32장

직접 만든 객체 타입 사용하기

32장 직접 만든 객체 타입 사용하기

32.1 스택 객체 정의하기

32.1 스택 객체 정의하기

26장에서 여러 append와 pop 연산을 리스트에 적용하여 스택을 구현해봤다.

- 연산을 수행할 때 각 연산이 스택의 동작과 일치하도록 주의를 기울였다. 즉, 항상 리스트의 맨 뒤에 원소를 추가하고, 리스트의 맨 뒤에서 원소를 제거했다.
- 클래스를 사용하면 스택 객체가 여러분 대신 스택에 적용해야 하는 규칙을 준수하게 만들 수 있다.
- 이런 스택 타입이 있으면 프로그램을 짤 때 스택 규칙을 지키기 위해 일일이 노력할 필요가 없다.

32.1 스택 객체 정의하기

```
class Stack(object):
    def __init__( self):
        self.stack = [] ---- 스택을 정의하는 데이터 속성인 리스트

    def get_stack_elements(self):
        return self.stack.copy() ---- 스택을 표현하는 데이터 속성의 복사본을 반환하는 메서드

    def add_one(self , item):
        self.stack.append(item) ---- 스택에 원소를 추가하는 메서드. 리스트 맨 끝에 원소를 추가함

    def add_many(self , item, n):
        for i in range(n):
            self.stack.append(item) ---- 같은 원소를 n개 추가하는 메서드

    def remove_one(self):
        self.stack.pop() ---- 원소를 하나 제거하는 메서드

    def remove_many(self , n):
        for i in range(n):
            self.stack.pop() ---- 스택에서 n개의 원소를 제거하는 메서드

    def size(self):
        return len(self.stack) ---- 스택에 들어 있는 원소의 개수를 알려주는 메서드

    def prettyprint(self):
        for thing in self.stack[::-1]:
            print('|_',thing, '_|') ---- 한 줄에 한 원소씩 스택을 출력하는 메서드.
                                        최근에 들어온 원소를 더 위쪽에 출력함
```

코드 32-1
Stack 클래스 정의

셀프 체크 32.1

32.2 Stack 객체 사용하기

32.2.1 부침개 스택 만들기

» 첫 단계로 부침개(pancakes)를 추가할 스택을 만든다.

- Stack 객체를 초기화해 빈 스택을 만든다.
- add_one을 사용해 스택에 해물 부침개를 하나 추가한다.
- add_many를 사용해 스택에 감자 부침개를 네 개 추가한다.

```
buchimgae = Stack() ---- Stack 객체를 만들어서 buchimgae라는 이름의 변수에 대입
buchimgae.add_one("해물") ---- 해물 부침개를 하나 추가
buchimgae.add_many("감자", 4) ---- 감자 부침개를 4개 추가
print(buchimgae.size()) ---- 5를 출력
buchimgae.remove_one() ---- 맨 마지막에 추가했던 문자열을 제거('감자'를 제거)
print(buchimgae.size()) ---- 4를 출력
buchimgae.prettyprint() ---- 한 줄에 하나씩 부침개를 출력.
                             감자가 먼저 세 번 출력되고 맨 마지막에 해물 출력
```

코드 32-2
Stack 객체를 만들고 부침개 추가하기

32.2.1 부침개 스택 만들기

» 그림 32-1은 스택에 원소를 추가하는 단계와 self.stack으로 접근할 수 있는 리스트 데이터 속성의 값을 나타낸 것이다.

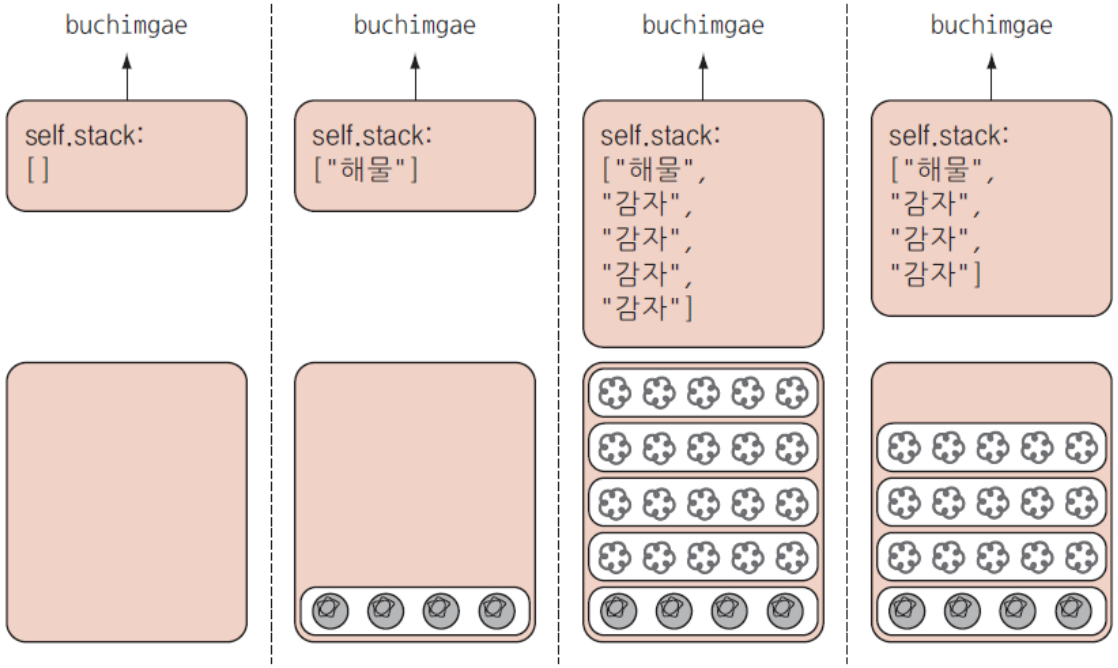


그림 32-1
(왼쪽부터) 첫 번째 패널은 빈 부침개 스택이다. 두 번째 패널은 '해물'을 하나 추가한 이후 스택의 모습이다. 세 번째는 감자를 한꺼번에 네 개 추가한 이후 스택의 상태다. 마지막 패널은 원소를 하나 제거한 다음의 상태다. 맨 마지막에 추가했던 '감자'가 하나 제거된다

32.2.2 원으로 이뤄진 스택 만들기

» Stack 객체가 있으면 원자적인 객체(int, float, bool등) 뿐 아니라 어떤 타입의 객체든 스택에 추가할 수 있다.

- 코드32-3 : 원을 표현하는 Circle 타입 객체를 생성해 스택에 추가

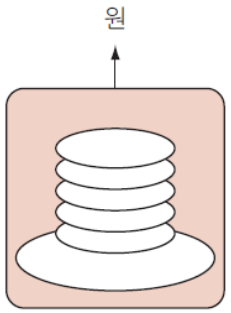
```
circles = Stack() ---- 스택을 만들고 그 객체를 circles라는 변수에 대입
one_circle = Circle()
one_circle.change_radius(2)
circles.add_one(one_circle)

for i in range(5): ---- 새 Circle 객체를 다섯 개 추가하는 루프
    one_circle = Circle()
    one_circle.change_radius(1)
    circles.add_one(one_circle)

print(circles.size()) ---- 6을 출력
circles.prettyprint() ---- 각 Circle 객체와 연관된 파이썬 정보(타입과 메모리 상의 위치)를 출력
```

새 Circle 객체를 만들고 반지름을 2로 설정한 후 스택에 추가

루프를 돌 때마다 새 Circle 객체를 만들어서 반지름을 1로 설정한 후 스택에 추가



코드 32-3
Stack 객체를 만들고 Circle 객체를 추가하기

32.2.2 원으로 이뤄진 스택 만들기

» 다음 코드는 루프를 돌면서 한 번에 하나씩 원을 추가하는 대신, 반지름이 1인 원을 만들고, 그 원을 스택의 add_many를 호출해 여러 번 스택에 넣었다.

```
circles = Stack()
one_circle = Circle()
one_circle.change_radius(2)
circles.add_one(one_circle)

one_circle = Circle()
one_circle.change_radius(1)
circles.add_many(one_circle, 5)

print(circles.size())
circles.prettyprint()
```

코드 32-3과 같은 연산

새 Circle 객체를 만들고 반지름을 1로 설정

똑같은 Circle 객체를 Stack 클래스의 메서드를 사용해 다섯 번 추가

이 시점에 스택에 들어 있는 원의 개수인 6을 출력

각 Circle 객체와 연관된 파이썬 정보(타입과 메모리 상의 위치)를 출력

코드 32-4
Stack 객체를 만들고 같은 Circle 객체를 여러 번 넣기

32.2.2 원으로 이뤄진 스택 만들기

코드 32-3과 32-4의 두 스택을 비교해 보자.

- 코드 32-3에서는 루프를 돌면서 매번 새 Circle 객체를 만들었다.

```
|_ <__main__.Circle object at 0x00000200B8B90BA8> _|
|_ <__main__.Circle object at 0x00000200B8B90F98> _|
|_ <__main__.Circle object at 0x00000200B8B90EF0> _|
|_ <__main__.Circle object at 0x00000200B8B90710> _|
|_ <__main__.Circle object at 0x00000200B8B7BA58> _|
|_ <__main__.Circle object at 0x00000200B8B7BF28> _|
```

- 코드 32-4에서는 Circle 객체를 하나만 만들고, 한 객체를 스택에 다섯 번 추가했다.

```
|_ <__main__.Circle object at 0x00000200B8B7BA58> _|
|_ <__main__.Circle object at 0x00000200B8B7BA58> _|
|_ <__main__.Circle object at 0x00000200B8B7BA58> _|
|_ <__main__.Circle object at 0x00000200B8B7BA58> _|
|_ <__main__.Circle object at 0x00000200B8B7BA58> _|
|_ <__main__.Circle object at 0x000001F1E0E0CA90> _|
```

셀프 체크 32.2

32.3 요약

32.3 요약

- 클래스를 정의하려면 그 클래스를 어떻게 표현할지 결정해야 한다.
- 또한, 클래스를 정의하려면 클래스를 어떤 메서드를 통해 어떻게 사용할지 결정해야 한다.메서드를 어떻게 구현할지도 결정해야 한다.
- 클래스는 한 객체 타입 안에 프로퍼티와 동작을 패키지화한다. 그 클래스에 속한 모든 객체는 똑같은 데이터와 메서드를 공유한다.
- 클래스를 사용하려면 해당 타입의 객체를 하나 이상 만들어야 하고, 일련의 연산을 그 객체에 적용해야 한다.

32.3 요약

» **(Q32.1) 큐(queue)를 표현하는 클래스를 스택과 비슷한 방식으로 작성하라. 큐에 원소를 추가할 때는 한쪽 끝에 추가하고, 원소를 꺼낼 때는 반대쪽 끝에서 꺼낸다는 사실을 기억하라.**

- 큐를 어떤 데이터 구조를 사용해 표현할지 결정하라.
- _ _init_ _를 구현하라.
- 큐의 크기를 얻는 연산, 원소를 하나 추가하는 연산, 원소 하나를 여러 번 추가하는 연산, 원소 하나를 제거하는 연산, 여러 원소를 제거하는 연산, 큐를 보여 주는 연산을 메서드로 구현하라.
- 큐 객체를 만들고 그 객체에 대해 여러 연산을 수행하는 코드를 작성하