```
1 Lab. Python's class basic II
 3 1. 사용 tool
      -Jupyter Notebook
 4
 5
      -Microsoft Visual Studio Code
 6
 7 2. Code
 8
 9
      class Point:
10
        def __init__(self, x, y):
11
        self.x = x
12
        self.y = y
13
14
        def print_pt(self):
15
        print('({}, {})'.format(self.x, self.y))
16
17
        def add(self, pt):
18
        new_x = self.x + pt.x
19
        new_y = self.y + pt.y
20
        return Point(new_x, new_y)
21
22
        def multiply(self, factor):
23
        return Point(self.x * factor, self.y * factor)
24
25
        def length(self):
        return self.x ** 2 + self.y ** 2
26
27
28
        def get_x(self):
29
        return self.x
30
31
        def get_y(self):
32
        return self.y
33
34
         #Base Overloading Methods
35
         def __str__(self):
        return '({}, {})'.format(self.x, self.y)
36
37
38
        def __add__(self, pt):
39
        new_x = self.x + pt.x
40
        new y = self.y + pt.y
41
        return Point(new_x, new_y)
42
        def __sub__(self, pt):
43
44
        new_x = self.x - pt.x
45
        new_y = self.y - pt.y
46
        return Point(new_x, new_y)
47
48
        def mul (self, factor):
49
        return Point(self.x * factor, self.y * factor)
50
        def __len__(self):
51
```

```
return self.x ** 2 + self.y ** 2
52
53
54
        def __getitem__(self, index):
55
       if index == 0: return self.x
56
       elif index == 1 : return self.y
57
58
59
     p1 = Point(100, 200)
60
     p2 = Point(300, 450)
61
62
     p1.print pt() #(100, 200)
     p2.print_pt() #(300, 450)
63
64
65
     print(p1) #(100, 200) call __str__()
66
     print(p2) #(300, 450)
67
68
     p3 = p1.add(p2)
69
     print(p3) #(400, 650)
70
71
     p4 = p1 + p2 \#call \_add_()
72
     print(p4) #(400, 650)
73
74
     p5 = p2 - p1 \#call \__sub__()
75
     print(p5)
                #(200, 250)
76
77
     p6 = p1.multiply(7)
78
     print(p6)
                  #(700, 1400)
79
80
     p7 = p1 * 7
81
     print(p7)
                  #(700, 1400)
82
83
84
     print('p1\'s length =', p1.length()) #p1's length = 50000
     print('p1\'s length = ', len(p1)) #call __len__() p1's length = 50000
85
86
87
     print('p1(x, y) = ({}, {})'.format(p1.get_x(), p1.get_y())) #p1(x, y) = (100, 200)
88
     print('p1(x, y) = ({}, {})'.format(p1[0], p1[1])) #p1(x, y) = (100, 200)
89
90
91
     Refer to 3.3.8. Emulating numeric
     types(https://docs.python.org/3/reference/datamodel.html?object.__add__#emulating-num
     eric-types)
```