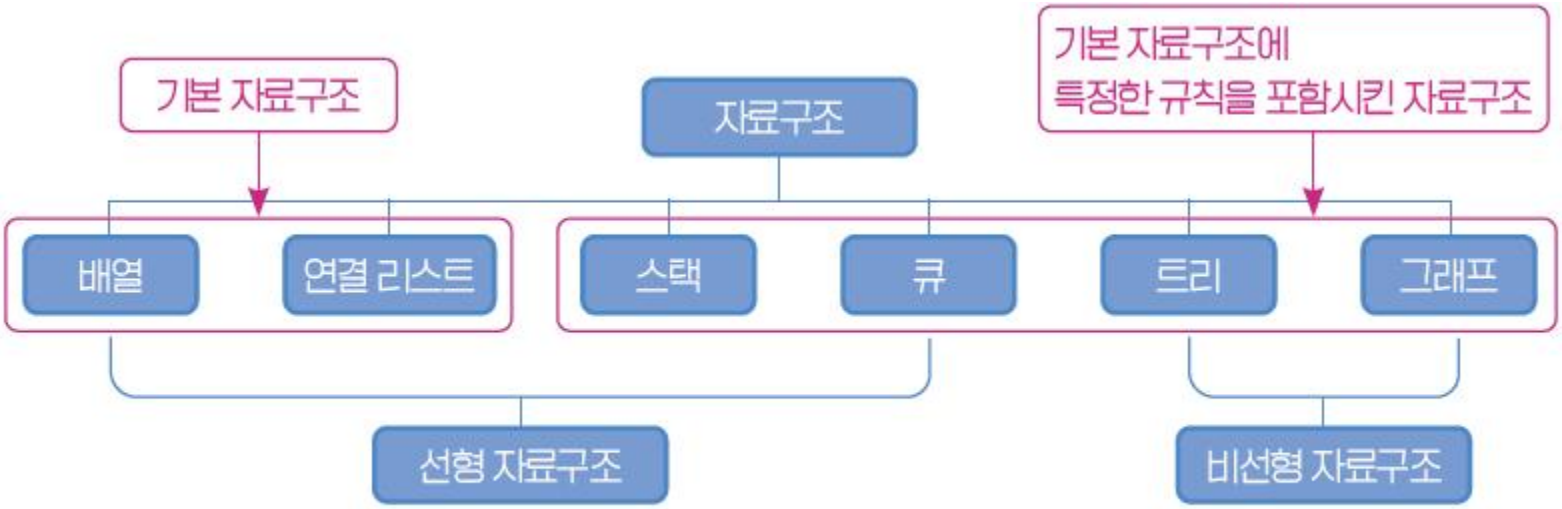
08. 자료구조 구현1

배열

연결리스트

배열

- 배열(Array)
  - 인덱스와 인덱스에 대응하는 데이터들로 이루어진 선형 자료구조
  - 순차적으로 데이터를 저장하고 저장된 데이터에 빨리 접근할 수 있는 장점
  - 파이썬 리스트는 스마트한 배열



자료구조

예제: 리스트 1

```
#리스트
list_1=[1,"가",3.14,["a","신호등","3"]]
print(list_1, type(list_1))

#리스트로 1차원 배열 표현하기 : for 문 사용
list_1=[]
for i in range(1,7):
   list_1.append(i)
print(list_1, type(list_1))

#하나의 리스트로 2차원 배열 표현하기 (1) : 직접 입력
list_1=[[1,2,3],
        [4,5,6] ]
print(list_1, type(list_1))

#하나의 리스트로 2차원 배열 표현하기 : for 문 사용 (1)
list_1=[]
for i in range(2):
   list_1.append([0]*3)
print(list_1, type(list_1))

#하나의 리스트로 2차원 배열 표현하기 : for 문 사용 (2)
list_1=[[0]*3 for i in range(2)]
print(list_1, type(list_1))
```

〈실행 결과 예〉

```
[1, '가', 3.14, ['a', '신호등', '3']] <class 'list'>

[1, 2, 3, 4, 5, 6] <class 'list'>

[[1, 2, 3], [4, 5, 6]] <class 'list'>

[[0, 0, 0], [0, 0, 0]] <class 'list'>

[[0, 0, 0], [0, 0, 0]] <class 'list'>
```

예제: 리스트2

```
#배열(2) – 2차원 배열(일반적인 파이썬을 이용한 방법)
#방법1 : for 문을 1번 사용해 2차원 배열 만들기
arr1=[[1,2],[3,4],[5,6]]
for i in arr1:
  print(i)

#방법2 : for 문을 1번 사용해 2차원 배열 만들기
arr1=[[1,2],[3,4],[5,6]]
for i,j in arr1:
  print(i,j)
#방법3 : while 문을 1번 사용해 2차원 배열 만들기
arr1=[[1,2],[3,4],[5,6]]
n=0   #n은 index, 인덱스는 0부터 시작
while n<len(arr1):
  i=arr1[n]
  print(i)
  n=n+1  #인덱스 1개씩 이동하면서 반복해야 하므로 증가

#방법4 : 중첩 반목문(2중 for 문) 사용해 2차원 배열 만들기
arr1=[[1,2],[3,4],[5,6]]
for i in arr1:
  for j in i:
    print(j,end=" ")
  print('')
```

〈실행 결과 예〉

```
[1, 2]
[3, 4]
[5, 6]

1 2
3 4
5 6

[1, 2]
[3, 4]
[5, 6]

1 2
3 4
5 6
```

```
arr1=[[1,2],[3,4],[5,6]]
  전달
for i in arr1:
  전달
  for j in i:
    print(j,end=" ") 출력  j에 순차적으로 1개씩 전달
  print('')
```

```
1 2
3 4
5 6
```

중첩 for 문을 이용해 2차원 배열 만들기

예제: 2차원 배열 값 변경하기

2차원 배열값 변경하기

```
#주어진 2차원 배열값 변경하기
arr1=[[1,2],[3,4],[5,6]]
for i in arr1:
  print(i)
print(arr1[1][1])  #2차원 배열(1,1) 위치의 값 출력
arr1[1][1]=10  #2차원 배열(1,1) 위치의 값을 10으로 변경
arr1  #변경이 적용된 내용 출력
```

〈실행 결과 예〉

```
[1, 2]
[3, 4]
[5, 6]
4
[[1, 2], [3, 10], [5, 6]]
```

	column(열) [0]	column(열) [1]
row(행) [0]	1	2
row(행) [1]	3	4
row(행) [2]	5	6

배열 구조(row, column)

row(행)과 column(열) 순서로 읽어야 함

[0][0] 1

[0][1] 2

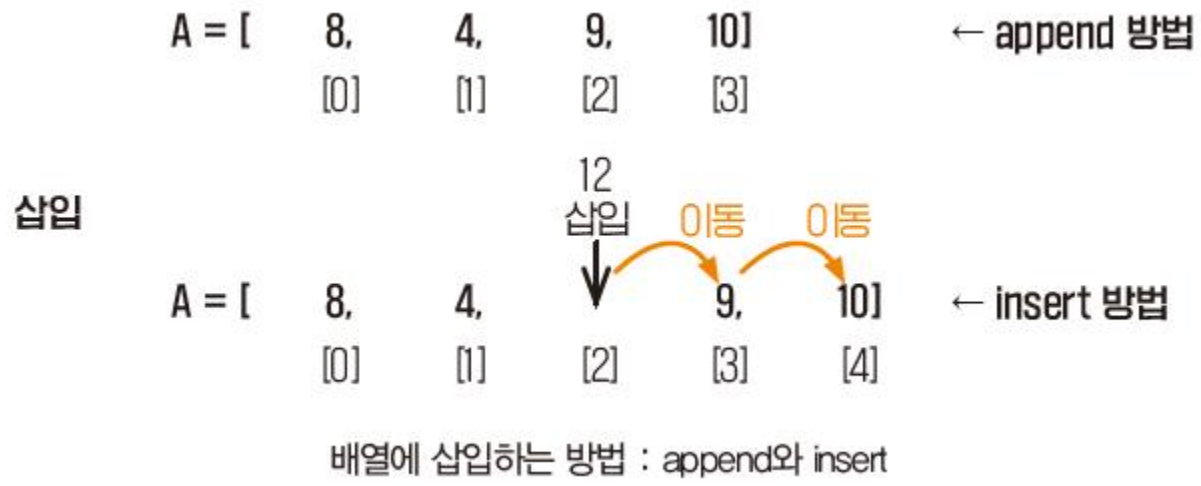
[1][0] 3

[1][1] 4

[2][0] 5

[2][1] 6

예제: 1차원 배열 값 추가하기



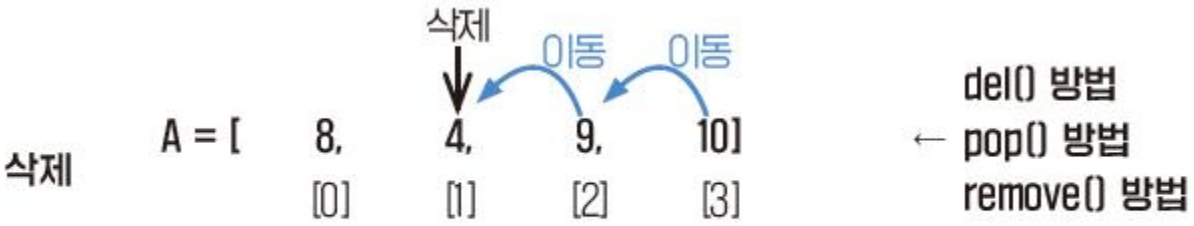
배열에 삽입하는 방법 : append와 insert

```
#숫자 8, 4, 9, 10을 arr2에 추가
arr2=[] #빈 리스트

#삽입1 : 순서대로 맨 뒤에 차례대로 입력(append)
for i in range(4):
  arr2.append(int(input(f'1차원 arr2의 인덱스[{i}]에 추가할 값은? ')))
print(arr2)

#삽입2 : 원하는 인덱스 위치에 숫자 값 추가(insert)
while True:
  i2=input("원하는 인덱스 위치 번호는 무엇입니까? ")

  if i2==" ":  #스페이스 입력되면 종료
    print("스페이스키를 입력하여 종료합니다.")
    break

  add_arr=int(input(f'arr2의 인덱스[{i2}]위치에 추가할 값은? '))

  arr2.insert(int(i2), add_arr)    #insert를 이용하여 원하는 위치에 값 입력
  print(arr2)
```

〈실행 결과 예〉

```
1차원 arr2의 인덱스[0]에 추가할 값은? 8
1차원 arr2의 인덱스[1]에 추가할 값은? 4
1차원 arr2의 인덱스[2]에 추가할 값은? 9
1차원 arr2의 인덱스[3]에 추가할 값은? 10
[8, 4, 9, 10]

원하는 인덱스 위치 번호는 무엇입니까? 2
arr2의 인덱스[2] 위치에 추가할 값은? 12
[8, 4, 12, 9, 10]
원하는 인덱스 위치 번호는 무엇입니까?
스페이스키를 입력하여 종료합니다.
```

예제: 1차원 배열 값 삭제하기



배열에서 삭제하는 방법 : pop과 remove, 그리고 del

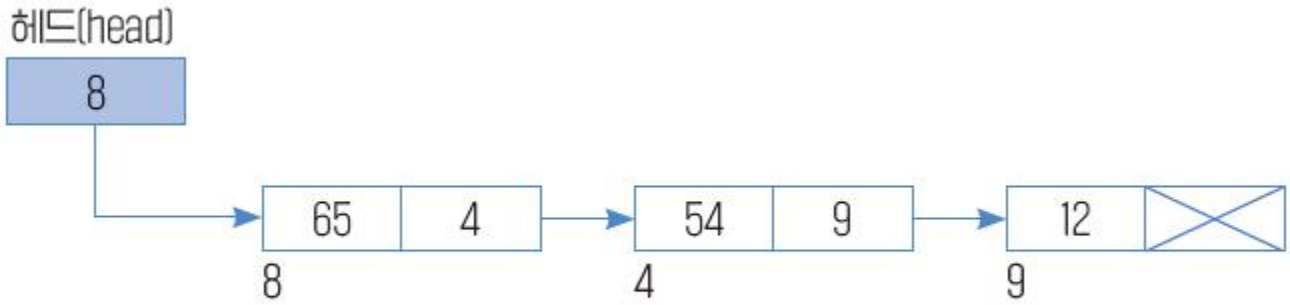
```python
arr3=[8,4,9,10]

while True:
  mode=int(input("삭제할 방법을 선택하세요(1: pop, 2:del, 3:remove): "))
  if mode==1:  #pop(인덱스 번호)로 삭제
    t1=int(input("삭제하기 원하는 인덱스 위치 번호는 무엇입니까? "))

    result=arr3.pop(t1) #pop을 이용해서 원하는 위치에 값을 입력해 삭제
    print(f'pop을 이용해 삭제한 값은 {result}입니다.')
    print(arr3)
    break
  elif mode==2: #del(1차원 배열 이름[인덱스])
    t1=int(input("삭제하기 원하는 인덱스 위치 번호는 무엇입니까? "))

    del(arr3[t1]) #del을 이용하여 원하는 위치에 값 입력해 삭제
    print(arr3)
    break

  else:
    print(arr3)
    t1=int(input("삭제하기 원하는 값은 무엇입니까? "))

    arr3.remove(t1)  #remove(삭제 원하는 값)을 이용해 삭제
    print(arr3)
    break
```

〈실행 결과 예〉

———pop을 이용해 삭제한 경우의 결과———
삭제할 방법을 선택하세요(1:pop, 2:del, 3:remove): 1
삭제하기 원하는 인덱스 위치 번호는 무엇입니까? 1
pop을 이용해 삭제한 값은 4입니다.
[8, 9, 10]
———del을 이용해 삭제한 경우의 결과———
삭제할 방법을 선택하세요(1:pop, 2:del, 3:remove): 2
삭제하기 원하는 인덱스 위치 번호는 무엇입니까? 1
[8, 9, 10]
———remove을 이용해 삭제한 경우의 결과———
삭제할 방법을 선택하세요(1:pop, 2:del, 3:remove): 3
[8, 4, 9, 10]
삭제하기 원하는 값은 무엇입니까? 4
[8, 9, 10]

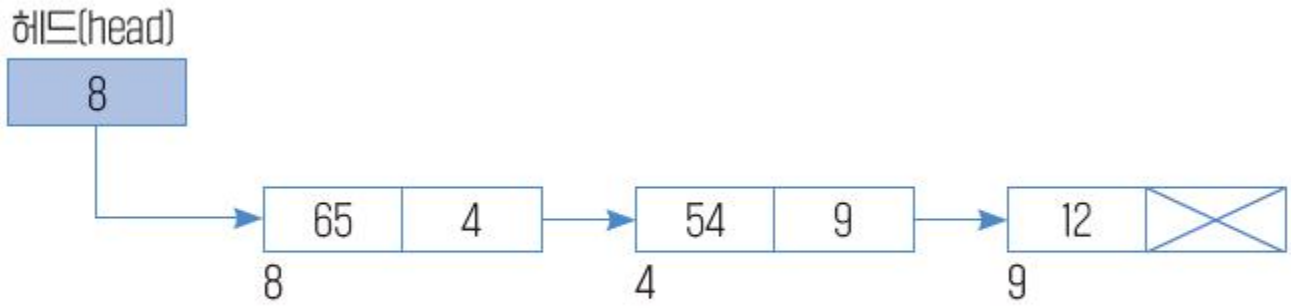
연결 리스트

- 연결 리스트(Linked List)
  - 리스트들이 한 줄로 연결된 방식의 선형 자료구조
  - 리스트의 각 원소를 노드(node)라 하고, 이 노드는 데이터와 뒤쪽 노드를 가지는 주소(포인터)를 가짐
  - 시작 노드는 헤드(head)



단순 연결 리스트 예 (1)

예제: 딕셔너리와 함수를 이용한 단순 연결 리스트

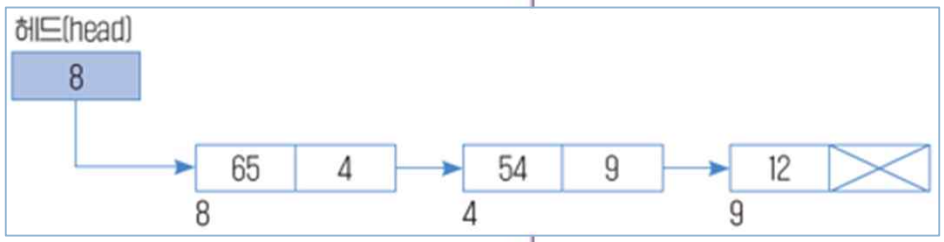


단순 연결 리스트 예 (1)

단계	설명
1	헤드 8이 가리키는 노드에 접근해 데이터를 보니 65로 이 데이터를 출력한다.
2	뒤쪽 노드를 가리키는 값 4를 따라 다음 노드에 접근해 데이터를 보니 54로 이 데이터를 출력한다.
3	뒤쪽 노드를 가리키는 값 9를 따라 다음 노드에 접근해 데이터를 보니 12로 이 데이터를 출력한다.
4	뒤쪽 노드를 가리키는 값을 확인해 보니 더는 없으므로 이 과정을 종료한다.

예제 1

```
A={8:[65,4],4:[54,9],9:[12,None]}
print(A)

#현재 데이터가 다음 데이터 주소와 함께 정의
A[8][0]=65
A[8][1]=4 #다음 데이터 키
A[4][0]=54
A[4][1]=9 #다음 데이터 키
A[9][0]=12
A[9][1]=None #다음 데이터 키

def SinglyLinkedList(A,root):
  head=root
  node_next=head

  while node_next!=None:
    node_value=A[node_next][0]
    node_next=A[node_next][1]
    print('value:',node_value)

SinglyLinkedList(A,8)
```



〈실행 결과 예〉

{8: [65, 4], 4: [54, 9], 9: [12, None]}

value: 65

value: 54

value: 12

예제2

```
#딕셔너리를 이용한 방법
a={"h":[8,"t"],"t":[4,"u"],"u":[9, None]}
print(a)

#현재 데이터가 다음 데이터 주소를 함께 정의
a["h"][0]=8
a["h"][1]="t" #다음 데이터 키
a["t"][0]=4
a["t"][1]="u" #다음 데이터 키
a["u"][0]=9
a["u"][1]=None #다음 데이터 키

#노드 순회
def test1(a,root_key):
    head=root_key
    node_next=head
    while True:
        node_value  = a[node_next][0]
        node_next = a[node_next][1]
        print('value:',node_value)

        if node_next == None:
            print('final node ')
            break
test1(a,"h")
```

〈실행 결과 예〉

{'h': [8, 't'], 't': [4, 'u'], 'u': [9, None]}

value: 8

value: 4

value: 9

final node

끝