Class and Package

```
import warnings
warnings.filterwarnings('ignore')
```

I. Class

- Class(Name Space): 개발자에 의해 지정된 독립된 메모리 공간(구조)
  - 함수와 변수를 하나의 객체(Object)로 묶어서 관리
  - 선언된 Class 사용을 위해 Instance 생성 필요
- Class Method: Class 내에 선언된 함수
- Class Member: Class 내에 선언된 변수
  - Instance Member: Class 내 함수에 선언된 변수

1) Class 선언

- self: 첫 번째 매개변수 self는 Class로 생성한 Instance 자체를 지정
  - Class로 생성된 Object(객체) 자신을 참조하는 매개변수
  - self를 사용하여 Class Member에 접근

```
class myClass:
    var_1 = "'Hello Class >>> Class Member'"

    def func_1(self):
        var_2 = "'Method >>> Instance Member'"
        print(self.var_1, 'AND', var_2)
```

2) Instance 생성

- Instance: Class로 생성된 객체(Object)
  - 'obj'는 'myClass( )'의 Instance

```
obj = myClass()
```

3) Class Member 호출

```
obj.var_1
```

```
''Hello Class >>> Class Member''
```

4) Instance Member 호출

- Error

```
obj.var_2
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-5-c39b47aafef0> in <module>
----> 1 obj.var_2

AttributeError: 'myClass' object has no attribute 'var_2'

SEARCH STACK OVERFLOW
```

5) Method 호출

- Class Member AND Instance Menber 호출 성공

```
obj.func_1()
```

```
'Hello Class >>> Class Member' AND 'Method >>> Instance Member'
```

II. Calculator Class

1) 'Calculator' Class 선언

```
class Calculator:
    def inputData(self, m, n):
        self.m = m
        self.n = n
    def addtion(self):
        result = self.m + self.n
        return result
```

2) Instance 생성

```
cal_1 = Calculator()
```

```
cal_2 = Calculator()
```

3) inputData Method

```
cal_1.inputData(3, 6)
```

```
cal_2.inputData(9, 5)
```

4) addition Method

```
cal_1.addtion()
```

```
9
```

```
cal_2.addtion()
```

```
14
```

III. Class 생성자 (Constructor)

```
cal_3 = Calculator()
```

- Error

```
cal_3.addtion()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-15-90e9d8d80b74> in <module>
----> 1 cal_3.addtion()

<ipython-input-7-f0f0ae9675f9> in addtion(self)
      4         self.n = n
      5     def addtion(self):
----> 6         result = self.m + self.n
      7         return result

AttributeError: 'Calculator' object has no attribute 'm'

SEARCH STACK OVERFLOW
```

1) 'Calculator_2' Class 선언

- __init__( ) 예약어

```
class Calculator_2:
    def __init__(self, m, n):
        self.m = m
        self.n = n
        print('메모리에 인스턴스가 %d, %d 값과 함께 생성되었습니다.' % (m, n))
    def inputData_2(self, m, n):
        self.m = m
        self.n = n
    def subtraction(self):
        result = self.m - self.n
        return result
```

2) 인스턴스 생성 메시지 출력

- __init__( ) Method 실행

```
cal_4 = Calculator_2(4, 8)
```

```
메모리에 인스턴스가 4, 8 값과 함께 생성되었습니다.
```

3) 생성된 초기값 사용

- subtraction Method

```
cal_4.subtraction()
```

```
-4
```

4) 변수값 재지정

```
cal_4.inputData_2(4, 2)
```

```
cal_4.subtraction()
```

```
2
```

IV. Class 상속(Inheritance)

1) 'myComputer' Class 선언

- Class Calculator, Calculator_2에 선언된 Method를 상속받아 사용 가능
- Class myComputer 내에 추가 Method를 선언하여 사용

```
class myComputer(Calculator, Calculator_2):
    def multiplication(self):
        result = self.m * self.n
        return result
    def division(self):
        result = self.m / self.n
        return result
```

2) Instance 생성

```
com_1 = myComputer(8, 2)
```

```
메모리에 인스턴스가 8, 2 값과 함께 생성되었습니다.
```

3) Method 사용

- Calculator Class로 부터 상속된 addition( ) Method

```
com_1.addtion()
```

```
10
```

- Calculator_2 Class로 부터 상속된 subtraction( ) Method

```
com_1.subtraction()
```

```
6
```

- myComputer Class에 선언된 multiplication( ) Method

```
com_1.multiplication()
```

```
16
```

- myComputer Class에 선언된 division( ) Method

```
com_1.division()
```

```
4.0
```

- Calculator Class로 부터 상속된 inputData( ) Method

```
com_1.inputData(9, 3)
```

```
com_1.addtion()
```

```
12
```

V. Package

- 관련된 모듈을 디렉토리 구조를 사용하여 계층적으로 관리
  - Package_Name.Module_name.Function_Name( )
  - __init__.py 파일을 사용하여 해당 디렉토리가 Package에 사용됨을 알려줌
- import 또는 from ~ import 구문으로 호출하여 사용

1) 사용자 패키지 만들기

- Colab에 'myPackage' 디렉토리 생성
- 'myPackage' 디렉토리 내에 __init__.py 생성
- __init__.py에 version=1.0 작성 후 UTF-8 Encoding으로 저장
- myLibrary.py 파일(Module) 생성(UTF-8 Encoding)
- myLibrary.py 내에 다양한 Fuction 및 Class 정의
- myLibrary.py 파일을 'myPackage' 디렉토리로 이동

2) 사용자 패키지 with import

- 사용구문: import Package_Name.Module_Name

```
import myPackage.myLibrary
```

- 사용구문: Package_Name.Module_Name.Function_Name

```
myPackage.myLibrary.hi()
```

```
Hello Package
```

- 사용구문: Object_Name = Package_Name.Module_Name.Class_Name()

```
ins_1 = myPackage.myLibrary.myClass()
```

```
com_2 = myPackage.myLibrary.myComputer(5, 9)
```

```
메모리에 인스턴스가 5, 9 값과 함께 생성되었습니다.
```

- 사용구문: Object_Name.Method_Name( )

```
ins_1.func_1()
```

```
Hello Class
```

```
com_2.multiplication()
```

```
45
```

3) 사용자 패키지 with Alias (import ~ as)

- 사용구문: import Package_Name.Module_Name as Alias

```
import myPackage.myLibrary as mp
```

- 사용구문 : Alias.Function_Name( )

```
mp.hi()
```

```
    Hello Package
```

- 사용구문: Object_Name = Alias.Class_Name()

```
ins_2 = mp.myClass()
```

- 사용구문: Object_Name.Method_Name( )

```
ins_2.func_1()
```

```
    Hello Class
```

4) 사용자 패키지 with from ~ import

- 사용구문: from Package_Name.Module_Name import Function_Name

```
from myPackage.myLibrary import hi
```

- 사용구문: Function_Name( )

```
hi()
```

```
    Hello Package
```

- 사용구문: from Package_Name.Module_Name import Class_Name

```
from myPackage.myLibrary import myComputer
```

- 사용구문: Class_Name( )

```
com_3 = myComputer(3, 6)
```

```
    메모리에 인스턴스가 3, 6 값과 함께 생성되었습니다.
```

```
com_3.addtion()
```

```
    9
```

#

#

#

The End

#

#

#