Lab. Python's class basic I

1. 사용 tool
   -Jupyter Notebook
   -Microsoft Visual Studio Code

2. Code

```
#Creating Classes
class Employee:
    """Common base class for all employees"""
    empCount = 0

    def __init__(self, name, salary):
   self.name = name
   self.salary = salary
   Employee.empCount += 1

    def displayCount(self):
   print("Total Employee %d" % Employee.empCount)

    def displayEmployee(self):
   print("Name : ", self.name,  ", Salary: ", self.salary)


#Creating Instance Objects
emp1 = Employee("Zara", 2000)
emp2 = Employee("Manni", 5000)

#Accessing Attributes
emp1.displayEmployee()
emp2.displayEmployee()
print("Total Employee %d" % Employee.empCount)
# Name :  Zara ,Salary:  2000
# Name :  Manni ,Salary:  5000
# Total Employee 2


emp1.age = 7  # Add an 'age' attribute.
emp1.age = 8  # Modify 'age' attribute.
#del emp1.age  # Delete 'age' attribute.

print(hasattr(emp1, 'age'))    # Returns true if 'age' attribute exists
print(getattr(emp1, 'age'))    # Returns value of 'age' attribute
print(setattr(emp1, 'age', 8)) # Set attribute 'age' at 8
#print(delattr(empl, 'age'))    # Delete attribute 'age'


# Built-In Class Attributes
print("Employee.__doc__:", Employee.__doc__)
print("Employee.__name__:", Employee.__name__)
print("Employee.__module__:", Employee.__module__)
print("Employee.__bases__:", Employee.__bases__)
print("Employee.__dict__:", Employee.__dict__)
# Employee.__doc__: Common base class for all employees
# Employee.__name__: Employee
# Employee.__module__: __main__
# Employee.__bases__: ()
# Employee.__dict__: {'__module__': '__main__', 'displayCount':...


# Destroying Objects (Garbage Collection)
class Point:
    def __init__( self, x=0, y=0):
   self.x = x
   self.y = y

    def __del__(self):
   class_name = self.__class__.__name__
   print(class_name, "is destroyed")

pt1 = Point()
pt2 = pt1
pt3 = pt1
print(id(pt1), id(pt2), id(pt3)) # prints the ids of the obejcts

del pt1
del pt2
del pt3
# 3083401324 3083401324 3083401324
# Point is destroyed


# Class Inheritance
class Parent:        # define parent class
    parentAttr = 100

    def __init__(self):
   print("Calling parent constructor")

    def parentMethod(self):
   print('Calling parent method')

    def setAttr(self, attr):
   Parent.parentAttr = attr

    def getAttr(self):
   print("Parent attribute :", Parent.parentAttr)


class Child(Parent): # define child class
    def __init__(self):
   print("Calling child constructor")
```

```

    def childMethod(self):
   print('Calling child method')

c = Child()          # instance of child
c.childMethod()      # child calls its method
c.parentMethod()     # calls parent's method
c.setAttr(200)       # again call parent's method
c.getAttr()          # again call parent's method
# Calling child constructor
# Calling child method
# Calling parent method
# Parent attribute : 200


# Overriding Methods
class Bumo:          # define parent class
    def myMethod(self):
   print('Calling parent method')

class Jasik(Bumo): # define child class
    def myMethod(self):
   print('Calling child method')

jasik = Jasik()          # instance of child
jasik.myMethod()         # child calls overridden method
# Calling child method


#Data Hiding
class JustCounter:
    __secretCount = 0

    def count(self):
   self.__secretCount += 1
   print(self.__secretCount)

counter = JustCounter()
counter.count()
counter.count()
#print(counter.__secretCount)   error 발생
# 1
# 2
# Traceback (most recent call last):...

print(counter._JustCounter__secretCount)
# 1
# 2
# 2
```