Week 4. 연습문제이자 숙제이자 괴롭힘

1) 다음의 조건을 만족하는 Point라는 클래스를 작성하세요.

- Point 클래스는 생성자 `__init__` 를 통해 (x, y) 좌표를 입력받는다.
- setx(x), sety(y) 메서드를 통해 x 좌표와 y 좌표를 따로 입력받아 수정 할 수도 있다.
- get() 메서드를 호출하면 튜플로 구성된 (x, y) 좌표를 반환한다.
- move(dx, dy) 메서드는 현재 좌표를 dx, dy만큼 이동시킨다.

모든 메서드는 인스턴스 메서드(클래스 내부에서 만드는 `def`)다.

In [19]:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
        print("좌표가 ({x_value},{y_value})로 초기 설정되었습니다"
              .format(x_value=self.x, y_value=self.y))

    def setx(self, new_x):
        self.x = new_x

    def sety(self, new_y):
        self.y = new_y

    def get(self):
        return (self.x, self.y)

    def move(self, dx, dy):
        self.x += dx
        self.y += dy
        print("좌표가 ({x},{y})로 변경되었습니다".format(x=self.x, y=self.y))
```

2) 문제 1) 에서 생성한 Point 클래스에 대한 객체(인스턴스)를 생성한 후 위의 4개의 메서드를 테스트 해 보세요
~

In [26]:

```
a = Point(1,1)
b = Point()
c = Point(y=3)

a.setx(2)
a.sety(2)

print(a.get())

a.move(dx=2, dy=3)
```

```
좌표가 (1,1)로 초기 설정되었습니다
좌표가 (0,0)로 초기 설정되었습니다
좌표가 (0,3)로 초기 설정되었습니다
(2, 2)
좌표가 (4,5)로 변경되었습니다
```

a_list = [1,3,2,4]

a_list.sort(reverse=True) a_list

def sort(key=len,reverse=False): aslkdfjal;sdfj laskd;kdfj a;lsdkfjasd;

수업시간에 만들었던 Character 클래스 를 업그레이드 해 봅시다! Improvement!!

수업시간에 설계하고 만들어 보았던 Character 클래스에 기능과 클래스변수, 객체(인스턴스)변수, 메서드들을 추가하여 여러가지 기능을 작동 시켜봅시다.

3) 캐릭터를 삭제하는 기능을 추가해 보세요.

In [45]:

```
class Character:
    num_characters = 0 # 모든 캐릭터의 개체 수

    def __init__(self, name, input_life=100): # __init__의 input 인자는 name
        self.name = name #본 객체의 변수인 name에 input으로 받았던 이름을 넣어줌
        self.life = input_life # 모든 Character 객체는 life 가 1000 을 가짐
        self.exp = 0 # 경험치는 0부터 시작
        self.__hidden_dps = 0.5 # 히든 dps
        print('이 캐릭터의 이름이 %s 로 정해졌습니다.' %(self.name))
        Character.num_characters += 1

    def attacked(self):
        self.life -= 10 # 공격을 받으면 life에서 10씩 감소함
        print('%s 이 공격을 받아 생명력이 %d 로 줄었습니다.' %(self.name, self.life))

    def attack(self, other):
        self.exp += 50 # 경험치 50 증가
        print('%s 가 %s 를 공격해서 경험치가 50 올랐습니다.' %(self.name, other.name))

    def __add__(self, other):
        print("%s 와 %s 가 파티를 맺었습니다." % (self.name, other.name))

    @classmethod
    def how_many_character(cls):
        print ("생성된 총 캐릭터 수:", cls.num_characters)

    def delete_character(self):
        print("{} 를 삭제합니다.".format(self.name))
        del self
        Character.num_characters -= 1
```

In [46]:

```
user1 = Character("한조", 1000)
print(user1.life)
user2 = Character("메르시")
print(user2.life)

Character.how_many_character()

user2.delete_character()
Character.how_many_character()
```

```
이 캐릭터의 이름이 한조 로 정해졌습니다.
1000
이 캐릭터의 이름이 메르시 로 정해졌습니다.
100
생성된 총 캐릭터 수: 2
메르시 를 삭제합니다.
생성된 총 캐릭터 수: 1
```

4) "공격하는 메서드(attack)를 수정하여 내가 공격을 했을 때, 생대방은 공격을 받는 기능이 (동시에 또는 한번에) 구현되도록 코드를 수정해 보세요.

In [47]:

```
class Character:
    num_characters = 0 # 모든 캐릭터의 개체 수

    def __init__(self, name): # __init__의 input 인자는 name
        self.name = name #본 객체의 변수인 name에 input으로 받았던 이름을 넣어줌
        self.life = 100 # 모든 Character 객체는 life 가 100 을 가짐
        self.exp = 0 # 경험치는 0부터 시작
        self.__hidden_dps = 0.5 # 히든 dps
        print('이 캐릭터의 이름이 %s 로 정해졌습니다.' %(self.name))
        Character.num_characters += 1

    def attacked(self):
        self.life -= 10 # 공격을 받으면 life에서 10씩 감소함
        print('%s 이 공격을 받아 생명력이 %d 로 줄었습니다.' %(self.name, self.life))

    def attack(self, other):
        self.exp += 50 # 경험치 50 증가
        print('%s 가 %s 를 공격해서 경험치가 50 올랐습니다.' %(self.name, other.name))
        other.attacked()

    def __add__(self, other):
        print("%s 와 %s 가 파티를 맺었습니다." % (self.name, other.name))

    @classmethod
    def how_many_character(cls):
        print ("생성된 총 캐릭터 수:", cls.num_characters)

    def delete_character(self):
        print("{} 를 삭제합니다.".format(self.name))
        del self
        Character.num_characters -= 1
```

In [49]:

```
user1 = Character("한조")
user2 = Character("메르시")

user1.attack(user2)
```

```
이 캐릭터의 이름이 한조 로 정해졌습니다.
이 캐릭터의 이름이 메르시 로 정해졌습니다.
한조 가 메르시 를 공격해서 경험치가 50 올랐습니다.
메르시 이 공격을 받아 생명력이 90 로 줄었습니다.
```

5) Character 객체가 각각의 공격력을 가지도록 객체변수를 추가해 보세요. 또한 attack와 attacked 기능에 상대방의 공격력이 반영되서 체력이 깎이는 기능으로 코드를 수정 해 보세요

In [56]:

```
class Character:
    num_characters = 0 # 모든 캐릭터의 개체 수

    def __init__(self, name): # __init__의 input 인자는 name
        self.name = name #본 객체의 변수인 name에 input으로 받았던 이름을 넣어줌
        self.life = 100 # 모든 Character 객체는 life 가 100 을 가짐
        self.exp = 0 # 경험치는 0부터 시작
        self.power = 15
        self.__hidden_dps = 0.5 # 히든 dps
        print('이 캐릭터의 이름이 %s 로 정해졌습니다.' %(self.name))
        Character.num_characters += 1

    def attacked(self, other):
        self.life -= other.power # 공격을 받으면 life에서 10씩 감소함
        print('%s 이 공격을 받아 생명력이 %d 로 줄었습니다.' %(self.name, self.life))

    def attack(self, other):
        self.exp += 50 # 경험치 50 증가
        print('%s 가 %s 를 공격해서 경험치가 50 올랐습니다.' %(self.name, other.name))
        other.attacked(self)

    def __add__(self, other):
        print("%s 와 %s 가 파티를 맺었습니다." % (self.name, other.name))

    @classmethod
    def how_many_character(cls):
        print ("생성된 총 캐릭터 수:", cls.num_characters)

    def delete_character(self):
        print("{} 를 삭제합니다.".format(self.name))
        del self
        Character.num_characters -= 1
```

In [57]:

```
user1 = Character("한조")
user2 = Character("메르시")
user1.power = 30
user2.power = 15

user1.attack(user2)
user2.attack(user1)
user2.attack(user1)
```

```
이 캐릭터의 이름이 한조 로 정해졌습니다.
이 캐릭터의 이름이 메르시 로 정해졌습니다.
한조 가 메르시 를 공격해서 경험치가 50 올랐습니다.
메르시 이 공격을 받아 생명력이 70 로 줄었습니다.
메르시 가 한조 를 공격해서 경험치가 50 올랐습니다.
한조 이 공격을 받아 생명력이 85 로 줄었습니다.
메르시 가 한조 를 공격해서 경험치가 50 올랐습니다.
한조 이 공격을 받아 생명력이 70 로 줄었습니다.
```

6) 경험치가 100이되면 레벨이 올라가는 기능을 구현해 보세요

In [25]:

```
class Character:
    num_characters = 0 # 모든 캐릭터의 개체 수

    def __init__(self, name): # __init__의 input 인자는 name
        self.name = name #본 객체의 변수인 name에 input으로 받았던 이름을 넣어줌
        self.life = 100 # 모든 Character 객체는 life 가 100 을 가짐
        self.exp = 0 # 경험치는 0부터 시작
        self.power = 15
        self.level = 1
        self.__hidden_dps = 0.5 # 히든 dps
        print('이 캐릭터의 이름이 %s 로 정해졌습니다.' %(self.name))
        Character.num_characters += 1


    def attacked(self, other):
        self.life -= other.power # 공격을 받으면 life에서 10씩 감소함
        print('%s 이 공격을 받아 생명력이 %d 로 줄었습니다.' %(self.name, self.life))


    def attack(self, other):
        other.attacked(self)
        print('%s 가 %s 를 공격해서 경험치가 50 올랐습니다.' %(self.name, other.name))
        self.exp += 50 # 경험치 50 증가

        if (self.exp >= 100):
            self.level += 1
            self.exp -= 100 # self.exp = 0
            print('%s 가 레벨이 %s로 올랐습니다.' %(self.name, self.level))


    def __add__(self, other):
        print("%s 와 %s 가 파티를 맺었습니다." % (self.name, other.name))


    @classmethod
    def how_many_character(cls):
        print ("생성된 총 캐릭터 수:", cls.num_characters)


    def delete_character(self):
        print("{} 를 삭제합니다.".format(self.name))
        del self
        Character.num_characters -= 1
```

In [26]:

```
user1 = Character("한조")
user2 = Character("메르시")

user2.attack(user1)
user2.attack(user1)
```

```
이 캐릭터의 이름이 한조 로 정해졌습니다.
이 캐릭터의 이름이 메르시 로 정해졌습니다.
한조 이 공격을 받아 생명력이 85 로 줄었습니다.
메르시 가 한조 를 공격해서 경험치가 50 올랐습니다.
한조 이 공격을 받아 생명력이 70 로 줄었습니다.
메르시 가 한조 를 공격해서 경험치가 50 올랐습니다.
메르시 가 레벨이 2로 올랐습니다.
```

7) __add__() 함수를 오버라이딩 하여, 유저간의 파티를 맺는 기능을 추가 해 보세요. 또한 유저의 정보에서 현재의 파티원을 볼 수 있도록 해 보세요.

In [38]:

```
class Character:
    num_characters = 0 # 모든 캐릭터의 개체 수

    def __init__(self, name): # __init__의 input 인자는 name
        self.name = name #본 객체의 변수인 name에 input으로 받았던 이름을 넣어줌
        self.life = 100 # 모든 Character 객체는 life 가 100 을 가짐
        self.exp = 0 # 경험치는 0부터 시작
        self.power = 15
        self.level = 1
        self.party = []
        self.__hidden_dps = 0.5 # 히든 dps
        print('이 캐릭터의 이름이 %s 로 정해졌습니다.' %(self.name))
        Character.num_characters += 1


    def attacked(self, other):
        self.life -= other.power # 공격을 받으면 life에서 10씩 감소함
        print('%s 이 공격을 받아 생명력이 %d 로 줄었습니다.' %(self.name, self.life))


    def attack(self, other):
        other.attacked(self)
        print('%s 가 %s 를 공격해서 경험치가 50 올랐습니다.' %(self.name, other.name))

        self.exp += 50 # 경험치 50 증가
        if (self.exp >= 100):
            self.level += 1
            self.exp -= 100 # self.exp = 0
            print('%s 가 레벨이 %s로 올랐습니다.' %(self.name, self.level))


    def __add__(self, other):
        self.party.append(other)
        other.party.append(self)
        print("%s 와 %s 가 파티를 맺었습니다." % (self.name, other.name))


    @classmethod
    def how_many_character(cls):
        print ("생성된 총 캐릭터 수:", cls.num_characters)


    def delete_character(self):
        print("{} 를 삭제합니다.".format(self.name))
        del self
        Character.num_characters -= 1


    def __repr__(self):
        return self.name
```