# xpop.py

```python
import pop
def caller(func):
    def wrapper(*args, **kwargs):
        print(func.__name__, "(", args, ")", end=', ')
        ret = func(*args, **kwargs)
        print(ret)
        return ret
    return wrapper
```

@데코레이터

함수이름 (인자), 결과출력

---

```python
class Out(pop.Out):
    def __init__(self, n):
        print("create Out")
        super().__init__(n)
    def __del__(self):
        print("delete Out")
        super().__del__()

    @caller
    def on(self):
        super().on()

    @caller
    def off(self):
        print("call Out.off")
```

생성자 메서드

create Out

소멸자 메서드

delete Out

super().off()

call Out.off

---

```python
class Led(Out, pop.Led):
    def __init__(self, n):
        print("create Led")
        Out.__init__(self, n)

    def __del__(self):
        print("delete Led")
        super().__del__()

    @caller
    def blink(self, period, second):
        super().blink(period, second)
```

생성자 메서드

create Led

소멸자 메서드

delete Led

---

```python
class Leds(pop.Leds):
    def __init__(self, led=None, debug=False):
        print("create Leds")
        self._max = pop.pinMax(pop.LED)

        for i in range(self._max):
            pin = pop.pinMap(pop.LED, i)
            if pin is 0xFF:
                self._leds = [None]
                break
            else:
                if debug:
                    print("Led Pin: %d"% pin)
                self._leds.append(Led(pin))

    @caller
    def __getitem__(self, item):
        return super().__getitem__(item)

    @caller
    def allOn(self):
        super().allOn()

    @caller
    def allOff(self):
        super.allOff()

if __name__== "__main__":
    import time

    leds = Leds()
    leds.allOn()
    time.sleep(2)
    for i in range(8):
        leds[i].off()
        time.sleep(.5)
```

생성자 메서드

create Leds

전역함수

class 안에 있지 않은 함수

Led Pin : 핀

모두 켜고,

2초 후,

순서대로 0.5초 간격으로 꺼짐

---

## pop 라이브러리

```python
class Out(object):
    def __init__(self, n):
        self.bind = binder.Out_new(n)

    def __del__(self):
        self.off()

    def on(self):
        binder.Out_on(self.bind)

    def off(self):
        binder.Out_off(self.bind)
```

---

```python
class Led(Out):
    def __init__(self, n):
        super().__init__(n)
    def __del__(self):
        super().__del__()

    def blink(self, period, second):
        for _ in range(second):
            self.on()
            delay(period)
            self.off()
            delay(period)
```

---

```python
class Leds(object):
    _leds = list()

    def __init__(self, debug=False):
        self._max = pinMax(LED)

        for i in range(self._max):
            pin = pinMap(LED, i)
            if pin is 0xFF:
                _leds = [None]   # 깊은복사
                break
            else:
                if debug:
                    print("Led Pin: %d"% pin)
                self._leds.append(Led(pin)) # 얕은복사

    def __getitem__(self, item):
        if(len(self._leds) == 0):
            raise ValueError("leds are empty")
        return self._leds[item]

    def allOn(self):
        for led in self._leds:
            led.on()

    def allOff(self):
        for led in self._leds:
            led.off()
```