고급통계프로그래밍 Solution 8

컴퓨터과학부 2017920054 이호준

과제 index는 번역본 기준입니다.

Ex 17.2

Solution

```
class Point():
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
```

- Point class를 선언한다.
  - `__init__` : Python의 생성자 키워드
    - x와 y를 인자로 받아서 이를 각각 self.x와 self.y의 값으로 대입한다.
    - 이 때, x와 y는 매개변수, self.x와 self.y는 Point class의 attribute로 이름만 같을 뿐 namespace에 차이가 있다.

Result

```
# Test
a = Point(3, 4)
print(a.x, a.y)

# in Stdout:
3 4
```

Ex 17.3

Solution

```
class Point():
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return ("점 ({}, {})".format(self.x, self.y))
```

- `__str__` : Python에서 `print()` 시 객체를 출력하는 방법
  - `.format()` 메서드를 이용하면 str에 있는 {}를 매개변수의 순서대로 치환해줄 수 있다.
    - 첫 번째 {}에는 self.x가 들어가며, 두 번째 {}에는 self.y가 들어간다. 따라서 **점 (x, y)** 형태로 출력이 된다.

Result

```
# Test 1
a = Point(3, 4)
print(a)

# in Stdout 1:
점 (3, 4)
```

```
# Test 2
b = Point(8, 2)
print(b)

# in Stdout 2:
점 (8, 2)
```

Ex 17.4

Solution

```
class Point():
    ...

    def __add__(self, other):
        return (self.x + other.x, self.y + other.y)
```

- `__add__` : Python에서 `print()` 시 객체를 출력하는 방법
- other를 인자로 받은 다음, self와 other의 각 element(x, y)를 elementwise하게 더한 후 이를 tuple로 묶어서 반환한다.

Result

```
# Test
a = Point(3, 4)
b = Point(8, 2)

print(a + b)

# in Stdout:
(11, 6)
```

Ex 17.5

Solution

```
class Point():
    ...

    def __add__(self, other):
        if isinstance(other, Point):
            return self.point_add(other)
        elif isinstance(other, tuple):
            return self.tuple_add(other)

    def point_add(self, other):
        return (self.x + other.x, self.y + other.y)

    def tuple_add(self, other):
        return (self.x + other[0], self.y + other[1])
```

- 인자 other의 class 비교를 통해서 다른 로직을 수행한다.
  - `point_add` (other가 Point)인 경우:
    - other에 .(dot access)를 통한 멤버 접근으로 x, y를 elementwise하게 더한 후 이를 tuple로 묶어서 반환
  - `tuple_add` (other가 tuple)인 경우:
    - other에 인덱스 0, 1를 통한 인덱싱으로 x, y를 elementwise하게 더한 후 이를 tuple로 묶어서 반환

Result

```
# Test 1 - with Point
a = Point(3, 4)
b = Point(8, 2)

print(a + b)
```

```
# in Stdout 1:
(11, 6)
```

```
# Test 2 - with tuple
a = Point(3, 4)
print(a + (8, 2))

# in Stdout 2:
(11, 6)
```

Ex 18 - 2 / 3

Solution

`Card` Class

```
import random

class Card():
    """
    suit : clubs, diamonds, hearts, spades
    rank_names = None, Ace, 2, ...
    """

    suit_names = ['Clubs', 'Diamonds', 'Hearts', 'Spades']
    rank_names = [None, 'Ace', '2', '3', '4', '5', '6', '7', '8', '9', '10',
                    'Jack', 'Queen', 'King']

    def __init__(self, suit=0, rank=2):
        self.suit = suit
        self.rank = rank

    def __str__(self):
        return "{} of {}".format(Card.rank_names[self.rank], Card.suit_names[self.suit])

    def __repr__(self):
        return "{} of {}".format(Card.rank_names[self.rank], Card.suit_names[self.suit])

    def __lt__(self, other):
        t1 = self.suit, self.rank
        t2 = other.suit, other.rank

        if t1[0] < t2[0]:
            return True
        if t1[0] > t2[0]:
            return False
        if t1[1] < t2[1]:
            return True
        if t1[1] > t2[1]:
            return False
        return False
```

- `__init__` : 파이썬이 객체를 다룰 때 각 객체를 나타내는(represent) 문자열을 지정
- `__repr__` : 파이썬이 객체를 다룰 때 각 객체를 나타내는(represent) 문자열을 지정
- `__str__` : 객체 출력(with `print()`)을 진행할 때 출력될 객체의 양식을 지정
- `__lt__` : `<` 연산자를 오버라이딩하는 특수 메서드 (**less than**)
  - 튜플 t1, t2를 생성하고, cmp의 조건에 맞게 비교하여 True, False를 반환하도록 함.

```
class Deck():

    def __init__(self):
        self.cards = []
        for suit in range(4):
            for rank in range(1, 14):
                card = Card(suit, rank)
                self.cards.append(card)

    def __str__(self):
        return '\n'.join([str(card) for card in self.cards])

    def add_card(self, card):
        self.cards.append(card)

    def pop_card(self):
        return self.cards.pop()

    def shuffle(self):
        random.shuffle(self.cards)

    # Solution 18-2
    def sort(self):
        self.cards.sort()

    def move_cards(self, hand, num):
        for _ in range(num):
            hand.add_card(self.pop_card())

    # Solution 18-3
    def deal_hands(self, n_of_participant, n_of_cards):
        game = []
        for _ in range(n_of_participant):
            one_hand = Hand()
            self.move_cards(one_hand, n_of_cards)
            game.append(one_hand)
        return game
```

Solution 18-2

- `sort()` - self.cards(리스트) 를 정렬하는 함수
  - 위 `Card` 클래스에서 `__lt__` 메서드를 정의하였으므로(`<` 를 오버라이딩하였으므로) 이를 사용해서 자동적으로 .sort()가 진행된다. 따라서 일반적인 리스트의 정렬방법과 똑같이 `.sort()` 를 호출해주기만 하면 된다.

Solution 18-3

- `deal_hands()` - 참가자에게 일정한 카드개수만큼 분배하는 함수
  - `n_of_participant`(참가자 수)와 `n_of_cards`(인당 카드 수)를 인자로 받음
  - Hand 객체의 모음을 담을 리스트 `game` 을 선언
  - Hand 객체 :

```
class Hand(Deck):
    def __init__(self, label=''):
        self.cards = []
        self.label = label

    def __repr__(self):
        return "{}".format(self.cards)
```

  - Hand 객체의 모음을 담을 리스트 `game` 을 선언
  - `n_of_participant` 번만큼 반복을 진행:
    - Hand 객체를 하나 생성하고(`one_hand`), 위에 만들어둔 `move_cards()` 메서드를 이용해 Deck으로부터 Hand으로 카드를 할당
  - 반복이 종료되면 Hand 객체가 `n_of_participant`개 담겨있는 리스트 `game`을 반환한다.

Result

```
# Test ex18-2 - I
card1 = Card(2, 11)
card2 = Card(2, 12)
print(card1 < card2)

# in Stdout I:
True
```

```
# Test ex18-2 - II
card1 = Card(3, 2)
card2 = Card(0, 5)
print(card1 < card2)

# in Stdout II:
False
```

```
# Test ex18-3 - I
deck = Deck()
deck.shuffle()
print("deck에서 뽑아내기 :", deck.deal_hands(4, 13), sep="\n")
print("남은 deck :", deck, sep="\n")

# in Stdout I:
deck에서 뽑아내기 :
[[10 of Spades, King of Spades, Jack of Spades, 2 of Spades, 6 of Hearts,
9 of Spades, Ace of Clubs, 9 of Clubs, 9 of Diamonds, Ace of Spades,
Jack of Clubs, 7 of Hearts, 3 of Spades], [Ace of Hearts, 3 of Clubs,
5 of Hearts, 9 of Hearts, 8 of Clubs, 3 of Hearts, 5 of Clubs, Queen of Clubs,
King of Hearts, 2 of Clubs, Ace of Diamonds, 2 of Diamonds, Jack of Hearts],
[4 of Clubs, 6 of Spades, 7 of Diamonds, 6 of Diamonds, 2 of Hearts, 7 of Spades,
10 of Hearts, 5 of Diamonds, 4 of Spades, Queen of Hearts, 10 of Diamonds,
Jack of Diamonds, 3 of Diamonds], [8 of Spades, 10 of Clubs, 8 of Diamonds,
4 of Hearts, King of Diamonds, 4 of Diamonds, 5 of Spades, Queen of Diamonds,
Queen of Spades, 8 of Hearts, 6 of Clubs, 7 of Clubs, King of Clubs]]
남은 deck :
```

```
# Test ex18-3 - II
deck = Deck()
deck.shuffle()
print("deck에서 뽑아내기 :", deck.deal_hands(7, 7), sep="\n")
print("남은 deck :", deck, sep="\n")

# in Stdout II:
deck에서 뽑아내기 :
[[5 of Clubs, 3 of Clubs, 7 of Diamonds, 9 of Hearts, Queen of Diamonds,
 8 of Spades, 9 of Clubs], [Queen of Clubs, 7 of Clubs, 6 of Diamonds,
 Jack of Diamonds, King of Diamonds, Ace of Clubs, Ace of Hearts],
[9 of Diamonds, Queen of Hearts, 6 of Clubs, 2 of Clubs, 10 of Clubs,
 3 of Hearts, 6 of Hearts], [10 of Spades, 3 of Diamonds, 9 of Spades,
 6 of Spades, 7 of Hearts, 5 of Spades, 10 of Hearts], [2 of Diamonds,
 8 of Clubs, 4 of Hearts, King of Clubs, Ace of Spades, Queen of Spades,
 Jack of Clubs], [4 of Diamonds, 10 of Diamonds, King of Spades, 7 of Spades,
 Jack of Hearts, 2 of Hearts, Ace of Diamonds], [8 of Hearts, Jack of Spades,
8 of Diamonds, 4 of Clubs, 3 of Spades, 5 of Hearts, 5 of Diamonds]]
남은 deck :
4 of Spades
King of Hearts
2 of Spades
```