구 리스트에서 두명씩 짝짓는 경우를 모두 보여라.

```
from itertools import combinations

friends = [ "홍길동", "성춘향", "이몽룡" ]

for k in combinations(friends, 2):
    print(k)
```

```
('홍길동', '성춘향')
('홍길동', '이몽룡')
('성춘향', '이몽룡')
```




“

random

”





random

- random 모듈은 난수를 발생시키는 모듈로 시뮬레이션, 테스트 등에서 사용
 - seed(<시드값>) : 난수를 발생시키는 <시드값> 설정, <시드값>이 같으면 항상 같은 난수 시퀀스 발생
 - randrange(stop) : 0~stop-1 수를 발생 (정수)
 - randrange(start, stop[, step]) : start~stop-1 수를 발생시킴 (step이 지정된 경우 건너뛰는 수는 발생 안됨)
 - choice(list) : list에서 한 개를 임의로 선택
 - shuffle(list) : list에 있는 아이템들을 임의로 섞음 (튜플, 문자열에서 허용 안됨)





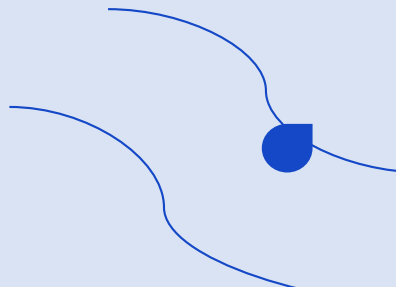
임의의 수 리스트 만들기

- random 모듈을 이용해서 임의의 수 리스트를 만들기

```
import random
```

```
x = [ random.randrange(1, 101) for _ in range(10) ]
```

```
print(x)
```





“

heapq

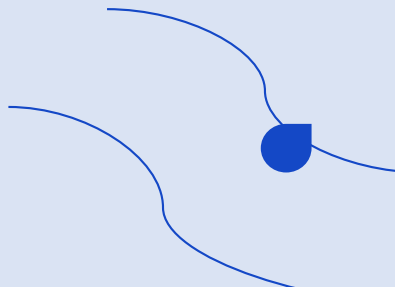
”





heapq

- heapq 모듈은 별도의 자료구조를 사용하지 않고 리스트를 이용해서 최소힙(min heap)을 사용할 수 있도록 제공
 - heapify(list) : list를 힙 구조로 생성
 - heappush(heap, x) : heap에 x 아이템을 추가
 - heappop(heap) : heap에서 제일 작은 아이템 가져오기



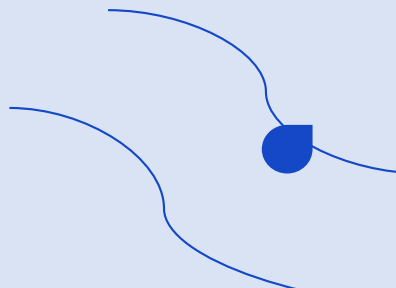


정렬하기

- 리스트를 정렬하는 프로그램 작성

```
import heapq

h = list(map(int, "숫자입력 : "))
heapq.heapify(h)
print([ heapq.heappop(h) for _ in range(len(h)) ])
```





“

bisect

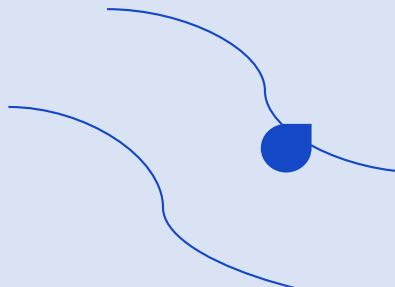
”





bisect

- bisect 모듈은 정렬된 리스트에서 원하는 아이템을 끼어넣기할 위치를 이진탐색으로 찾는다.
 - bisect_left(list, x) : list에서 x를 끼어넣기할 위치를 왼쪽에서 찾는다.
 - bisect_right(list, x) : list에서 x를 끼어넣기할 위치를 오른쪽에서 찾는다.
 - x의 값과 같은 값이 list에 존재하지 않는다면, 같은 결과를 반환함에 주의





bisect를 이용한 예제

- 이진탐색을 이용한 예제

```
from bisect import *  
  
a = [ 1, 2, 2, 4, 5 ]  
print(bisect_left(3))  
print(bisect_right(3))  
print(bisect_left(4))  
print(bisect_right(4))
```



3
3
3
4



“

실습

”





재귀함수로 순열 구하기

- 재귀함수를 이용하여 리스트의 내용을 차례대로 순열로 출력하는 프로그램 작성
 - 리스트의 내용을 입력
 - 순열 출력할 개수를 입력 (리스트의 아이템 개수보다 작아야함)

리스트의 내용 입력 : 홍길동 성춘향 이몽룡

표시할 개수 입력 : 2

('홍길동', '성춘향')

('홍길동', '이몽룡')

('성춘향', '홍길동')

('성춘향', '이몽룡')

...



Itertools로 구현

- 앞에서 한 실습 내용을 itertools.permutations(...) 함수를 이용하여 구현

리스트의 내용 입력 : 홍길동 성춘향 이몽룡

표시할 개수 입력 : 2

('홍길동', '성춘향')

('홍길동', '이몽룡')

('성춘향', '홍길동')

('성춘향', '이몽룡')

...





가장 긴 길이의 증가하는 부분 수열

- 여러 개의 수가 주어졌을 때, 해당 수들의 순서를 지키는 부분 수열 중 가장 길이가 긴 경우의 길이를 구하세요
 - (1, 2, 1, 3, 1, 5)인 경우 가장 긴 길이의 증가하는 부분 수열은 (1, 2, 1, 3, 1, 5)이고, 이때의 길이는 4이다.
 - 이 문제를 bisect를 이용해서 풀어보도록 한다.
 - 참고 : <https://www.acmicpc.net/problem/12015>

