클래스 네임스페이스

네임스페이스라는 것은 변수가 객체를 바인딩할 때 그 둘 사이의 관계를 저장하고 있는 공간을 의미.
예를 들어, 'a = 2'라고 했을 때 a 라는 변수가 2 라는 객체가 저장된 주소를 가지고 있는데 그러한 연결 관계가 저장된 공간이 바로 네임스페이스이다.
파이썬에서는 클래스가 정의되면 하나의 독립적인 네임스페이스가 생성된다. 그리고 클래스 내에 정의된 변수나 메서드는 그 네임스페이스 안에 파이썬 딕셔너리 타입으로 저장.

```
>>> class Stock:
    market = "kospi"

>>> dir()
['Stock', '__builtins__', '__doc__', '__loader__', '__name__', '__package__

>>> print("Stock.market => ", Stock.market)
Stock.market =>  kospi
```

생성된 s1, s2 인스턴스가 네임스페이스에 있는지 코드를 통해 확인.
dir() 내장함수의 반환값을 확인하면 s1, s2가 Stock과 마찬가지로 존재하는것을 확인할 수 있다.

```
>>> print("s1 = Stock() => ", id(s1))
s1 = Stock() =>  58237688
>>> print("s2 = Stock() => ", id(s2))
s2 = Stock() =>  58516432
```

s1과 s2 인스턴스의 네임스페이스는 현재 비어있는것을 확인할 수 있다.

```
>>> print("s1.__dict__ => ", s1.__dict__)
s1.__dict__ =>  {}
>>> print("s2.__dict__ => ", s2.__dict__)
s2.__dict__ =>  {}
```

s1 인스턴스에 market이라는 변수 추가
다시 확인해보면 kosdaq라는 키:값 쌍이 추가

```
>>> s1.market = 'kosdaq'
>>> print("s1.__dict__ => ", s1.__dict__)
s1.__dict__ =>  {'market': 'kosdaq'}
>>> print("s2.__dict__ => ", s2.__dict__)
s2.__dict__ =>  {}
```

만약 s1.market, s2.market과 같이 인스턴스를 통해 market이라는 값에 접근하면?

```
>>> print("s1.market => ", s1.market)
s1.market =>  kosdaq
>>> print("s2.market => ", s2.market)
s2.market =>  kospi

s2 인스턴스를 통해 변수에 접근하면 파이썬은 먼저 s2 인스턴스의 네임스페이스에서
해당 변수가 존재하는지 찾습니다.
즉, s2.market 이라는 문장이 실행되면 Stock 클래스의 네임스페이스에 있는
'market':'kospi' 키:값 쌍에서 'kospi'라는 문자열을 출력하게 됩니다.
```

클래스 변수와 인스턴스 변수

```
Account 클래스
생성자 (__init__) : 클래스의 인스턴스가 생성될 때 자동으로 호출되는 함수
소멸자 (__del__) : 클래스의 인스턴스가 소멸될 때 자동으로 호출되는 함수

class Account:
num_accounts = 0
def __init__(self, name):
    self.name = name
    Account.num_accounts += 1

def __del__(self):
    Account.nrm_accounts -= 1
```

여러 인스턴스 간에 서로 공유해야 하는 값은 클래스 변수를 통해 바인딩

```
>>> kim = Account("kim")
>>> lee = Account("lee")
>>> print("kim.name => ",kim.name)
kim.name =>  kim
>>> print("lee.name => ", lee.name)
lee.name =>  lee
```

먼저 인스턴스의 네임스페이스에서 num_accounts를 찾았지만 해당 이름이 없어서 클래스의 네임스페이스로 이동한 후, 다시 해당이름을 찾았고 그 값이 반환된 것

```
>>> print("kim.num_accounts => ",kim.num_accounts)
kim.num_accounts =>  2
>>> print("lee.num_accounts => ",lee.num_accounts)
lee.num_accounts =>  2
```

클래스 상속

자식이 부모님으로부터 재산 등을 상속받는 것처럼 다른 클래스에 이미 구현된 메서드나 속성을 상속한 클래스에서는 그러한 메서드나 속성을 그대로 사용할 수 있게 된다.
클래스의 상속을 또 다른 관점에서 생각해보면 클래스를 상속한다는 것은 부모 클래스의 능력을 그대로 전달받는 것을 의미한다. 인간으로 치면 부모로부터 유전형질을 물려받아 부모의 능력을 그대로 물려받는 것과 비슷함.

```
class Parent:
def can_sing(self):
    print("Sing a song")

>>> father = Parent()
>>> father.can_sing()
Sing a song
```

```
class Parent:
def can_sing(self):
    print("Sing a song")

class LuckyChild(Parent):
```

```
pass

>>> child1 = LuckyChild()
>>> child1.can_sing()
Sing a song
```

```
class Parent:
def can_sing(self):
    print("Sing a song")

class LuckyChild(Parent):
pass

class LuckyChild2(Parent):
def can_dance(self):
    print("Shuffle Dance")
```