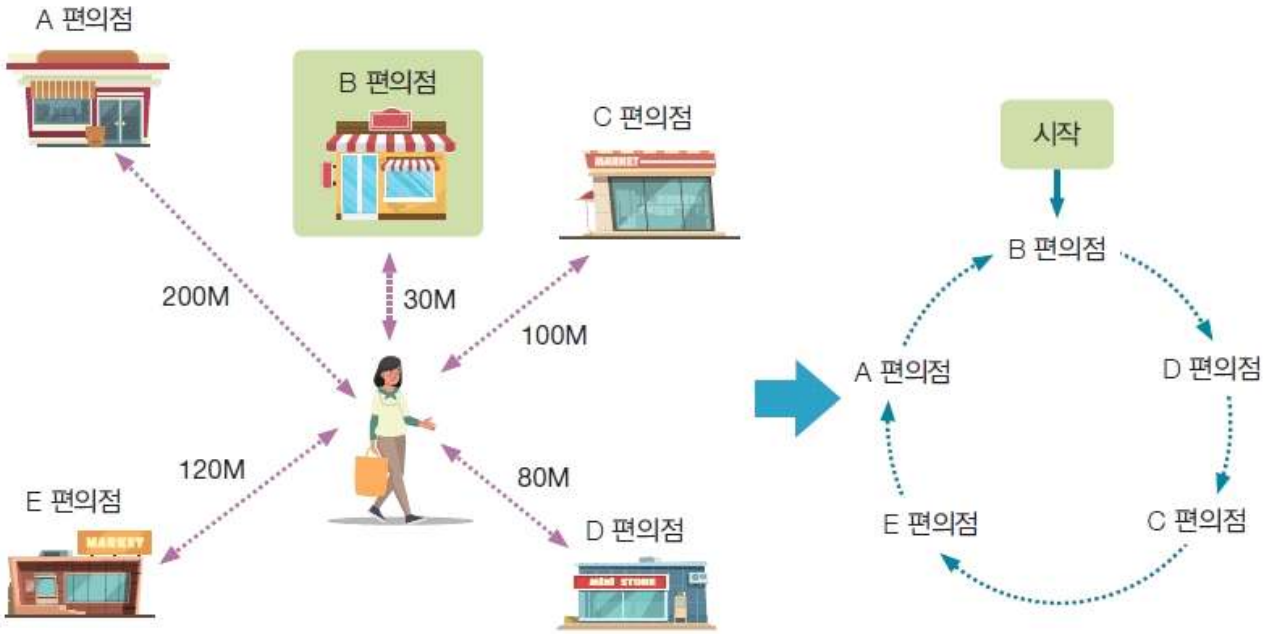
응용예제 01 현재 위치부터 가까운 편의점 관리하기

난이도★★★☆☆

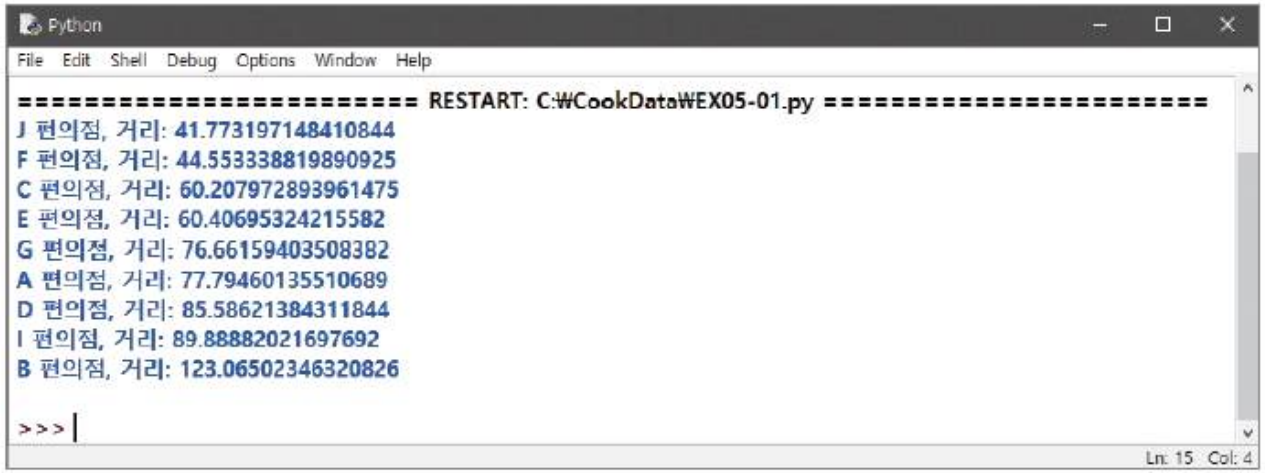
예제 설명 현재 위치를 (0, 0)이라 가정하고, 편의점 위치(x, y)와 거리가 가까운 순서대로 원형 연결 리스트를 생성하는 프로그램을 다음 조건에 맞게 작성한다.

- 편의점 10개를 A, B, C, … 순서로 이름을 부여한다.
- 편의점 위치 x와 y는 1부터 100까지 랜덤하게 좌표가 생성되도록 한다.
- 현재 위치와 편의점 거리는 ($x^2 + y^2$)의 제곱근(sqrt)으로 계산한다.
- 편의점 데이터 1개는 (편의점이름, x좌표, y좌표) 형식의 튜플로 구성한다.



응용예제 01 현재 위치부터 가까운 편의점 관리하기

실행 결과

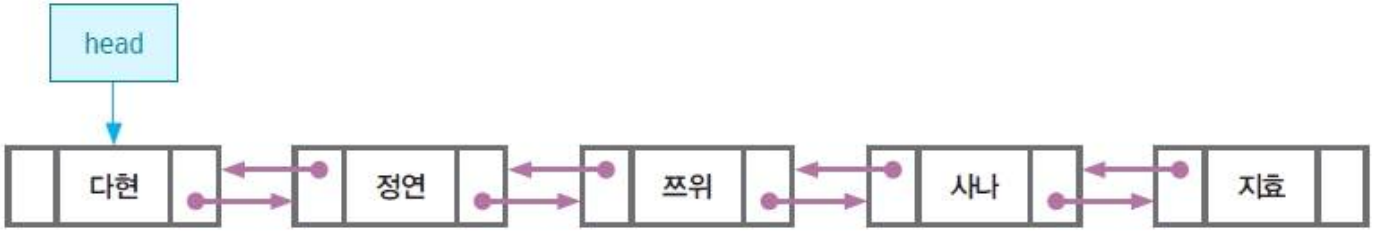


```
import random
import math

## 클래스와 함수 선언 부분 ##
class Node() :
        def __init__ (self) :
                self.data = None
                self.link = None

def printStores(start) :
        current = start
        if current == None :
                return

        while current.link != head:
                current = current.link
                x, y = current.data[1:]
                print(current.data[0], '편의점, 거리:', math.sqrt(x*x + y*y))
        print()

def  makeStoreList(store) :
        global memory, head, current, pre

        node = Node()
        node.data = store
        memory.append(node)

        if head == None : # 첫 번째 편의점
                head = node
                node.link = head
                return

        # 새 편의점이 첫 번째 편의점보다 가까우면 첫 편의점으로 만듦
        nodeX, nodeY = node.data[1:]
        nodeDist = math.sqrt(nodeX*nodeX + nodeY*nodeY)
        headX, headY = head.data[1:]
        headDist = math.sqrt(headX*headX + headY*headY)
        if headDist > nodeDist :                # 헤드 앞에 삽입
                node.link = head
                last = head
                while last.link != head :
                        last = last.link
                last.link = node
                head = node
                return

        current = head                          # 중간에 데이터를 넣을 경우
        while current.link != head :
                pre = current
                current = current.link
                currX, currY = current.data[1:]
                currDist = math.sqrt(currX*currX + currY*currY)
                if currDist > nodeDist :
                        pre.link = node
                        node.link = current
                        return

        current.link = node
        node.link = head


## 전역 변수 선언 부분 ##
memory = []
head, current, pre = None, None, None

## 메인 코드 부분 ##
if __name__ == "__main__" :

        storeArray = []
        storeName = 'A'
        for _ in range(10) :
                store = (storeName, random.randint(1, 100),
random.randint(1, 100) )
                storeArray.append(store)
                storeName = chr(ord(storeName) + 1)   # 편의점 이름을
A->B->C… 으로 변경

        for store in storeArray :
                makeStoreList(store)

        printStores(head)
```

응용예제 02 이중 연결 리스트 구현하기

난이도★ ★☆☆☆

예제 설명 다음과 같이 양방향으로 링크가 연결되는 이중 연결 리스트를 만든다. 헤드부터 차례대로 출력한 후 이어서 마지막 노드부터 거꾸로 다시 출력해 보자.



실행 결과

```
## 클래스와 함수 선언 부분 ##
class Node2() :
        def __init__ (self) :
                self.plink = None               # 앞쪽 링크
                self.data = None
                self.nlink = None               # 뒤쪽 링크

def printNodes(start):
        current = start
        if current.nlink == None :
                return
        print("정방향 --> ", end=' ')
        print(current.data, end=' ')
        while current.nlink != None:
                current = current.nlink
                print(current.data, end=' ')
        print()
        print("역방향 --> ", end=' ')
        print(current.data, end=' ')
        while current.plink != None:
                current = current.plink
                print(current.data, end=' ')

## 전역 변수 선언 부분 ##
memory = []
head, current, pre = None, None, None
dataArray = ["다현", "정연", "쯔위", "사나", "지효"]

## 메인 코드 부분 ##
if __name__ == "__main__" :

        node = Node2()                          # 첫 번째 노드
        node.data = dataArray[0]
        head = node
        memory.append(node)

        for data in dataArray[1:] :             # 두 번째 이후 노드
                pre = node
                node = Node2()
                node.data = data
                pre.nlink = node
                node.plink = pre
                memory.append(node)

        printNodes(head)
```

CHAPTER 06

스택

학습목표

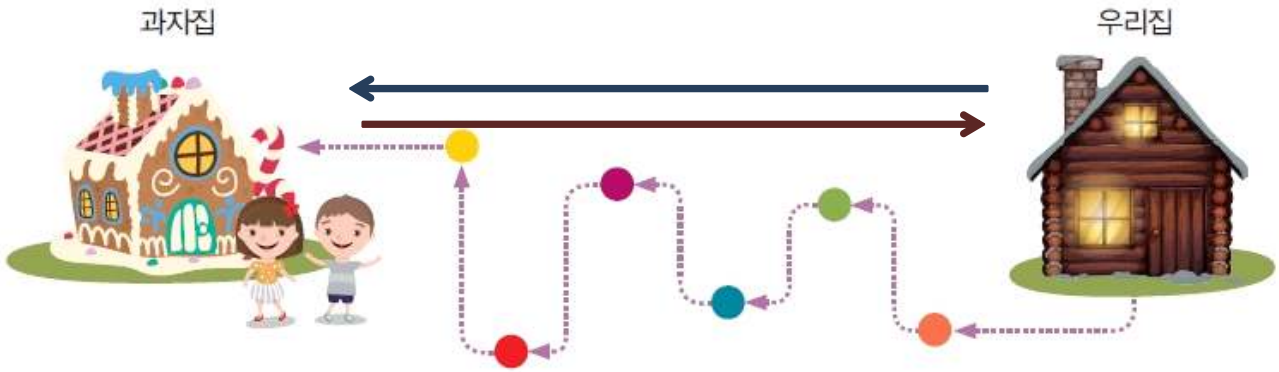
- 스택의 개념을 파악한다.
- 스택에 데이터를 넣거나 추출하는 원리를 이해한다.
- 파이썬으로 스택을 조작하는 코드를 작성한다.
- 스택으로 활용되는 다양한 응용 프로그램을 작성한다.

응용예제 헨젤과 그레텔의 집으로 돌아가기

난이도 ★☆☆☆☆

예제 설명

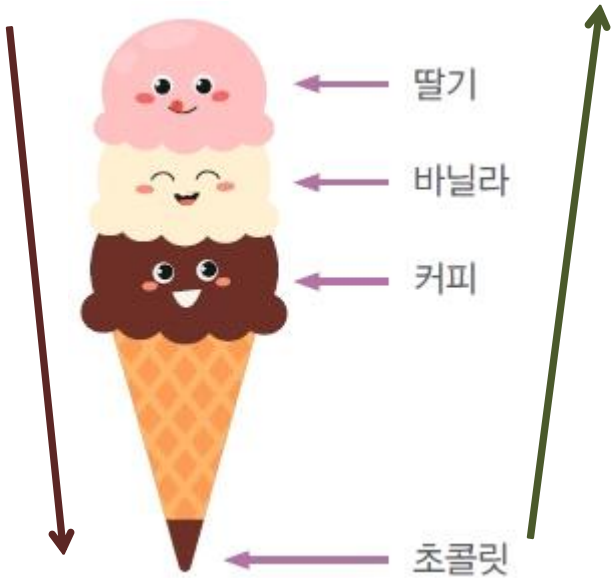
헨젤과 그레텔이 돌을 떨어뜨리면서 숲으로 들어간다. 과자집에서 집으로 돌아갈 때는 떨어뜨린 순서와 반대로 돌을 주워야 한다. 스택을 이용해서 집으로 무사히 돌아가도록 하자.



실행 결과

스택 구조란?

- 아이스크림 콘에 여러 가지 맛을 쌓을 때, 가장 먼저 넣은 맛을 가장 나중에 먹을 수 있는 구조



초콜릿을 먹으려면 순서대로 쌓여 있는
딸기 → 바닐라 → 커피를 차례대로 먹어야
초콜릿을 먹을 수 있다.

Section 01 스택의 기본

스택의 개념

- 스택(Stack) 자료구조는 한쪽 끝이 막힌 형태(ex : 한쪽 끝이 막힌 주차장, 종이컵 수거함 등)
- 입구가 하나이기 때문에 먼저 들어간 것이 가장 나중에 나오는 구조(선입후출, 후입선출)

그림 6-1 스택의 실생활 사례

스택의 개념

- 종이컵 수거함에 종이컵을 넣고 빼는 예

그림 6-2 종이컵 수거함의 작동 원리

스택 원리

스택 기본 구조

- 스택에 데이터를 삽입하는 작동 : push
- 스택에 데이터를 추출하는 작동 : pop
- 스택에 들어 있는 가장 위의 데이터 : top

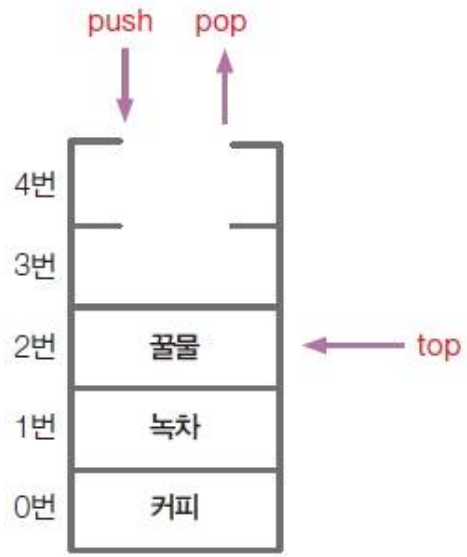


그림 6-3 스택 기본 구조

Section 01 스택의 기본

- 데이터 삽입 : push

그림 6-4 스택에 데이터를 삽입(push)하는 과정

- 데이터 추출 : pop

그림 6-5 스택에서 데이터를 추출(pop)하는 과정

Section 02 스택의 간단 구현

스택 생성

- 배열 크기를 지정한 후 빈 스택 생성

```
stack = [None, None, None, None, None]
top = -1
```

그림 6-6 크기가 5칸인 스택의 초기 상태

Section 02 스택의 간단 구현

데이터 삽입 : push

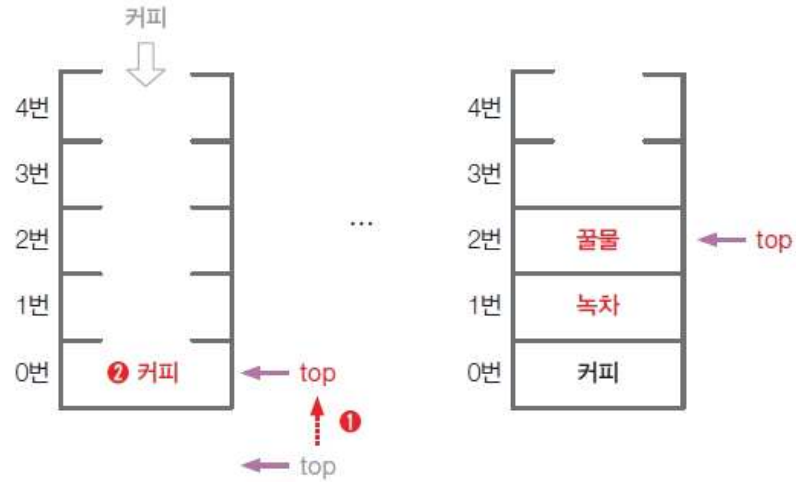


그림 6-7 스택의 데이터 삽입 과정

Code06-01.py 크기가 5칸인 스택의 생성과 데이터 3개 입력

```
1  stack = [None, None, None, None, None]
2  top = -1
3
4  top += 1 ❶
5  stack[top] = "커피" ❷
6  top += 1
7  stack[top] = "녹차"
8  top += 1
9  stack[top] = "꿀물"
10
11 print("----- 스택 상태 -----")
12 for i in range(len(stack)-1, -1, -1) :
13     print(stack[i])
```

실행 결과

```
None
None
꿀물
녹차
커피
```

Section 02 스택의 간단 구현

데이터 추출 : pop

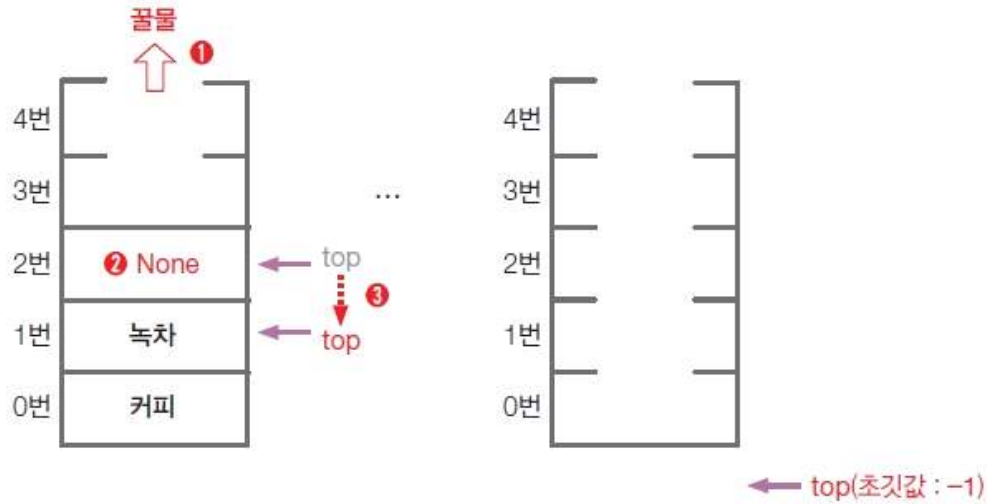


그림 6-8 스택의 데이터 추출

Code06-02.py 스택에서 데이터 3개 추출

```
1  stack = ["커피", "녹차", "꿀물", None, None]
2  top = 2
3
4  print("----- 스택 상태 -----")
5  for i in range(len(stack)-1, -1, -1) :
6      print(stack[i])
7
8  print("--------------------")
```

Section 02 스택의 간단 구현

```
9  data = stack[top] ❶
10 stack[top] = None ❷
11 top -= 1 ❸
12 print("pop -->", data)
13
14 data = stack[top]
15 stack[top] = None
16 top -= 1
17 print("pop -->", data)
18
19 data = stack[top]
20 stack[top] = None
21 top -= 1
22 print("pop -->", data)
23 print("------------------")
24
25 print("----- 스택 상태 -----")
26 for i in range(len(stack)-1, -1, -1) :
27     print(stack[i])
```

실행 결과

```
----- 스택 상태 -----
None
None
꿀물
녹차
커피
------------------
pop --> 꿀물
pop --> 녹차
pop --> 커피
------------------
----- 스택 상태 -----
None
None
None
None
None
```

Section 03 스택의 일반 구현

스택 초기화

- 5개짜리 빈 스택을 생성하는 코드

```
stack = [None, None, None, None, None]
```

- SIZE 값만 변경하면 원하는 크기의 빈 스택 생성(초기화)

```
SIZE = 5    # 스택 크기
stack = [None for _ in range(SIZE)]
top = -1
```

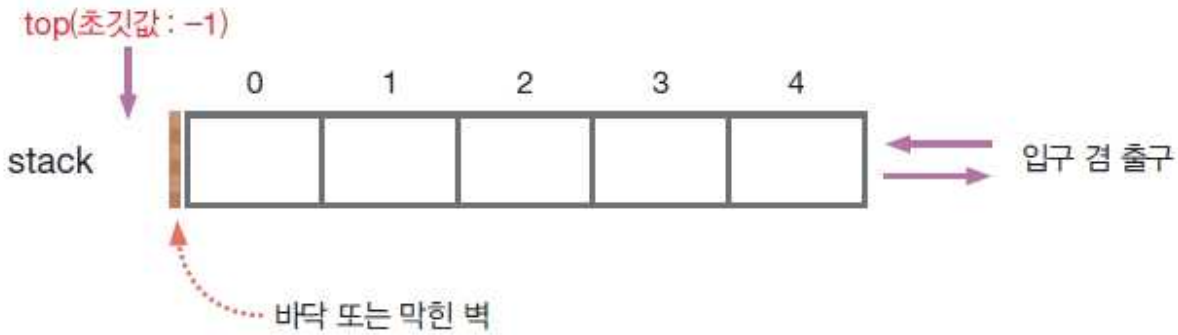


그림 6-9 초기화된 스택

Section 03 스택의 일반 구현

데이터 삽입 과정

1) 스택이 꽉 찼는지 확인하는 함수

- **top 값이 '스택 크기-1'과 같다**면 스택이 꽉 찬 상태

그림 6-10 스택이 꽉 찬 상태

```
if (top값 == 스택크기-1) :
    스택이 꽉 찼음
```

Code06-03.py 스택이 꽉 찼는지 확인하는 함수

```
1  def isStackFull() :
2      global SIZE, stack, top
3      if (top >= SIZE-1) :
4          return True
5      else :
6          return False
7
8  SIZE = 5
9  stack = ["커피", "녹차", "꿀물", "콜라", "환타"]
10 top = 4
11
12 print("스택이 꽉 찼는지 여부 ==>", isStackFull())
```

실행 결과

```
스택이 꽉 찼는지 여부 ==> True
```

Section 03 스택의 일반 구현

2) 스택에 데이터를 삽입하는 함수

Code06-04.py 스택에 데이터를 삽입하는 함수

```
1  def isStackFull() :
…  # 생략(Code06-03.py의 2~6행과 동일)
7
8  def push(data) :
9      global SIZE, stack, top
10     if (isStackFull()) :
11         print("스택이 꽉 찼습니다.")
12         return
13     top += 1
14     stack[top] = data
15
16 SIZE = 5
17 stack = ["커피", "녹차", "꿀물", "콜라", None]
18 top = 3
19
20 print(stack)
21 push("환타")
22 print(stack)
23 push("게토레이")
```

실행 결과

```
['커피', '녹차', '꿀물', '콜라', None]
['커피', '녹차', '꿀물', '콜라', '환타']
스택이 꽉 찼습니다.
```

Section 03 스택의 일반 구현

SELF STUDY 6-1

Code06-04.py의 isStackFull() 함수를 없애고, 대신에 push() 함수만으로 그 기능을 모두 구현하자. 실행 결과는 Code06-04.py와 동일하다.

실행 결과

```
['커피', '녹차', '꿀물', '콜라', None]
['커피', '녹차', '꿀물', '콜라', '환타']
스택이 꽉 찼습니다.
```

Section 03 스택의 일반 구현

데이터 추출 과정

1) 스택이 비었는지 확인하는 함수

top 값이 -1이라면 스택은 비어 있는 상태

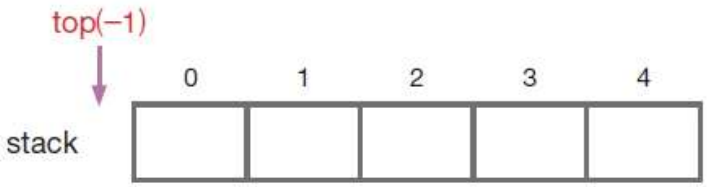


그림 6-11 스택이 비어 있는 상태

```
if (top값 == -1) :
    스택이 비었음
```

Code06-05.py 스택이 비었는지 확인하는 함수

```
1  def isStackEmpty() :
2      global SIZE, stack, top
3      if (top == -1) :
4          return True
5      else :
6          return False
7
8  SIZE = 5
9  stack = [None for _ in range(SIZE)]
10 top = -1
11
12 print("스택이 비었는지 여부 ==>", isStackEmpty())
```

실행 결과

```
스택이 비었는지 여부 ==> True
```

Section 03 스택의 일반 구현

2) 스택에서 데이터를 추출하는 함수

Code06-06.py 스택에서 데이터를 추출하는 함수

```
1  def isStackEmpty() :
…  # 생략(Code06-05.py의 2~6행과 동일)
7
8  def pop() :
9      global SIZE, stack, top
10     if (isStackEmpty()) :
11         print("스택이 비었습니다.")
12         return None
13     data = stack[top]
14     stack[top] = None
15     top -= 1
16     return data
17
18 SIZE = 5
19 stack = ["커피", None, None, None, None]
20 top = 0
21
22 print(stack)
23 retData = pop()
24 print("추출한 데이터 -->", retData)
25 print(stack)
26 retData = pop()
```

실행 결과

```
['커피', None, None, None, None]
추출한 데이터 --> 커피
[None, None, None, None, None]
스택이 비었습니다.
```

Section 03 스택의 일반 구현

SELF STUDY 6-2

Code06-06.py의 isStackEmpty() 함수를 없애고, 대신에 pop() 함수만으로 그 기능을 모두 구현하자. 실행 결과는 Code06-06.py와 동일하다.

실행 결과

```
['커피', None, None, None, None]
추출한 데이터 --> 커피
[None, None, None, None, None]
스택이 비었습니다.
```

- **데이터 확인 : peek**
  - top 위치의 데이터를 확인만 하고 스택에 그대로 두는 것 : peek

그림 6-12 데이터를 확인하는 peek 작동

Section 03 스택의 일반 구현

Code06-07.py 스택에서 top 위치의 데이터를 확인하는 함수

```
1  def isStackEmpty() :
2      global SIZE, stack, top
3      if (top == -1) :
4          return True
5      else :
6          return False
7
8  def peek() :
9      global SIZE, stack, top
10     if (isStackEmpty()) :
11         print("스택이 비었습니다.")
12         return None
13     return stack[top]
14
15 SIZE = 5
16 stack = ["커피", "녹차", "꿀물", None, None]
17 top = 2
18
19 print(stack)
20 retData = peek()
21 print("top의 데이터 확인 -->", retData)
22 print(stack)
```

실행 결과

```
['커피', '녹차', '꿀물', None, None]
top의 데이터 확인 --> 꿀물
['커피', '녹차', '꿀물', None, None]
```

스택 완성

Code06-08.py 스택 작동을 위한 통합 코드

```
## 함수 선언 부분 ##
def isStackFull() :                    # 스택이 꽉 찼는지 확인하는 함수
    global SIZE, stack, top
    if (top >= SIZE-1) :
        return True
    else :
        return False

def isStackEmpty() :                   # 스택이 비었는지 확인하는 함수
    global SIZE, stack, top
    if (top == -1) :
        return True
    else :
        return False

def push(data) :                       # 스택에 데이터를 삽입하는 함수
    global SIZE, stack, top
    if (isStackFull()) :
        print("스택이 꽉 찼습니다.")
        return
    top += 1
    stack[top] = data

```

Section 03 스택의 일반 구현

```
24 def pop() :                                  # 스택에서 데이터를 추출하는 함수
25     global SIZE, stack, top
26     if (isStackEmpty()) :
27         print("스택이 비었습니다.")
28         return None
29     data = stack[top]
30     stack[top] = None
31     top -= 1
32     return data
33
34 def peek() :                                 # 스택에서 top 위치의 데이터를 확인하는 함수
35     global SIZE, stack, top
36     if (isStackEmpty()) :
37         print("스택이 비었습니다.")
38         return None
39     return stack[top]
40
41 ## 전역 변수 선언 부분 ##
42 SIZE = int(input("스택 크기를 입력하세요 ==> "))
43 stack = [None for _ in range(SIZE)]
44 top = -1
45
```

Section 03 스택의 일반 구현

```
46 ## 메인 코드 부분 ##
47 if __name__ == "__main__" :
48     select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")
49
50     while (select != 'X' and select != 'x') :
51         if select == 'I' or select == 'i' :
52             data = input("입력할 데이터 ==> ")
53             push(data)
54             print("스택 상태 : ", stack)
55         elif select == 'E' or select == 'e' :
56             data = pop()
57             print("추출된 데이터 ==> ", data)
58             print("스택 상태 : ", stack)
59         elif select == 'V' or select == 'v' :
60             data = peek()
61             print("확인된 데이터 ==> ", data)
62             print("스택 상태 : ", stack)
63         else :
64             print("입력이 잘못됨")
65
66         select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")
67
68     print("프로그램 종료!")
```

Section 03 스택의 일반 구현

실행 결과

```
스택 크기를 입력하세요 ==> 5
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 커피
스택 상태 : ['커피', None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 녹차
스택 상태 : ['커피', '녹차', None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 꿀물
스택 상태 : ['커피', '녹차', '꿀물', None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> V
확인된 데이터 ==> 꿀물
스택 상태 : ['커피', '녹차', '꿀물', None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 꿀물
스택 상태 : ['커피', '녹차', None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 녹차
스택 상태 : ['커피', None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 커피
스택 상태 : [None, None, None, None, None]
```

Section 03 스택의 일반 구현

```
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
스택이 비었습니다.
추출된 데이터 ==> None
스택 상태 : [None, None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> X
프로그램 종료!
```

웹 서핑에서 이전 페이지 돌아가기

- 코드로 http://www.hanbit.co.kr에 접속(webbrowser 패키지 사용) 예

```
import webbrowser
webbrowser.open("http://www.hanbit.co.kr")
```

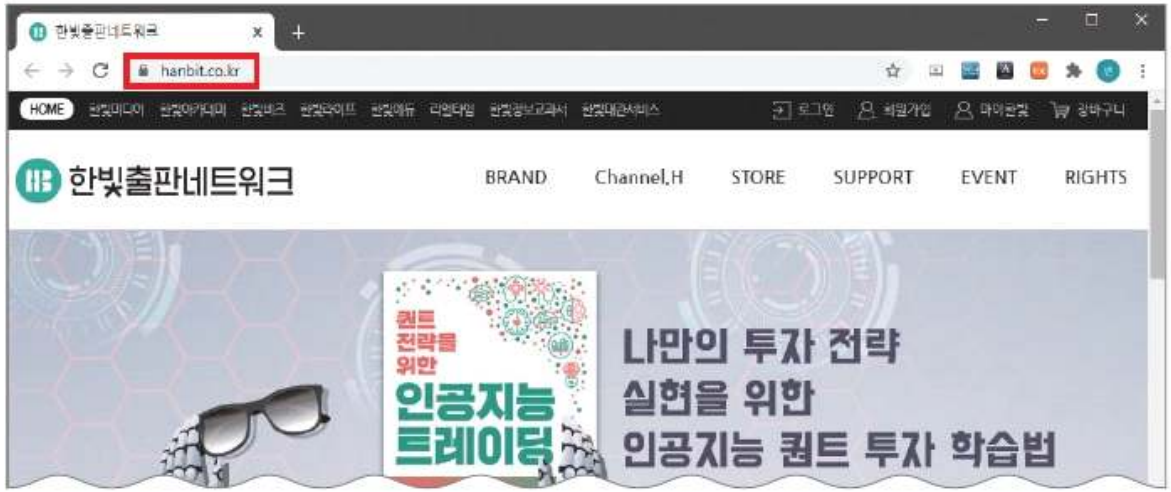


그림 6-13 webbrowser.open() 함수로 실행한 웹 브라우저

Section 04 스택의 응용

- 방문한 URL을 스택에 push하는 과정

그림 6-14 스택에서 추가로 URL 데이터를 push하는 과정

Code06-09.py 스택을 활용한 웹 브라우저의 [이전 페이지] 버튼 기능 구현

```
1  import webbrowser
2  import time
3
4  ## 함수 선언 부분 ##
…  # 생략(Code06-08.py의 2~39행과 동일)
43
44 ## 전역 변수 선언 부분 ##
45 SIZE = 100
46 stack = [None for _ in range(SIZE)]
47 top = -1
48
```

Section 04 스택의 응용

```
49 ## 메인 코드 부분 ##
50 if __name__ == "__main__" :
51     urls = ['naver.com', 'daum.net', 'nate.com']
52
53     for url in urls :
54         push(url)
55         webbrowser.open('http://'+url)
56         print(url, end = '-->')
57         time.sleep(1)
58
59     print("방문 종료")
60     time.sleep(5)
61
62     while True :
63         url = pop()
64         if url == None :
65             break
66         webbrowser.open('http://'+url)
67         print(url, end = '-->')
68         time.sleep(1)
69     print("방문 종료")
```

실행 결과

```
naver.com-->daum.net-->nate.com-->방문 종료
nate.com-->daum.net-->naver.com-->방문 종료
```

Section 04 스택의 응용

그림 6-15 실행 결과

Section 04 스택의 응용

괄호의 매칭 검사

- 올바른 괄호
  - (A+B) : 여는 괄호와 닫는 괄호의 쌍이 맞다.
  -)A+B(: 괄호 개수는 맞지만 여는 괄호로 시작하지 않아서 틀렸다.
  - ((A+B)-C : 여는 괄호와 닫는 괄호의 개수가 달라서 틀렸다.
  - (A+B] : 여는 괄호와 닫는 괄호의 종류가 달라서 틀렸다.
  - (〈A+{B-C} / [C*D]〉) : 다양한 괄호가 나오지만 쌍이 맞다.

- 스택을 활용한 괄호 검사

❶ 닫는 괄호를 만났을 때 스택은 비어 있지 않아야 한다.
❷ 닫는 괄호를 만났을 때 추출한 괄호는 여는 괄호여야 한다.
❸ ❷를 만족해도 두 괄호의 종류(소괄호, 중괄호, 대괄호)가 같아야 한다.
❹ 모든 수식의 처리가 끝나면 스택은 비어 있어야 한다.

Section 04 스택의 응용

스택에서 괄호 검사 처리 과정 예

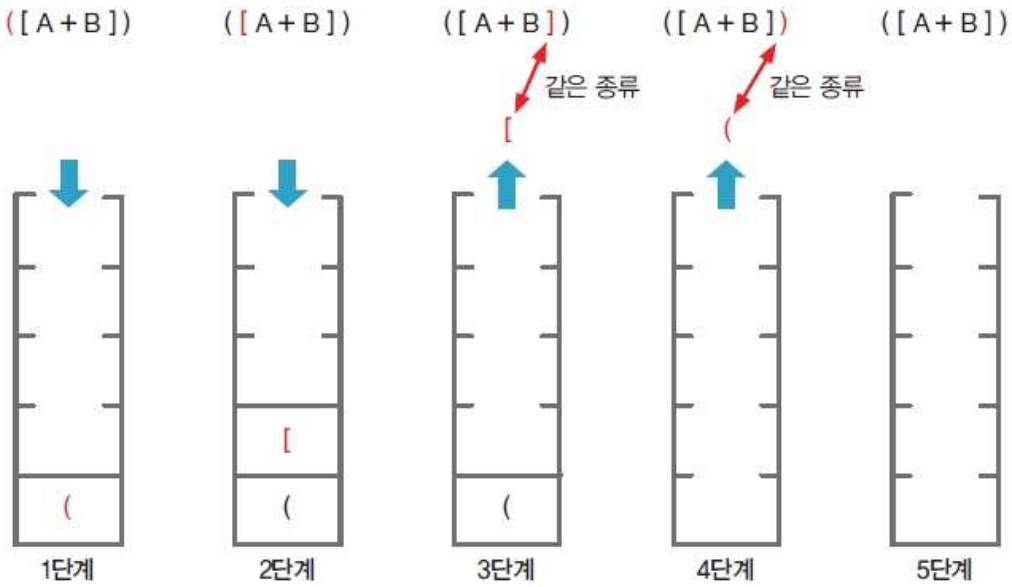


그림 6-16 스택에서 괄호 검사를 처리하는 과정 예 : ([A + B])

Section 04 스택의 응용

- 여는 괄호는 push하고, 닫는 괄호를 만나면 pop 해야 한다.
- 수식 중 글자 하나가 추출, 괄호라면 다음 과정을 처리한다.

```
if '(', '[', '{', '<' 중 하나면
    push()
elif ')', ']', '}', '>' 중 하나면
    열린 괄호 = pop()
    if 두 괄호의 쌍이 맞으면
        통과
    else
        오류
```

- 모든 수식의 글자를 처리한 후 스택이 비었는지 확인

```
if 스택이 비었으면
    정상
else
    오류
```

Section 04 스택의 응용

Code06-10.py 스택을 활용한 괄호 매칭 검사 구현

```
## 함수 선언 부분 ##
... # 생략(Code06-08.py의 2~39행과 동일)

def checkBracket(expr) :
    for ch in expr :
        if ch in '([{<' :
            push(ch)
        elif ch in ')]}>' :
            out = pop()
            if ch == ')' and out == '(' :
                pass
            elif ch == ']' and out == '[' :
                pass
            elif ch == '}' and out == '{' :
                pass
            elif ch == '>' and out == '<' :
                pass
            else :
                return False
        else :
            pass
    if isStackEmpty() :
        return True
    else :
        return False

```

Section 04 스택의 응용

```
64 ## 전역 변수 선언 부분 ##
65 SIZE = 100
66 stack = [None for _ in range(SIZE)]
67 top = -1
68
69 ## 메인 코드 부분 ##
70 if __name__ == "__main__" :
71
72     exprAry = ['(A+B)', ')A+B(', '((A+B)-C', '(A+B]', '(<A+{B-C}/[C*D]>)']
73
74     for expr in exprAry :
75         top = -1
76         print(expr, '==>', checkBracket(expr))
```

실행 결과

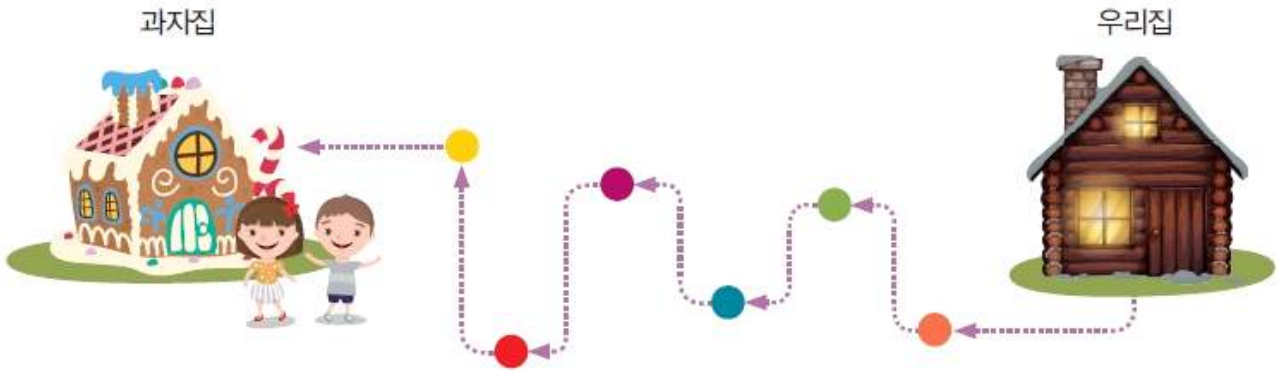
```
(A+B) ==> True
)A+B( ==> False
((A+B)-C ==> False
(A+B] ==> False
(<A+{B-C}/[C*D]>) ==> True
```

응용예제 01 헨젤과 그레텔의 집으로 돌아가기

난이도 ★☆☆☆☆

예제 설명

헨젤과 그레텔이 돌을 떨어뜨리면서 숲으로 들어간다. 과자집에서 집으로 돌아갈 때는 떨어뜨린 순서와 반대로 돌을 주워야 한다. 스택을 이용해서 집으로 무사히 돌아가도록 하자.

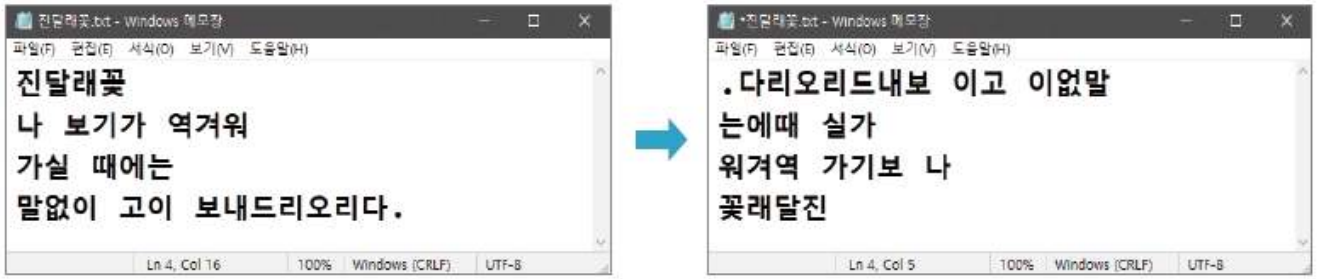


실행 결과

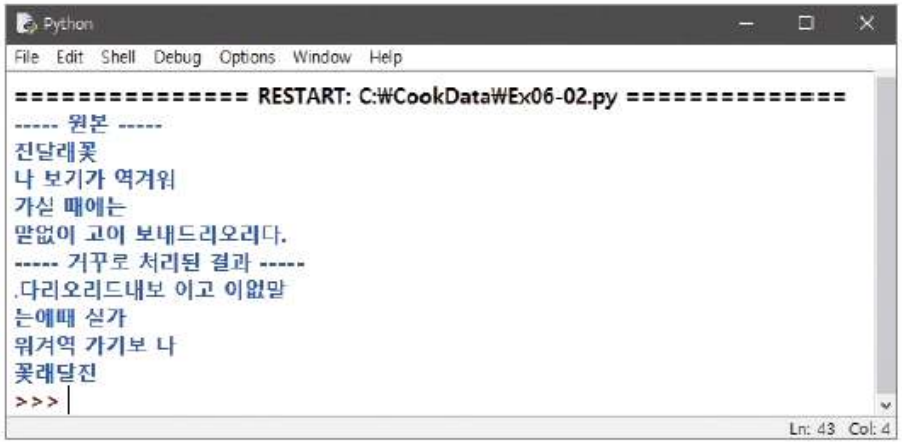
응용예제 02 파일 내용을 완전히 거꾸로 출력하기

난이도★★★☆☆

예제 설명 텍스트 파일에서 전체 줄을 거꾸로 하고, 각 줄의 글자도 거꾸로 출력하는 프로그램을 작성한다. 다음은 네 줄짜리 파일을 처리한 결과다.



실행 결과



Thank You