

06

CHAPTER

스택

학습목표

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

SECTION 00 생활 속 자료구조와 알고리즘

SECTION 01 스택의 기본

SECTION 02 스택의 간단 구현

SECTION 03 스택의 일반 구현

SECTION 04 스택의 응용

연습문제

응용예제



■ 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

커피 종이컵 넣기 녹차 종이컵 넣기 꿀물 종이컵 넣기

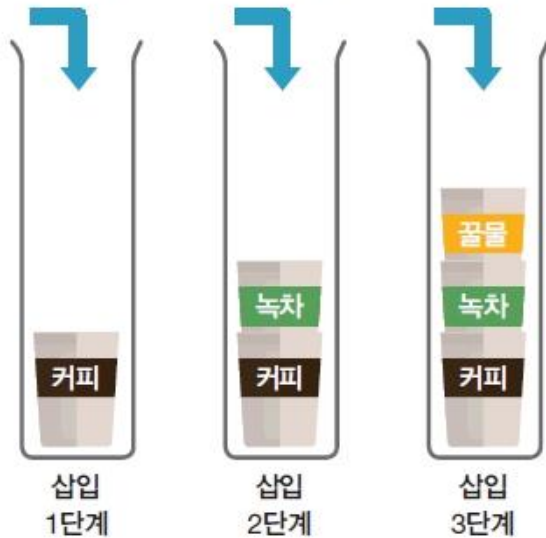


그림 6-2 종이컵 수거함의 작동 원리

Section 01 스택의 기본

■ 스택 원리

■ 스택 기본 구조

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

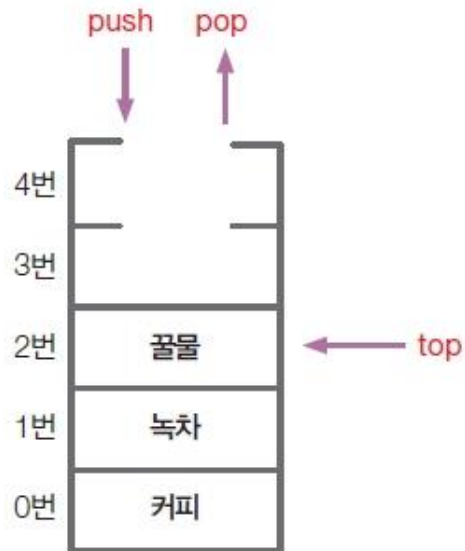


그림 6-3 스택 기본 구조

Section 01 스택의 기본

- 데이터 삽입 : push

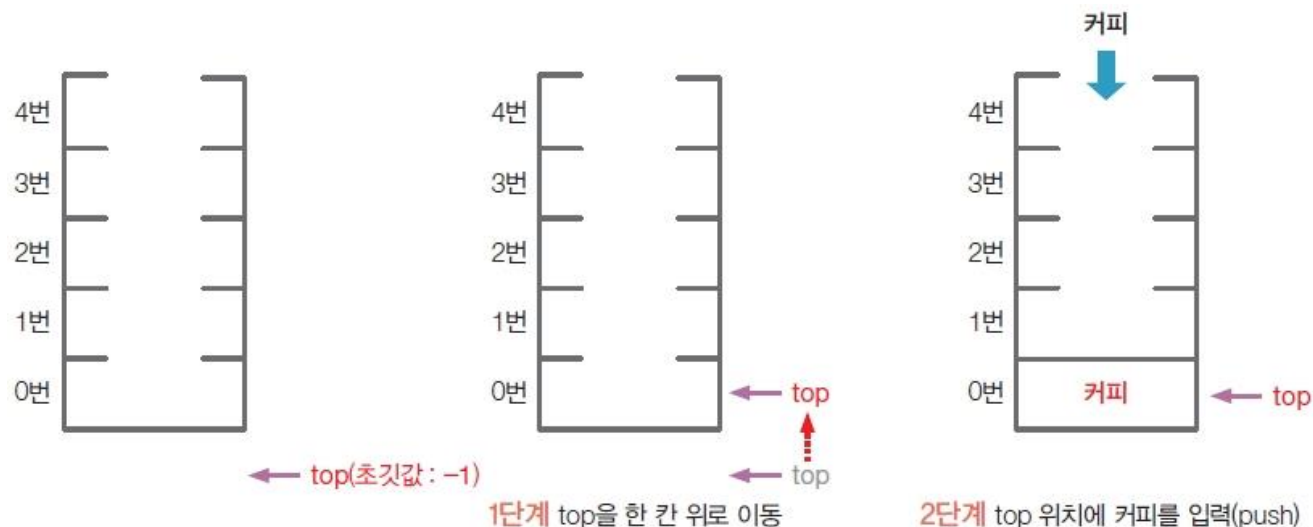


그림 6-4 스택에 데이터를 삽입(push)하는 과정

- 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
```

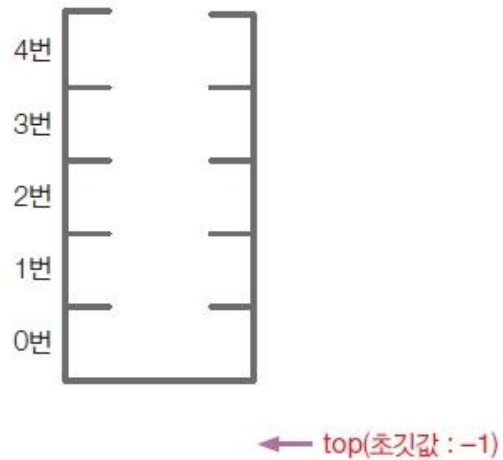


그림 6-6 크기가 5칸인 스택의 초기 상태

Section 02 스택의 간단 구현

■ 데이터 삽입 : push

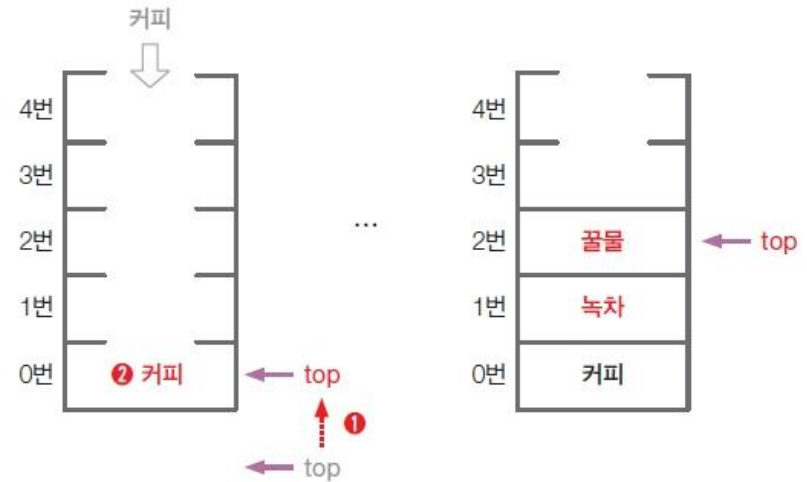


그림 6-7 스택의 데이터 삽입 과정

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

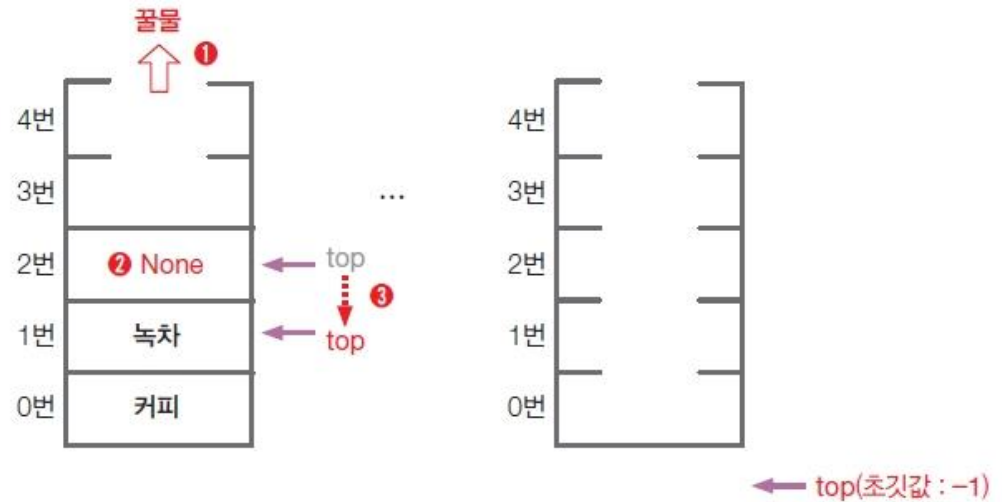


그림 6-8 스택의 데이터 추출

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
```

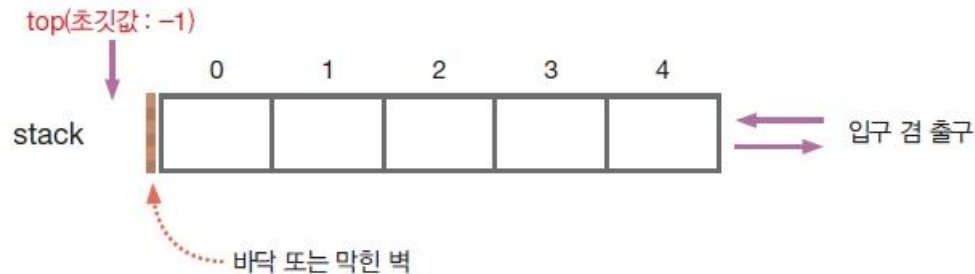


그림 6-9 초기화된 스택

Section 03 스택의 일반 구현

■ 데이터 삽입 과정

- 스택이 꽉 찼는지 확인하는 함수
 - top 값이 '스택 크기-1'과 같다면 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

- 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

■ 데이터 추출 과정

- 스택이 비었는지 확인하는 함수
 - top 값이 -1이라면 스택은 비어 있는 상태

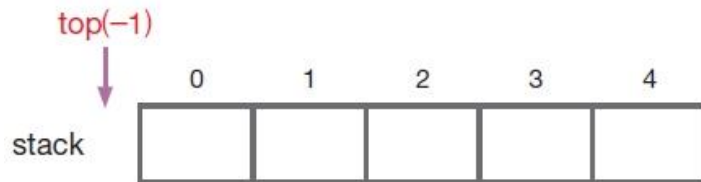


그림 6-11 스택이 비어 있는 상태

```
if (top값 == -1) :  
    스택이 비었음
```

Code06-05.py 스택이 비었는지 확인하는 함수

```
1 def isEmpty() :  
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 ==>", isEmpty())
```

실행 결과

스택이 비었는지 여부 ==> True

Section 03 스택의 일반 구현

■ 스택에서 데이터를 추출하는 함수

Code06-06.py 스택에서 데이터를 추출하는 함수

```
1 def isEmpty() :  
... # 생략(Code06-05.py의 2~6행과 동일)  
7  
8 def pop() :  
9     global SIZE, stack, top  
10    if (isEmpty()) :  
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

■ 데이터 확인

- top 위치의 데이터를 확인만 하고 스택에 그대로 두는 것 : peek(픽)



그림 6-12 데이터를 확인하는 peek 작동

Section 03 스택의 일반 구현

Code06-07.py 스택에서 top 위치의 데이터를 확인하는 함수

```
1 def isEmpty() :
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 (isEmpty()) :
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]
```

Section 03 스택의 일반 구현

■ 스택 완성

Code06-08.py 스택 작동을 위한 통합 코드

```
1  ## 함수 선언 부분 ##
2  def isStackFull() :                # 스택이 꽉 찼는지 확인하는 함수
3      global SIZE, stack, top
4      if (top >= SIZE-1) :
5          return True
6      else :
7          return False
8
9  def isStackEmpty() :              # 스택이 비었는지 확인하는 함수
10     global SIZE, stack, top
11     if (top == -1) :
12         return True
13     else :
14         return False
15
16 def push(data) :                   # 스택에 데이터를 삽입하는 함수
17     global SIZE, stack, top
18     if (isStackFull()) :
19         print("스택이 꽉 찼습니다.")
20         return
21     top += 1
22     stack[top] = data
23
```

Section 03 스택의 일반 구현

```
24 def pop() :                                # 스택에서 데이터를 추출하는 함수
25     global SIZE, stack, top
26     if (isEmpty()) :
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 (isEmpty()) :
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!

Section 04 스택의 응용

■ 웹 서핑에서 이전 페이지 돌아가기

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

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

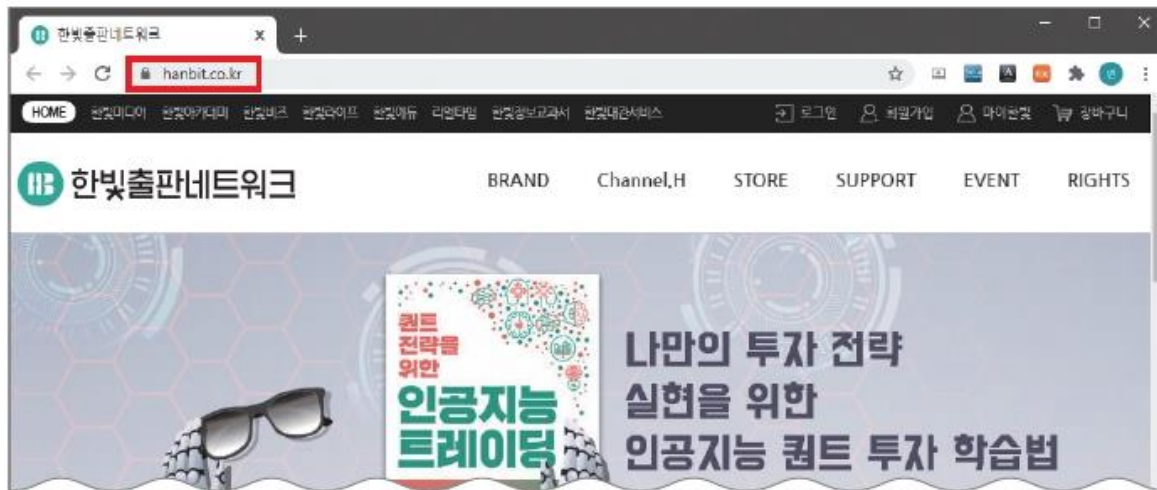


그림 6-13 webbrowser.open() 함수로 실행한 웹 브라우저

Section 04 스택의 응용

- 방문한 URL을 스택에 push하는 과정

NAVER → Daum → nate



nate → Daum → NAVER



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

■ 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.
- $(\langle A+\{B-C\} / [C*D]\rangle)$: 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

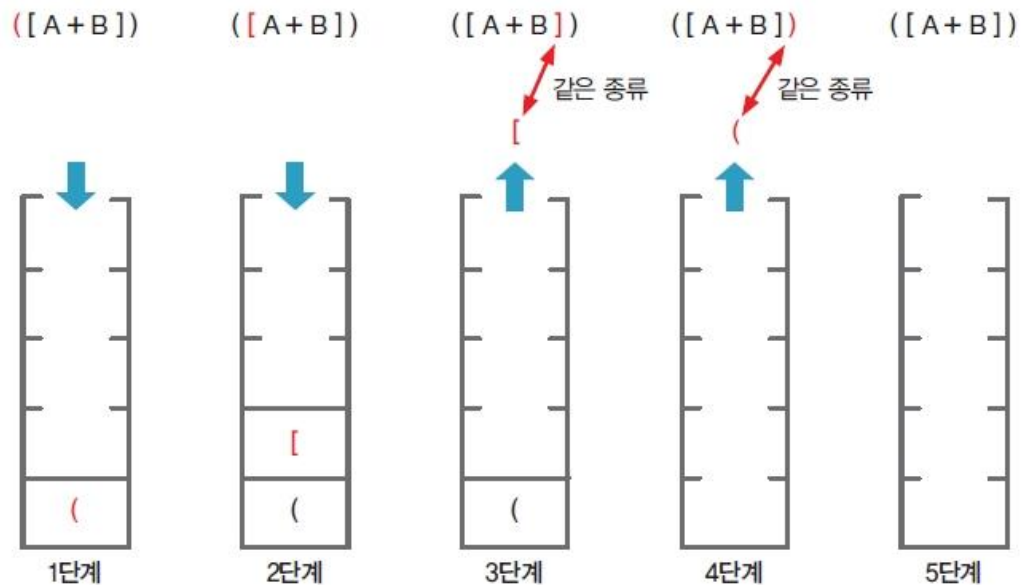


그림 6-16 스택에서 괄호 검사를 처리하는 과정 예 : $([A+B])$

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

```
1  ## 함수 선언 부분 ##
... # 생략(Code06-08.py의 2~39행과 동일)
40
41 def checkBracket(expr) :
42     for ch in expr :
43         if ch in '([{' :
44             push(ch)
45         elif ch in ')]>' :
46             out = pop()
47             if ch == ')' and out == '(' :
48                 pass
49             elif ch == ']' and out == '[' :
50                 pass
51             elif ch == '}' and out == '{' :
52                 pass
53             elif ch == '>' and out == '<' :
54                 pass
55             else :
56                 return False
57     else :
58         pass
59     if isEmpty() :
60         return True
61     else :
62         return False
63
```

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

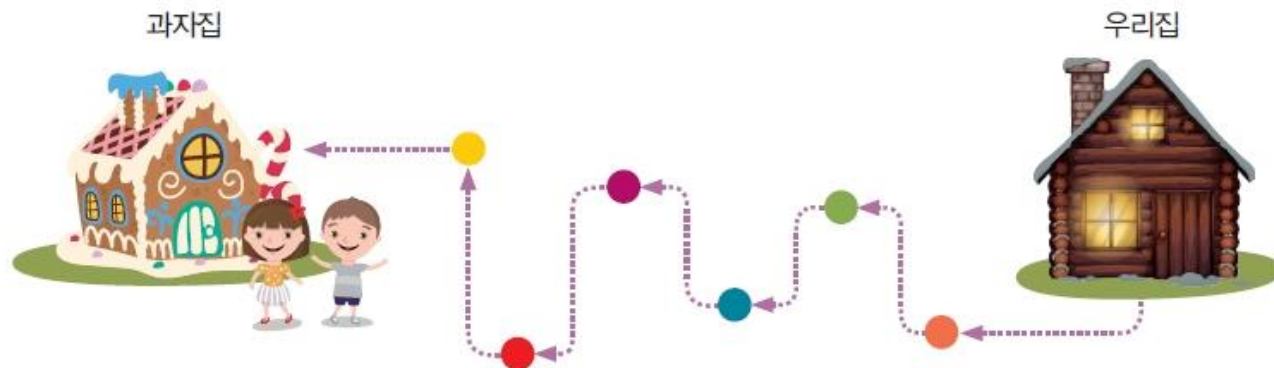
```
(A+B) ==> True
)A+B( ==> False
((A+B)-C ==> False
(A+B] ==> False
<A+{B-C}/[C*D]> ==> True
```

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

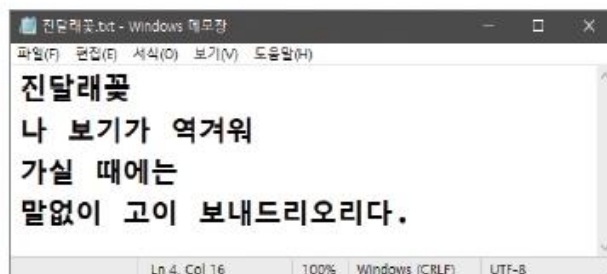
```
Python
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookData\Ex06-01.py =====
과자집에 가는길 : 주황 --> 초록 --> 파랑 --> 보라 --> 빨강 --> 노랑 --> 과자집
우리집에 오는길 : 노랑 --> 빨강 --> 보라 --> 파랑 --> 초록 --> 주황 --> 우리집
>>> |
```


응용예제 02 파일 내용을 완전히 거꾸로 출력하기

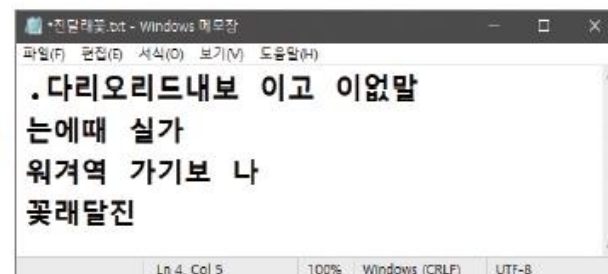
난이도★★★★☆

예제 설명

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



진달래꽃
나 보기가 역겨워
가실 때에는
말없이 고이 보내드리오리다.



.다리오리드내보 이고 이없말
는에때 실가
워겨역 가기보 나
꽃래달진

실행 결과



```
==== RESTART: C:\CookData\WEx06-02.py =====  
----- 원본 -----  
진달래꽃  
나 보기가 역겨워  
가실 때에는  
말없이 고이 보내드리오리다.  
----- 거꾸로 처리된 결과 -----  
.다리오리드내보 이고 이없말  
는에때 실가  
워겨역 가기보 나  
꽃래달진  
>>>
```