

---

## 숙제2 –수학함수, 리스트, 다중리스트, 튜플, 세트, 딕셔너리


---

**\*2.14** (기하: 삼각형의 넓이) 삼각형의 세 꼭짓점 좌표  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$ 를 입력하고 삼각형의 넓이를 출력하는 프로그램을 작성하시오.

$$s = (\text{변1} + \text{변2} + \text{변3})/2$$

$$\text{넓이} = \sqrt{s(s - \text{변1})(s - \text{변2})(s - \text{변3})}$$

다음은 프로그램의 실행 예이다.

삼각형의 세 꼭짓점을 입력하세요: 1.5, -3.4, 4.6, 5, 9.5, -3.4   
삼각형의 넓이는 33.6 입니다.

**\*3.2** (기하학: 대권 거리) 구의 표면에서 두 점 사이의 거리를 대권 거리(great circle distance)라고 한다.  $(x_1, y_1)$ 과  $(x_2, y_2)$ 를 두 점의 지리학적인 위도와 경도라고 하자. 두 점 사이의 대권 거리는 다음 공식을 사용하여 계산할 수 있다.

$$d = \text{반지름} \times \arccos(\sin(x_1) \times \sin(x_2) + \cos(x_1) \times \cos(x_2) \times \cos(y_1 - y_2))$$

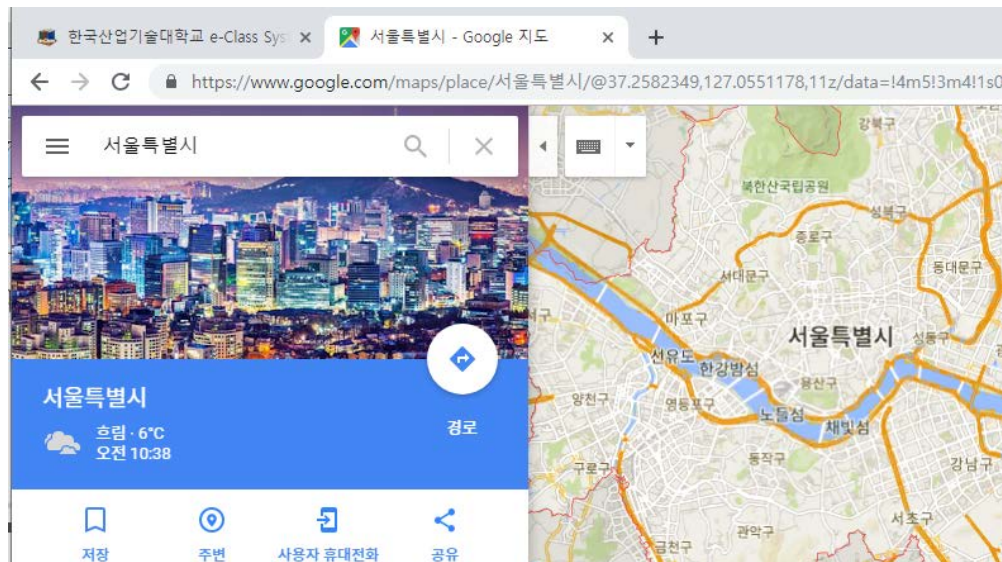
사용자로부터 지구에서 두 점에 대한 위도와 경도를 60분법 단위로 입력하게 하고 두 점의 대권 거리를 출력하는 프로그램을 작성하시오. 지구의 평균 반지름은 6,370.01km이다. 파이썬은 삼각함수에 라디안(radian)을 사용하기 때문에 `math.radians` 함수를 사용할 때 각도를 라디안으로 변환하는 것을 주의해야 한다. 이 공식에서 위도와 경도는 북향과 서향에 대한 각이다. 남향과 동향에 대한 각을 나타내기 위해서는 음수를 사용한다. 실행 예는 다음과 같다.

첫 번째 점(위도와 경도)을 60분법 각으로 입력하세요: 39.55, -116.25

두 번째 점(위도와 경도)을 60분법 각으로 입력하세요: 41.5, 87.37

두 점 사이의 거리는 10691.79183231593 km입니다.

**\*3.3** (지리학: 넓이 추정하기) 전라남도 광주, 경상남도 부산, 강원도 강릉 그리고 서울 특별시에 대한 GPS 위치를 <http://www.gps-data-team.com/map/>에서 찾고 이들 네 도시에 의해 둘러 쌓여진 넓이의 추정 넓이를 계산하시오.  
(힌트: 두 도시의 거리를 계산하기 위해 프로그래밍 연습문제 3.2의 공식을 사용하시오. 다각형을 두 삼각형으로 분할하고 삼각형의 넓이를 계산하기 위해 프로그래밍 연습문제 2.14의 공식을 사용하시오.)



광주광역시/@35.1768201,126.7735892

부산광역시/@35.1645701,129.0015892

강원도+강릉시/@37.7637326,128.8824475

서울특별시/@37.565289,126.8491259

- 3-6. (ASCII 코드의 문자 찾기) ASCII 코드(0 부터 127 사이의 정수)를 입력 받고 이에 대한 문자를 출력하는 프로그램을 작성하시오. 예를 들면, 사용자가 97을 입력하면 프로그램은 a를 출력한다. 다음은 프로그램의 실행 예이다.

ASCII 코드를 입력하세요:

69

Enter

문자는 E입니다.

- 3-7. (랜덤 문자) time.time() 함수를 이용하여 랜덤으로 대문자를 출력하는 프로그램을 작성하시오.

- 3-8. (금융 애플리케이션: 급여) 다음 정보를 읽고 급여 명세서를 출력하는 프로그램을 작성하시오.

사원 이름(예, 정용제) 주당 근무시간(예 40)

시간 당 급여(예, 9750) 원천 징수 세율(예, 20%) 주민 세율(예, 9%)

- 다음은 프로그램의 실행 예이다.

사원 이름을 입력하세요: 정용제 Enter  
주 당 근무시간을 입력하세요: 40 Enter  
시간 당 급여를 입력하세요: 9075 Enter  
원천징수세율을 입력하세요: 0.2 Enter  
지방세율을 입력하세요: 0.09 Enter

사원 이름: 정용제

주당 근무시간: 40

임금: 9075

총 급여: 363000

공제:

원천징수세(20.0%): 72600

주민세(9.0%): 32670

총 공제: 105270

공제 후 급여: 257730

- 10-2. (입력된 숫자 역순 정렬하기) 정수 리스트를 읽고, 읽은 순서의 역순으로 정수를 출력하는 프로그램을 작성하시오.
- 10-3. (숫자의 빈도수) 1과 100 사이의 정수를 입력 받고 각 정수의 빈도수를 세는 프로그램을 작성하시오.

1과 100 사이의 정수를 입력하세요:

2 5 6 5 4 3 23 43 2

Enter

2 - 2번 나타납니다.

3 - 1번 나타납니다.

4 - 1번 나타납니다.

5 - 2번 나타납니다.

6 - 1번 나타납니다.

23 - 1번 나타납니다.

43 - 1번 나타납니다.

- 10-4. (점수 분석하기) 지정되지 않은 개수만큼 정수를 읽어 들이고 얼마나 많은 점수가 평균보다 크거나 같은지 또는 얼마나 많은 점수가 평균 미만인지를 판단하는 프로그램을 작성하시오. 입력할 숫자는 한 칸 씩 띄어서 한 행에 입력한다.
- 10-5. (고유 숫자 출력하기) 한 칸 씩 띄어서 한 행으로 숫자들을 읽어 들이고, 중복 없는 고유 숫자만을 출력해 주는 프로그램을 작성하시오(예를 들어, 숫자가 여러 번 나타나면, 단 한 번만 출력한다). (힌트: 모든 숫자들을 읽고 list1에 저장한다. 그리고 새로운 리스트인 list2를 생성한다. list1에 있는 숫자를 list2에 추가하는데, 만일 어떤 숫자가 list2에 이미 존재하면 그 숫자는 추가하지 않는다.) 다음은 프로그램 실행 예이다.

10 개의 숫자를 입력하세요: 1 2 3 2 1 6 3 4 5 2

Enter

중복을 제거한 고유한 숫자: 1 2 3 6 4 5



- 
- Ref. 10-2. `lst = [30, 1, 12, 14, 10, 0]`가 주어 졌을 때 `lst`에 몇 개의 원소가 있는가? `lst`의 첫 번째 원소의 인덱스는? `lst`의 마지막 원소의 인덱스는? `lst[2]`의 값은? `lst[-2]`의 값은?
  - Ref. 10-3. `lst = [30, 1, 2, 1, 0]`가 주어 졌을 때 다음 명령문을 각각 적용한 후에 생성되는 리스트는? 코드의 각 라인은 서로 독립적이다.

```
lst.append(40)
lst.insert(1, 43)
lst.extend([1, 43])
lst.pop(1)
lst.pop()
lst.sort()
lst.reverse()
random.shuffle(lst)
```

- 
- Ref. 10-4. lst = [30, 1, 2, 1, 0]가 주어 졌을 때, 다음 명령문 각각의 반환 값은?

lst.index(1)

lst.count(1)

len(lst)

max(lst)

min(lst)

sum(lst)

- Ref. 10-5. list1 = [30, 1, 2, 1, 0]와 list2 = [1, 21, 13]가 주어 졌을 때, 다음 명령문 각각의 반환 값은?

list1 + list2

2 \* list1

list2 \* 2

list1[1 : 3]

list1[3]

---

**\*10.8** (가장 작은 원소의 인덱스 찾기) 정수 리스트에서 가장 작은 원소의 인덱스를 반환하는 함수를 작성하시오. 만약 그러한 원소가 두 개 이상이라면, 그 중에 가장 작은 인덱스를 반환한다. 다음의 헤더를 사용하라.

```
def indexOfSmallestElement(lst):
```


사용자로부터 정수 리스트를 입력받고, 위 함수를 호출하여 가장 작은 원소의 인덱스를 찾아서 그 인덱스를 출력하는 테스트 프로그램을 작성하시오.


---

**\*\*10.15** (정렬되었는가?) 리스트 내의 원소들이 이미 오름차순으로 정렬되어 있다면 `True` 값을 반환하는 다음의 함수를 작성하시오.

```
def isSorted(lst):
```

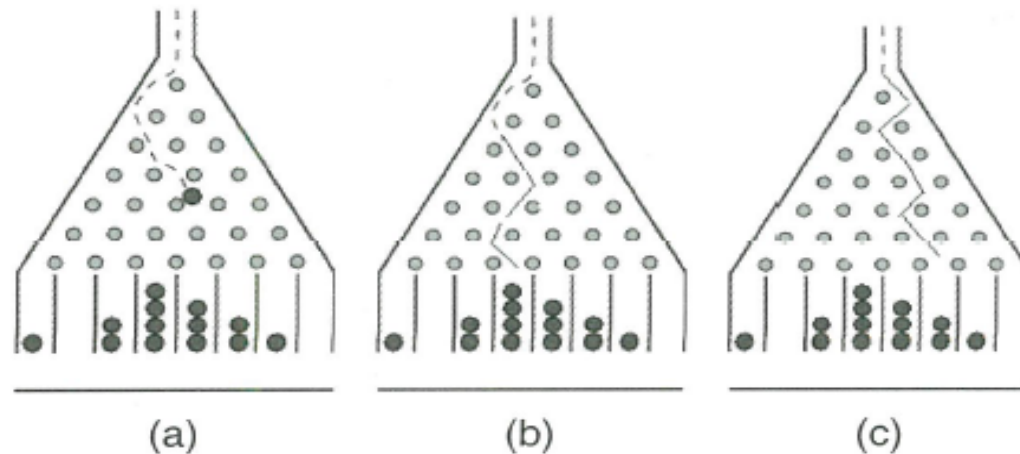
사용자로부터 정수 리스트를 입력받고, 이 리스트 내의 정수들의 정렬 여부를 출력하는 테스트 프로그램을 작성하시오

리스트를 입력하세요: 1 1 3 4 4 5 7 9 10 30 11   
리스트는 정렬되어 있지 않습니다.

리스트를 입력하세요: 1 1 3 4 4 5 7 9 10 30   
리스트는 이미 정렬되어 있습니다.

**\*\*\*10.19** (게임: 콩 기계) 오엽배열(quincunx) 혹은 골턴 박스(Galton box)로 알려진 콩 기계(bean machine)는 영국의 과학자 Francis Galton 경의 이름을 딴 통계 실험기구이다. 이 실험기구는 그림 10.15에서 볼 수 있듯이 삼각형 보드와 이 보드에 골고루 분포된 못들로 구성되어 있다.

보드 입구에서 공을 떨어뜨리면, 공이 내려가면서 못을 건드리게 되는데 50%의 기회로 왼쪽 혹은 오른쪽으로 떨어진다. 내려간 공은 결국 보드 바닥 부분의 슬롯에 쌓이게 된다.



**[그림 10.15]** 각 공들은 랜덤하게 경로를 선택하여 슬롯으로 떨어진다.

콩 기계를 시뮬레이션하는 프로그램을 작성하시오. 사용자로부터 공의 수와 기계 내의 슬롯 수를 입력받아야 한다. 공의 경로를 출력하면서 공이 떨어지는 과정을 시뮬레이션하시오. 예를 들어, 그림 10.15(b)에서 공의 경로는 LLRRLLR이고, 그림 10.15(c)에서 공의 경로는 RLRRLRR이다. 슬롯 내 공들의 최종 축적 상태를 막대그래프로 출력하시오. 다음은 프로그램의 실행 예이다.

떨어뜨릴 공의 개수를 입력하세요: 5

콩 기계의 슬롯 개수를 입력하세요: 8

LRLRLRR

RRLLLLR

LLRRLRR

RRLLLLL

LRLRRLR

0

0

000


(힌트: `slots`라는 이름의 리스트를 생성하시오. `slots` 내의 각 원소는 슬롯 내 공의 개수를 저장한다. 각각의 공은 경로를 따라서 슬롯으로 떨어진다. 어떤 경로에서 R의 개수는 볼이 떨어지는 슬롯 위치와 같다. 예를 들어, LRLRLRR 경로는 R이 4번 나왔으므로 공은 `slots[4]`로 떨어지며, RRLLLLL 경로는 `slots[2]`로 공이 떨어진다.)


---


**\*11.1** (열별 원소 합하기) 다음 함수 헤더를 사용하여 행렬의 특정 열에 포함된 모든 원소의 합계를 반환하는 함수를 작성하시오.

```
def sumColumn(m, columnIndex):
```

3×4 행렬 값을 읽고 각 열의 합계를 출력하는 예제 프로그램을 작성하시오. 실행 예는 다음과 같다.

3×4 행렬의 행 0 번에 대한 원소를 입력하세요: 1.5 2 3 4 

3×4 행렬의 행 1 번에 대한 원소를 입력하세요: 5.5 6 7 8 

3×4 행렬의 행 2 번에 대한 원소를 입력하세요: 9.5 1 3 1 

열 0 번 원소의 총 합은 16.5 입니다.

열 1 번 원소의 총 합은 9.0 입니다.

열 2 번 원소의 총 합은 13.0 입니다.

열 3 번 원소의 총 합은 13.0 입니다.



---

**\*14.2** (숫자 빈도수 세기) 지정되지 않은 개수만큼 정수를 읽고 이 중에 가장 많이 나온 숫자를 찾는 프로그램을 작성하시오. 예를 들어, 2 3 40 3 5 4 -3 3 3 2 0을 입력하면, 숫자 3이 가장 많이 나온 숫자이다. 한 행에 모든 숫자들을 입력하라. 가장 많이 나온 숫자의 개수가 여러 개일 경우, 이들 숫자 모두를 찾아야 한다. 예를 들어, 9 30 3 9 3 2 4를 입력했을 경우 9와 3이 두 번씩 가장 많이 나온다. 이 경우, 9와 3을 모두 찾아야만 한다.



---

**\*11.27** (열 정렬하기) 2차원 리스트의 열을 정렬하는 다음의 함수를 구현하시오. 원래의 리스트는 그대로 두고 새로운 리스트가 반환된다.

```
def sortColumns(m):
```

사용자로부터 3×3 행렬에 해당하는 숫자를 입력받고 열이 정렬된 새로운 행렬을 출력하는 테스트 프로그램을 작성하시오. 실행 예는 다음과 같다.

3×3 행렬을 한 행씩 입력하세요:

```
0.15 0.875 0.375 → Enter
0.55 0.005 0.225 → Enter
0.30 0.12 0.4 → Enter
```

열 정렬된 리스트는 다음과 같습니다.

```
0.15 0.005 0.225
0.3 0.12 0.375
0.55 0.875 0.4
```


---

**\*11.34** (기하학: 최우하단 점) 계산기하학에서 점의 집합 중 최우측 최하단의 점을 찾는 경우가 있다. 점의 집합에서 최우하단 점을 반환하는 다음의 함수를 작성하시오.

# 점의 두 값에 대한 리스트를 반환한다

**def** getRightmostLowestPoint(points):

사용자로부터 6개 점의 좌표를 입력받고 최우하단 점을 출력하는 예제 프로그램을 작성하시오. 실행 예는 다음과 같다.

6개의 점을 입력하세요: 1.5 2.5 -3 4.5 5.6 -7 6.5 -7 8 1 10 2.5 

최우측하단의 점은 (6.5, -7) 입니다.