고급통계프로그래밍 # 8

2017580034 통계학과 이유민

(#1) Ex17.2

x 와 y를 선택적인 매개변수로 받아서 해당 애트리뷰트에 대입하도록 Point 클래스의 init 메쏘드를 작성하세요.

In [1]:

```
class Point:
    def __init__(self, x=0, y=0): #init method
        self.x = x #x,y를 해당 애트리뷰트에 대입
        self.y = y
```

(#2) Ex17.3

Point 클래스의 str 메쏘드를 작성하세요. Point 객체를 만들고 인쇄하세요.

In [2]:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __str__(self): # str method
        return 'X is %2d and Y is %2d' %(self.x, self.y) #x,y좌표 return
```

In [3]:

```
p = Point(1,5)
print(p) #Point 객체 print
```

```
X is  1 and Y is  5
```

(#3) Ex17.4

Point 클래스의 add 메쏘드를 작성하세요.

In [4]:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __str__(self):
        return 'X is %2d and Y is %2d' %(self.x, self.y)
    def __add__(self, other): #add method
        x = self.x + other.x #x,y 각각에 add
        y = self.y + other.y
        return x,y
```

In [5]:

```
p1 = Point(1,9)
p2 = Point(2,5)
print(p1+p2) #p1, p2 add 결과 print
```

```
(3, 14)
```

(#4) Ex17.5

Point 객체나 튜플과 함께 쓸 수 있는 Point 의 add 메쏘드를 작성하세요:

만약 두 번째 피 연산자가 Point 이면, 메쏘드는 x 좌표가 피 연산자들의 x 좌표들의 합이고, y 좌표도 마찬가지인 새 Point 를 돌려줘야 합니다. 만약 두 번째 피 연산자가 튜플이면, 이 메쏘드는 튜플의 첫 번째 요소를 x 좌표에, 두번째 요소를 y 좌표에 더한 결과를 갖는 새 Point 를 돌려줘야 합니다.

In [6]:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __str__(self):
        return 'X is %2d and Y is %2d' %(self.x, self.y)
    def __add__(self, other):
        if isinstance(other, Point): #두번째 피연산자가 Point
            return Point(self.x+other.x, self.y+other.y) #x,y 각각의 좌표 합한 새 point
        if isinstance(other, tuple): #두번째 피연산자가 tuple
            other_x = other[0] # 튜플 첫번째 요소를 x좌표에
            other_y = other[1] # 튜플 두번째 요소를 x좌표에
            return Point(self.x+other.x, self.y+other.y)
```

In [7]:

```
p1 = Point(2,3)
p2 = Point(1,4)
print(p1+p2)
```

```
X is  3 and Y is  7
```

(#5) Ex18.2

리스트 메쏘드 sort 를 사용해서 Deck 에 있는 카드들을 정렬하는 Deck 메쏘드 sort를 작성하세요. sort 는 정렬 순서를 정하기 위해 정의한 **cmp** 메쏘드를 사용합니다.

In [8]:

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

In [9]:

```
class Card:
    def __init__(self, suit=0, rank=2):
        self.suit=suit
        self.rank=rank
    def __str__(self):
        return '%s of %s' %(rank_names[self.rank],
                            suit_names[self.suit])
    def cmp(self, other):
         return (self>other) - (other>self)
```

In [10]:

```
class Deck: #Deck에 있는 카드 정렬
    def __init__(self):
        self.cards=[] #card 애트리뷰트
        for suit in range(4):
            for rank in range(1,14):
                card=Card(suit, rank)
                self.cards.append(card)
                #카드 52장 생성
    def __str__(self):
        res=[]
        for card in self.cards:
            res.append(str(card))
        return 'Wn'.join(res) #줄 넘김 문자로 표현

    def sort(self):  #Deck에 있는 카드들 정렬하는 sort method
        self.cards.sort(Card.cmp)
```

In [11]:

```
card1=Card(2,11)
card2=Card(1,12)
d=Deck()
print(d)
```

```
7 of Hearts
8 of Hearts
9 of Hearts
10 of Hearts
Jack of Hearts
Queen of Hearts
King of Hearts
Ace of Spades
2 of Spades
3 of Spades
4 of Spades
5 of Spades
6 of Spades
7 of Spades
8 of Spades
9 of Spades
10 of Spades
Jack of Spades
Queen of Spades
```

(#6) Ex18.3

두 개의 매개변수, 패를 든 사람의 수와 패당 카드 수,를 받아서 새 Hand 객체들을 만들고, 패마다 적당한 수의 카드를 나눈 다음, Hand 객체의 리스트를 돌려주는 Deck 이 메쏘드 deal_hands를 작성하세요.

In [12]:

```
class Card(object):
    def __init__(self, suit=0, rank=2):
        self.suit = suit
        self.rank = rank

    suit_names = ['Clubs', 'Diamonds', 'Hearts', 'Spades']
    rank_names = [None, 'Ace', '2', '3', '4', '5', '6', '7',
                  '8', '9', '10', 'Jack', 'Queen', 'King']
    def __str__(self):
        return '%s fo %s' % (Card.rank_names[self.rank],
                             Card.suit_names[self.suit])
```

In [13]:

```
import random #for shuffle

class Deck(object):

    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):
        res = []
        for card in self.cards:
            res.append(str(card))
        return 'Wn'.join(res)

    def pop_card(self): #카드 나누기
        return self.cards.pop() #list의 마지막 카드 제거

    def add_card(self, card): #카드 추가
        self.cards.append(card)

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

    def move_cards(self, hand, num): # num:나눌 카드 수
        for i in range(num):
            hand.add_card(self.pop_card()) #self / hand 수정 > 마지막에는None return

    def deal_hands(self, nhand, ncard): #nhand:사람수, ncard: 카드수
        for i in range(nhand):
            hand=Hand(str(i)) #hand 객체 생성
            for j in range(ncard):
                hand.add_card(self.pop_card()) #패마다 적당한 수의 카드 나눔
            print(hand) #hand object list print
            print('------next person-------')
```

In [14]:

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

In [15]:

```
d=Deck()
d.shuffle()
d.deal_hands(4,5)
#4 person 5 cards each
```

```
Ace fo Diamonds
3 fo Clubs
3 fo Spades
3 fo Diamonds
9 fo Spades
------next person-------
3 fo Hearts
Jack fo Diamonds
King fo Hearts
Ace fo Hearts
Ace fo Spades
------next person-------
4 fo Diamonds
Ace fo Clubs
King fo Diamonds
7 fo Clubs
9 fo Clubs
------next person-------
Queen fo Clubs
Queen fo Hearts
10 fo Spades
8 fo Hearts
4 fo Spades
------next person-------
```