모든 계열을 위한 기초 교육

지능형로봇프로그램밍의 이해

6 : 리스트

목차

1. 파이썬 자료형(Data Type)의 종류

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

- 기본 자료형
  - 정수형: 소수점 이하가 없는 값
  - 실수형: 소수점 이하의 정밀한 값 표현
  - 문자열: 문자를 큰 따옴표나 작은 따옴표로 감싸 나열한 것
  - 부울형: 참 또는 거짓 두 가지 상태만을 표현한 것
- 컬렉션 자료형
  - 리스트: [ ] 안에 여러 자료를 하나의 변수에 모아 놓은 것
  - 튜플: ( ) 안에 여러 자료를 하나의 변수에 모아 놓은 것
  - 딕셔너리: { } 안에 'key':'value'의 형태로 모아 놓은 것
  - 집합: set() 함수로 정의, 집합과 관련된 자료들을 쉽게 처리

1. 컬렉션 자료형 – 리스트 (1)

- 리스트 정의
  - 여러 자료를 하나의 변수에 저장하는 컬렉션 자료형이다. 10개의 숫자 모두를 하나의 리스트에 저장할 수 있고, 리스트변수를 이용하여 10개의 숫자를 액세스 하여 활용할 수 있다. 다음은 list를 나타내는 구조이다.

variable_name = [element 1, element 2... element n]	[ ]: 리스트 기호

1. 컬렉션 자료형 – 리스트 (2)

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

- 리스트 기초1

➢ list를 만들 때는 대괄호([ ])로 감싸주고 안에 들어가는 원소는 쉼표를 통해 구분한다.

– EX)

```
>>> [1, 2, 3, 4]
[1, 2, 3, 4]
>>> ['h', 'e', 'l', 'l', 'o']
['h', 'e', 'l', 'l', 'o']
>>> [10, 2.3, 'etc', True, False]
[10, 2.3, 'etc', True, False]
```

➢ Empty list : 요소를 하나도 갖지 않는 list이다.

– [ ]

1. 컬렉션 자료형 – 리스트 (2)

- 리스트 기초2
  - 이중 리스트 정의
    - 요소들은 어떠한 type도 가능하며 심지어 다른 list도 요소가 될 수 있다.
    - EX) a = [1, 2, 3, [‘one’, ‘two’, ‘three’]]
  - 삼중 리스트 정의
    - EX) b = [10, 20, [‘aa’, ‘bb’, [‘may’, ‘june’]]]

1. 컬렉션 자료형 – 리스트 (3)

한양대학교 ERICA
Education Research Industry Cluster @ Ansan

- 리스트 활용
  - 변수는 사용하기 전에 반드시 할당 되어야한다
    - EX)

```
>>> numbers = [10, 20, 30, 40]                #숫자를 가지는 list
>>> stringVar = ['hanyang', 'AI', 'data', 'Computer']          #문자열을 가지는 list
>>> empty_list_name=[]                # 빈 list
>>> print(numbers, stringVar, empty_list_name)
[10, 20, 30, 40] ['hanyang', 'AI', 'data', 'Computer'] [ ]
```

2. 리스트의 인덱스

- 리스트 인덱스
  - list는 클래스를 바탕으로 정의된 시퀀스이다. list안에 요소들은 인덱스를 통해 접근된다. 즉 list의 각 요소들은 list 변수 내에서 index를 통해 개별적으로 다루어 진다. list는 집합과 유사하지만, index를 통해 접근할 수 있다는 점에서 차이가 난다. 파이썬에서 list와 string의 공통점은 index를 가진다는 것이다. 이는 string의 문자들의 시퀀스이고, list는 요소들의 시퀀스이기 때문이다. string과 같이 list의 경우에도 index는 요소가 배열된 순서를 나타내며, 항상 '0'번째 요소부터 순서가 매겨진다. 또한 list의 index를 활용하여 index에 해당하는 값을 반환 가능하다.

2. 리스트의 인덱스

- 다음은 list의 index구조이다.

season =['spring', 'summer', 'fall', 'winter']

0	1	2	3
spring	summer	fall	winter

```
>>>season[0]
'spring'
```

- 참고로 list의 index에 음수 값을 넣을 수도 있다. list형인 season =['spring', 'summer', 'fall', winter']가 주어졌을 때, season[-1] 값을 확인해보면 'winter'가 나온다. index값이 음수로 가더라도 왼쪽에서 오른쪽으로 한 칸씩 이동하여 접근한다. 단, list의 길이를 넘어서지는 못한다.

2. 리스트의 인덱스

- 다음 조건에 맞는 코드를 작성하여 생각하여 보자. 1부터 15까지 숫자 중 홀수만 오름차순으로 정렬된 list_odd=[1,3,5,7,9,11,13,15]가 있다. 이 홀수들 중 첫번째 값과 다섯 번째 값을 출력해보면 다음과 같다.
  - EX)

```
>>> list_odd=[1,3,5,7,9,11,13,15]
>>> print(list_odd[0], list_odd[-3])
1 11
```

2. 리스트의 인덱스

- 다음 이중 리스트에서 조건에 맞는 코드를 작성하여 생각하여 보자.
  - EX)

```
>>> b = [1,2,3,['Monday', 'Tuesday', 'Wednesday']]
>>> b
[1, 2, 3, ['Monday', 'Tuesday', 'Wednesday']]
```

  - b의 첫 번째 요소 값 출력

```
>>> b[0]     # b 리스트의 첫 번째 요소 값 1
```

  - b의 맨 마지막 요소 값 출력

```
>>> b[-1] # b 리스트의 마지막 요소 값 ['Monday', 'Tuesday', 'Wednesday']
```

  - b의 맨 마지막 요소 중에서 첫 번째 요소 값을 출력

```
>>> b[-1][0]     # b[3][0]과 같음
'Monday'
```

2. 리스트의 인덱스

- **list 응용: 요소에 접근하기**
  - **list 안에 요소는 index연산자를 통해 접근될 수 있다. index도 연산이 가능하다. 단, 나눗셈 연산자는 사용이 불가하다. 또한 연산 결과는 반드시 정수(integer type)가 되어야한다.**

list_name [int_a θ int_b]	int_a θ int_b의 연산 결과가 index가 됨 list는 index에 해당하는 element값을 반환함 θ는 연산자를 의미함, θ = {+, - , *, %}

  - **numbers=[0,10,20,30,40,50,60,70], temp=3의 데이터가 있다. 다음 질문에 출력된 결과를 확인해 보자.**

numbers에서 temp값 보다 3만큼 증가된 값의 요소를 index로 접근한 경우

```
numbers[temp+3]     # 60
```

numbers에서 temp값을 2로 나누어진 나머지 index로 접근한 경우

```
number[temp%2]      # 10
```

2. 리스트의 인덱스

- **list의 membership: 'in' & 'not in'**
  - **문자열에서 사용했던 방식과 같이 왼쪽의 element가 오른쪽 list에 포함되어 있는지를 판단하는 연산자이다. 또한 이 연산자를 이용하여 list안에 어떤 원소가 있는지를 검사할 수 있다.**

in	list의 element 인가를 결정하는 연산자
not in	list의 element가 아닌 element를 결정하는 연산자

  - **다음 예를 살펴보자. 12의 악수로 이루어진 list1=[1,2,3,4,6,12]가 있다. 다음 질문에 대한 출력 결과를 확인해보자**

```
list1에서 1이 12의 약수인가?
        1 in list1              # True
list1에서 5가 12의 약수인가?
        5 in list1              # False
list1에서 7은 12의 약수가 아닌가?
        7 not in list1          # True
```

2. 리스트의 인덱스

- list와 함께 파이썬의 내장함수를 사용 할 수 있다. 그 중 len()함수는 list의 요소 개수를 int형으로 반환한다.

len(list or list_name)	list의 길이(요소의 개수)를 반환하는 함수

  - list가 다른 list를 포함하고 있어도 그것은 하나의 요소로 간주된다.

- 다음 예를 살펴보자. numbers=[0,10,20,30,40,[50,60,70]]인 list의 요소 개수가 5개인가를 검사하는 프로그램을 작성해보자
  - EX)

```
numbers=[0,10,20,30,40,[50,60,70]]
if len(numbers)<=5:
   print('Pass')
else:
   print('Fail')
```

3. 리스트 연산

- 리스트 연산
  - 여러 개의 list를 병합하기위해 연결 연산자(+)를 사용하며, list내의 요소들을 반복하기 위해 반복 연산자(*)를 사용한다. 다음 예제를 살펴보자.
  - EX)

```
>>> list1=[1,2,3,4]
>>> list2=[5,6,7,8]
>>> list3=list1+list2
>>> list3
[1, 2, 3, 4, 5, 6, 7, 8]
>>> list4=list1*3
>>> list4
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
>>>
```

  - 리스트 연산을 위한 강제 타입 변환: str(인덱스 값) 함수

EX) list5 = [70, 80, 90]

```
e[2] + "회"          # 오류
str(e[2]) + "회"     # 90회 출력
```

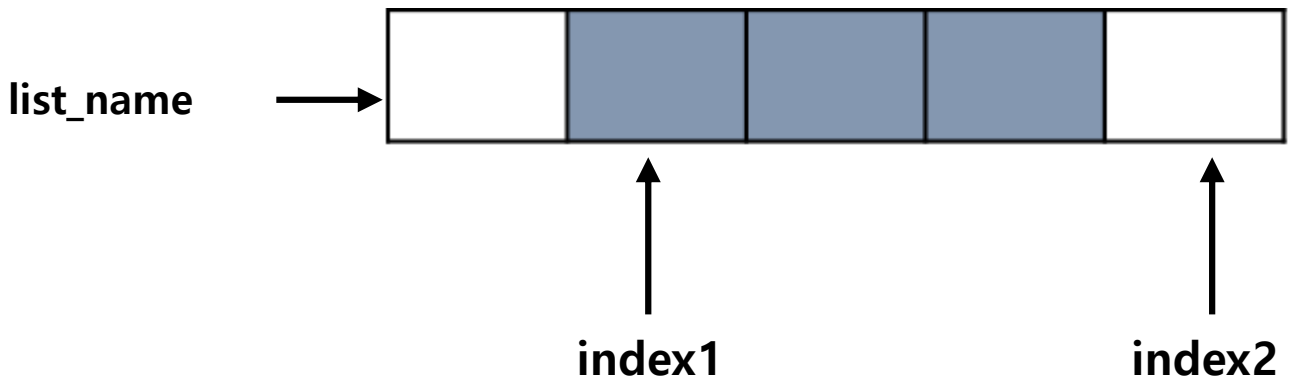
4. 리스트의 슬라이스

- 리스트 슬라이스
  - list도 slice기법을 사용하여 자를 수 있다. index연산자는 지정된 index에 해당하는 요소를 접근할 수 있도록 해준다. 반면 slice 연산자는 대괄호 안에 시작과 종료 index를 지정하여 list의 일부를 추출한다. 주의할 점은 종료 index요소는 포함되지 않는다. 다음 4가지의 index 조건으로 나누어 설명한다.

4. 리스트의 슬라이스

- 시작과 종료 index가 포함된 경우

list_name[index1 : index2]	index1이상, index2미만인 범위의 부분list를 생성하는 연산 index1값은 포함되고, index2값은 포함되지 않음



4. 리스트의 슬라이스

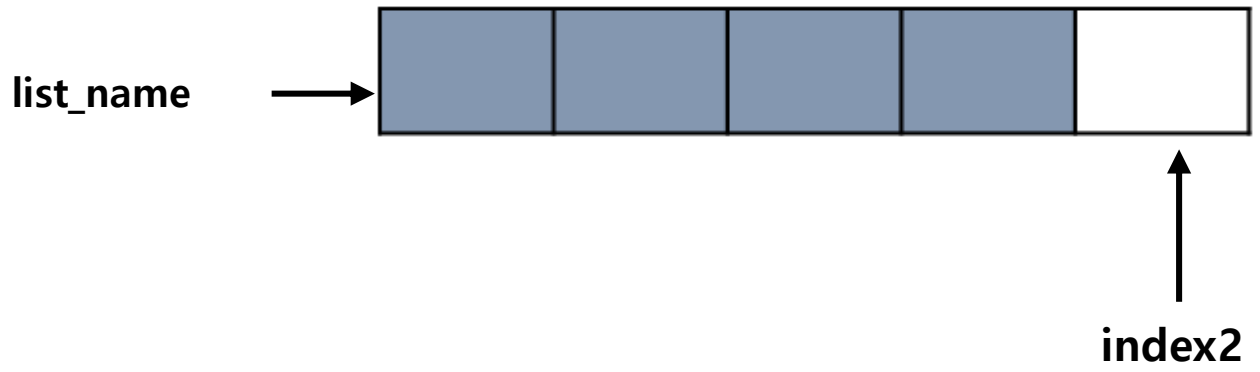
- 시작과 종료index가 포함되지 않은 경우

list_name[:]	모든 범위의 list를 생성하는 연산

4. 리스트의 슬라이스

- **종료index만 포함된 경우**

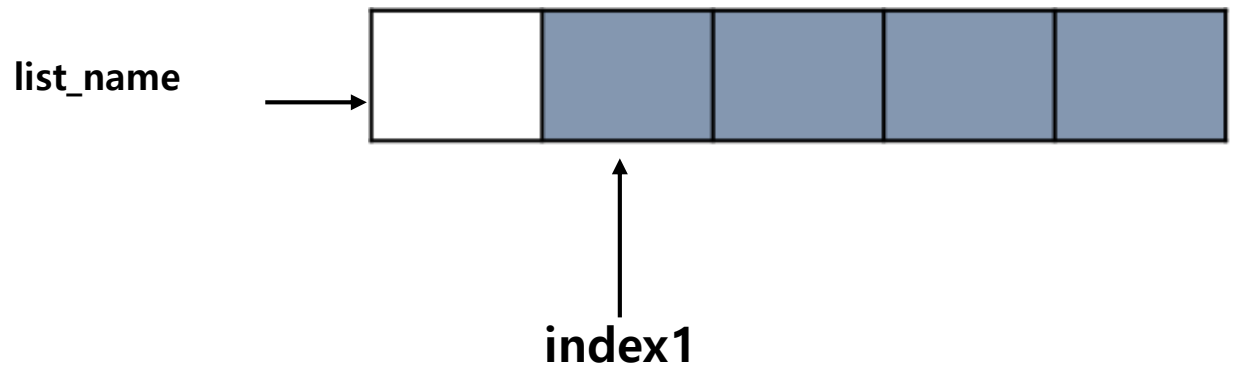
list_name[:index2]	Index2미만의 범위와 부분list를 생성하는 연산



4. 리스트의 슬라이스

- 시작index만 포함된 경우

list_name[index1:]	Index1 이상 범위의 부분list를 생성하는 연산



4. 리스트의 슬라이스

- 리스트 슬라이스 실습
  - **EX)**

```
mylist = [10, 20, 30, 40 ,50]

mylist[0:5]
mylist[1:3]
mylist[2:]
mylist[:4]
```

인덱스 번호 0~(5-1)인 4에 해당하는 [10, 20, 30, 40, 50]

인덱스 번호 1~(3-1)인 2에 해당하는 [20, 30]

인덱스 번호 2~마지막 데이터 값 [30, 40, 50]

인덱스 번호 0~(4-1)인 3에 해당하는 [10, 20, 30, 40]

4. 리스트의 슬라이스

- **다음 예를 살펴보자 시작 index 및 종료 index는 생략될 수 잇다. 이 경우 시작index는 0이고, 종료index는 마지막 index가 된다.**

word = ['h', 'a', 'n', 'd', 'i', 'c', 'r', 'a', 'f', 't']인 경우

1) word[1:5]의 결과는?
2) word[:]의 결과는?
3) word[:4]의 결과는?
4) word[5:]의 결과는?
5) word[:2]+word[9:]의 결과는?

참고로 list slice에도 음수 index를 사용할 수 있다. 번호 1번의 word[1:5]는 word[1:-5]와 동일한 결과를 나타낸다.

```
>>> word = ['h', 'a', 'n', 'd', 'i', 'c', 'r', 'a', 'f', 't']
>>> word[1:5]
['a', 'n', 'd', 'i']
>>> word[:]
['h', 'a', 'n', 'd', 'i', 'c', 'r', 'a', 'f', 't']
>>> word[:4]
['h', 'a', 'n', 'd']
>>> word[5:]
['c', 'r', 'a', 'f', 't']
>>> word[:2]+word[9:]
['h', 'a', 't']
```

5. 리스트의 요소추가

- 리스트 요소추가
  - list가 생성되면 list의 함수를 이용하여 list를 다룰 수 있다. 다음은 요소를 추가하는 함수의 기능을 나타내는 구조이다.
  - append함수

list_name.append(요소)	list뒤에 요소를 추가

  - insert함수

list_name.insert(위치, 요소)	list중간에 요소를 추가

  - extend함수

list_name.extend(list)	원래 list뒤에 새로운 list의 모든 요소 추가

5. 리스트의 요소추가

- 다음 예를 살펴보자. list_a에 차례대로 요소4, 5를 추가하며, list_a의 0번째 위치에 10이라는 요소를 추가한다. 이어서 list_a.extend() 함수를 호출하여 매개변수로 [40,50,60]인 새로운 list를 list_a에 추가한다.

```
# 리스트를 선언한다.

list_a = [1, 2, 3]

# 리스트 뒤에 요소 추가하기
print("# 리스트 뒤에 요소 추가하기")
list_a.append(4)
list_a.append(5)
print(list_a)
print()

# 리스트 중간에 요소 추가하기
print("# 리스트 중간에 요소 추가하기")
list_a.insert(0, 10)
print(list_a)
```

```
list_a.extend([40, 50, 60])
print(list_a)
```

```
* 실행결과

# 리스트 뒤에 요소 추가하기
[1, 2, 3, 4, 5]

# 리스트 중간에 요소 추가하기
[10, 1, 2, 3, 4, 5]

# 리스트에 새 리스트 요소 추가
[10, 2, 3, 4, 5, 40, 50, 60]
```

6. 리스트의 요소변경

- **list형 원소 변경**
  - **문자열과 달리 list는 할당문을 활용하여 element를 교체할 수 있다. 또한 slice와 함께 사용하여 요소를 교체하는 것도 가능하다.**

list_name[index]=new_element	**list에서 index에 해당하는 element를 new_element로 교체함**
list_name[index1 : index2] = [element1, element2, ...]	list에서 [index1:index2]의 범위에 해당하는 element들을 [element1, element2, ...]로 교체함
list_name[index1:] = [element1, element2, ...]	list에서 index1이상의 범위에 해당하는 모든 element들을 [element1, element2, ...]로 교체함
list_name[:index2] = [element1, element2, ...]	list에서 index2 미만의 범위에 해당하는 모든 element들을 [element1, element2, ...]로 교체함

6. 리스트의 요소변경

- 다음 예를 통해 list의 요소를 변경해보자. word = ['m', 'a', 'r', 'c', 'h'] 인 list에 다음 조건을 만족하도록 코드를 작성하여 출력 결과를 확인한다.
  - word을 ['m', 'u', 'l', 'c', 'h']로 변경
  - word을 ['m', 'a', 'r', 's']로 변경
  - word을 ['v', 'e', 'n', 'u', 's']로 변경

```
>>> word = ['m', 'a', 'r', 'c', 'h']
>>> word[1:3] = ['u', 'l']
>>> word            #['m', 'u', 'l', 'c', 'h']
>>> word[1:] = ['a', 'r', 's']
>>> word                          #['m', 'a', 'r', 's']
>>> word[:3] = ['v', 'e', 'n', 'u']
>>> word                          #['v', 'e', 'n', 'u', 's']
```

7. 리스트의 요소삭제

- **빈 대괄호 이용: []**
  - **리스트 요소 중에서 특정 요소를 삭제할 때 사용한다.**
  - **다음 예를 살펴보자. 정수 5개로 이루어진 list1=[1,2,3,4,5]가 있다. 다음 질문에 대한 출력 결과를 확인해보자**

list1에서 인덱스 1~2의 위치 값을 삭제하시오.

```
>>> list1 = [1,2,3,4,5]
>>> list1[1:3] = []  # 인덱스 1~2 위치의 값을 삭제
>>> list1
[1, 4, 5]
```

(주의) 대괄호를 사용해서 특정 요소를 삭제할 때는, 반드시 위 예제처럼 인덱스 범위로 지정하여 사용해야 함.

7. 리스트의 요소삭제

- **del**
  - **리스트 요소 중에서 특정 요소를 삭제할 때 사용한다. 또한 여러 개의 요소를 삭제 할 수 있다.**
  - **다음 예를 살펴보자. 정수 5개로 이루어진 list1=[1,2,3,4,5]가 있다. 다음 질문에 대한 출력 결과를 확인해보자**

list1에서 인덱스 1의 위치 값을 삭제하시오.

```
>>> list1 = [1,2,3,4,5]
>>> del list1[1]                # 인덱스 1 위치의 값을 삭제
>>> list1
[1, 3, 4, 5]
```

list1에서 인덱스 2부터 끝까지 요소를 삭제하시오.

```
>>> list1 = [1,2,3,4,5]
>>> del list1[2:]               # 인덱스 2~끝까지 값을 삭제
>>> a
[1, 2]
```

7. 리스트의 요소삭제

- 함수 이용: remove()
  - 기존 리스트에 존재하는 요소 값들 중에서 해당 값을 삭제
  - 동일한 값이 2개 이상 존재할 경우 제일 앞에 있는 값이 삭제되고 뒤에 존재하는 요소 값은 그대로 존재

```
>>> a = [10,20,30,20]
>>> a.remove(20)              # 첫번째 값 20 삭제
>>> a
[10, 30, 20]

>>> a=['aa','bb','cc']
>>> a.remove('bb')            # 문자열 'bb' 삭제
>>> a
['aa', 'cc']
```

7. 리스트의 요소정렬

- 함수 이용: sort()
  - 리스트 요소를 오름차순 순서대로 정렬
  - 다음 정수로 이루어진 예를 통해 리스트를 정렬시켜보자

```
>>> a = [15,20,3,10,40]
>>> a.sort()                    # 오름차순 정렬
>>> a
[3, 10, 15, 20, 40]
>>> a.sort(reverse=True)        # 내림차순 정렬
>>> a
[40, 20, 15, 10, 3]
```

  - 다음 문자열로 이루어진 리스트를 정렬시켜보자

```
>>> b = ['one', 'three', 'two']
>>> b.sort() # 오름차순 정렬
>>> b
['one', 'three', 'two']
```

연습문제

1. 다음 중 리스트를 생성하는 기호로 맞는 것은 무엇인가?

① [ ]　② { }　③ ( )　④ “ ”

연습문제

2. 다음의 리스트 연산자에서 문자열에서 문자열 합치기 역할을 하는 연산자는 무엇인가?

① +　② *　③ -　④ ^

연습문제

3. 다음 데이터 타입 중 실행되는 동안 주어진 값이 변화되지 않고 항상 동일하게 적용되고자 할 때 사용하는 데이터 타입은 무엇인가?

① 리스트 ② 튜플 ③ 딕셔너리 ④ 집합

연습문제

4. 다음과 같은 리스트 자료형에서 첫 번째 값부터 3번째 값 인 1, 2, 3을 슬라이싱하기 위해서 코드를 작성하시오.

<코드>

>>> a = [1,2,3,4,5,6,7]

>>> (코드 작성)

(결과) [1,2,3]

연습문제

5. 다음과 같이 학생들의 키에 대한 정보를 지닌 리스트 자료형에서 키가 170cm이상인 학생들을 출력하는 코드를 작성하시오.

<코드>

```
>>> student = [180, 170, 164, 199, 182, 172, 177]
```

연습문제

- ls = [10,5,20,7,9,31,12,11,19,32]

6 리스트 2개를 만들어서 홀수번째의 값, 짝수번째의 값을 따로 넣고 짝수번째와 홀수번째의 차를 또다른 리스트에 넣어놓고 출력하시오.
(결과 : [5, 13, -22, 1, -13])

7 ls의 값 중 인덱스 홀수 번째와 짝수 번째의 합과 차를 구하시오 (짝수번째 [0,2,4,8] – [1,3,5,7,9]홀수번째)
(결과 : -16)

8 ls에 저장된 값을 invertLs에 거꾸로 저장하시오

연습문제

6. 리스트 2개를 만들어서 홀수번째의 값, 짝수번째의 값을 따로 넣고 짝수번째와 홀수번째의 차를 또 다른 리스트에 넣어놓고 출력하시오.

ls = [10,5,20,7,9,31,12,11,19,32]

even : [10, 20, 9, 12, 19]
odd : [5, 7, 31, 11, 32]
[5, 13, -22, 1, -13]

```
evenSum=[0,0,0,0,0]
oddSum=[0,0,0,0,0]
result = [0,0,0,0,0]
k=0
for i in range(0,10,2):
    evenSum[k] = ls[i]
    k+=1
k=0
for i in range(1,10,2):
    oddSum[k] = ls[i]
    result[k]=evenSum[k]-oddSum[k]
    k+=1
print('even : ',evenSum)
print('odd : ',oddSum)
print(result)
```

감사합니다.