Static method

In [ ]:
```
import datetime
```

In [ ]:
```
now = datetime.datetime.now()
```

In [ ]:
```
print(now, type(now))
```

In [ ]:
```
KST = datetime.timezone(datetime.timedelta(hours=9))
```

In [ ]:
```
KST
```

In [ ]:
```
oop_day = datetime.datetime(2021, 1, 27, tzinfo=KST)
```

In [ ]:
```
oop_day
```

In [ ]:
```
oop_day.tzname()
```

class method?

In [1]:
```
class Person:
    population = 0

    def __init__(self):
        Person.population += 1

    @classmethod
    def get_population(cls):
        # 얘는 자식 클래스가 아무리 호출해도 Person의 인구
        print(Person.population)

class Student(Person):
    population = 0

    def __init__(self):
        super().__init__()
        Student.population += 1

# p1 = Person()
# s1 = Student()
# Person.get_population() # 결과 2
# Student.get_population() # 결과 2
```

In [ ]:
```
class Person:
    population = 0

    def __init__(self):
```

```
        Person.population += 1

    @classmethod
    def get_population(cls):
        # cls는 호출시 파이썬이 클래스를 넘겨줌
        # 그래서 Student.get_population()은 Student.population을 출력할 것
        # Person.get_population()은 Person.population을 출력할 것!
        print(cls.population)

class Student(Person):
    population = 0

    def __init__(self):
        super().__init__()
        Student.population += 1

# p1 = Person()
# s1 = Student()
# Person.get_population() # 결과 2
# Student.get_population() # 결과 1 <<<<<< 다르다!
```