시작하기전에

<wikipedia - 객체지향 프로그래밍> >

> 객체 지향 프로그래밍(영어: Object-Oriented Programming, OOP)은 컴퓨터 프 로그래밍의 패러다임의 하나이다. 객체 지향 프로그래밍은 컴퓨터 프로그램을 명령어의 목록으로 보는 시각에서 벗어나 여러 개의 독립된 단위, 즉 "객체"들의 모임으로 파악하고자 하는 것이다. 각각의 객체는 메시지를 주고받고, 데이터를 처리할 수 있다.
>
> 명령형 프로그래밍인 절차지향 프로그래밍에서 발전된 형태를 나타내며, 기본 구성요소는 다음과 같다.

- 클래스(Class) - 같은 종류(또는 문제 해결을 위한)의 집단에 속하는 속성(attribute)과 행위(behavior)를 정의한 것으로 객체지향 프로그램의 기본적인 사용자 정의 데이터형(user define data type)이라고 할 수 있다. 클래스는 프로그래머가 아니지만 해결해야 할 문제가 속하는 영역에 종사하는 사람이라면 사용할 수 있고, 다른 클래스 또는 외부 요소와 독립적으로 디자인하여야 한다.
- 인스턴스 - 클래스의 인스턴스(실제로 메모리상에 할당된 것)이다. 객체는 자신 고유의 속성(attribute)을 가지며 클래스에서 정의한 행위(behavior)를 수행할 수 있다. 객체의 행위는 클래스에 정의된 행위에 대한 정의를 공유함으로써 메모리를 경제적으로 사용한다.
- 메서드(Method) - 클래스로부터 생성된 객체를 사용하는 방법으로서 객체에 명령을 내리는 것이라 할 수 있다. 메서드는 한 객체의 속성을 조작하는 데 사용된다.

In [ ]:

```
# 복소수를 하나 만들어보고, 타입을 출력해봅시다.
a = 3+4j
print(type(a))
```

In [ ]:

```
# 허수부랑 실수부를 함께 출력해봅시다.
print(a.real, a.imag)
```

In [ ]:

```
# 리스트를 하나 만들고 정렬해봅시다.
L = [5, 1, 3]
L.sort()    # sort는 list의 Method, sorted는 Built-in
print(L)
print(type(L))
```

In [90]:

```
# 리스트가 할 수 있는 것들을 알아봅시다.
print(dir(list))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__',
'__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remov
e', 'reverse', 'sort']
```

실습 문제

> 프로그래밍으로 나와 친구의 이름을 저장해보세요.

각자의 명함과 지갑을 만들어봅시다.

- 내 생일, 전화번호, 이메일주소 정보를 담은 변수를 확인해봅시다.
- 주머니(pocket)에는 돈(won)을 포함하여 현재 가지고 있는 것을 작성해보세요.
- 나는 인사를 하면서 내 명함에 있는 정보 하나를 이야기합니다. `greeting` 함수를 만듭시다.

- 나는 주머니에 원하는 것과 갯수를 지정하여 넣을 수 있습니다.

  기존에 값이 있으면, 갯수를 추가하고 없으면 새로 만드는 `in_my_pocket` 함수를 만듭시다.

친구의 정보와 지갑도 만들어봅시다.

In [ ]:

```
# 아래에 자유롭게 코드를 작성해보세요.
my_name = 'kim'
you_name = 'park'
my_birthday = '1234'
my_phone = '01012345678'

my_pocket = {'won' : 500, 'phone' : 1, 'candy' : 5}
your_pocket = {'won' : 10000, 'dollar' : 1}

def in_my_pocket(pocket, stuff, value):
    if stuff in pocket:
        pocket[stuff] += value
    else:
        pocket[stuff] = value
    return pocket


def greeting(name, phone):
    print(f'안녕, 나는 {name}야 내 번호는 {phone}야')
```

In [ ]:

```
# in_my_pocket 함수를 통해 내 주머니에 내용을 추가해봅시다.
def in_my_pocket(pocket, stuff, count):
    if pocket.get(stuff):
        pocket[stuff] += count
    else:
        pocket[stuff] = count
    return pocket

in_my_pocket(my_pocket, 'dollar', 10)
```

In [ ]:

```
# greeting 함수를 통해 인사를 해봅시다.
greeting(my_name, my_phone)
```

클래스 및 인스턴스

클래스 객체

```
class ClassName:
```

- 선언과 동시에 클래스 객체가 생성됨.
- 또한, 선언된 공간은 지역 스코프로 사용된다.
- 정의된 어트리뷰트 중 변수는 멤버 변수로 불리운다.
- 정의된 함수(`def`)는 메서드로 불리운다.

In [ ]:

```
class Person:
    name = '고길동'
    def sayhello(self):
        print(f"Hello, I'm {self.name}.")
```

- 선언시 self는 반드시 작성해주세요! 나중에 설명드립니다.

인스턴스 객체

- 인스턴스 객체는 `ClassName()`을 호출함으로써 선언된다.
- 인스턴스 객체와 클래스 객체는 서로 다른 이름 공간을 가지고 있다.
- 인스턴스 -> 클래스 -> 전역 순으로 탐색을 한다.

In [ ]:

```
# iu라는 클래스 Person의 인스턴스를 만들어봅시다.
iu = Person()
```

In [ ]:

```
# 인사하는 메서드를 호출해봅시다.
iu.sayhello()
```

In [ ]:

```
# iu의 이름을 확인해봅시다.
iu.name
```

In [ ]:

```
# iu로 이름을 바꿔주세요.
iu.name = 'iu'
iu.name
```

In [ ]:

```
# iu가 인사를 합니다.
iu.sayhello()
```

In [ ]:

```
# iu와 Person이 같은지 확인해보겠습니다.
isinstance(iu, Person)
```

In [ ]:

```
# iu를 출력해봅시다.
iu
```

In [ ]:

```
# iu를 출력해봅시다 2.
print(iu)
```

In [ ]:

```
# map을 만들어 비교해 봅시다.
print(map(int, '123'))
```

In [ ]:

```
a = (1, 2)
a
```

In [ ]:

```
b = dict(x=11, y=22)
b
```

In [ ]:

```
c = list(range(10))
```

```
c.append(100)
c
```

In [ ]:

```
print(type(a))
print(type(b))
print(type(c))
```

In [ ]:

```
class Person:
    name = '고길동'
    def sayhello():     #self : ()안에 인자가 없어서 self로..
        print(f"Hello, I'm {name}.")
```

In [ ]:

```
iu = Person()
iu.sayhello()
Person.sayhello(iu)
```

실습 문제 발전

> 지금까지 배운 것을 통해서 Person 클래스를 만들고, 친구와 나를 표현해봅시다.
>
> 주머니와 정보를 가지고 있고 (멤버 변수)
>
> 인사(`greeting()`)와 주머니에 내용을 추가(`in_my_pocket()`)할 수 있습니다. (메서드)
>
> 추가적으로 `get_my_pocket()` 으로 지갑에 담긴 정보를 가져와 봅시다.

In [83]:

```
# 아래에 코드를 작성해주세요.
class Person:
    name = 'kim'
    phone = '01012345678'
    pocket = {'won' : 500, 'phone' : 1, 'candy' : 5}

    def in_my_pocket(self, stuff, count):
        if self.pocket.get(stuff):
            self.pocket[stuff] += count
        else:
            self.pocket[stuff] = count
        return self.pocket

    def greeting(self):
        print(f'안녕, 나는 {self.name}야 내 번호는 {self.phone}야')

    def get_my_pocket(self):
        return self.pocket
```

In [84]:

```
kim = Person()
kim.name = '고길동'
kim.name
kim.phone
```

Out[84]:

```
'01012345678'
```

In [86]:

```
kim.greeting()
```

안녕, 나는 고길동야 내 번호는 010123456678야

In [87]:

```
kim.get_my_pocket()
```

Out[87]:

```
{'won': 500, 'phone': 1, 'candy': 5}
```

In [89]:

```
kim.in_my_pocket('dollar', 5)
```

Out[89]:

```
{'won': 500, 'phone': 1, 'candy': 5, 'dollar': 5}
```

In [57]:

```
# 아래에 코드를 작성해주세요.
class Person:
    name = 'kim'
    phone = '01012345678'
    pocket = {'won' : 500, 'phone' : 1, 'candy' : 5}


    def in_my_pocket(self, stuff, count):
#         self.stuff = stuff
#         self.count = count
        if self.pocket.get(stuff):
            self.pocket[stuff] += count
        else:
            self.pocket[stuff] = count
        return self.pocket

    def greeting(self):
        print(f'안녕, 나는 {self.name}야 내 번호는 {self.phone}야')

    def get_my_pocket(self):
        return self.pocket
```

In [93]:

```
# 나를 만들어 봅시다.
me = Person()
```

In [91]:

```
# greeting, in_my_pocket, get_my_pocket
me.greeting()
me.in_my_pocket('dollar', 10)
me.get_my_pocket()
```

안녕, 나는 kim야 내 번호는 010123456678야

Out[91]:

```
{'won': 500, 'phone': 1, 'candy': 5, 'dollar': 30}
```

In [95]:

```
class Person:
    def __init__(self, name, age=0):
        self.name = name
        self.age = age
    def greeting(self):
        print(f'안녕하세요. {self.name}. {self.age}살 입니다')
```

In [97]:

```
p1 = Person('홍길동', 20)
p1.greeting()
```

안녕하세요. 홍길동. 20살 입니다

In [98]:

```
p2 = Person('둘리')
p2.greeting()
```

안녕하세요. 둘리. 0살 입니다