مثالی در شی‌گرایی

چون شی‌گرایی و مفاهیم آن از حوصله‌ی این فصل و به طور کل این دوره، خارج است، در این درسنامه فقط به مثالی از مفاهیم پایه‌ی شی‌گرایی در پایتون می‌پردازیم.

Copy Python

```python
class Human:

    species = "H. sapiens"

    def __init__(self, name):
        # Assign the argument to the instance's name attribute
        self.name = name

        # Initialize property
        self._age = 0

    def say(self, msg):
        print("{name}: {message}".format(name=self.name, message=msg))


    def sing(self):
        return 'yo... yo... microphone check... one two... one two...'

    @classmethod
    def get_species(cls):
        return cls.species

    @staticmethod
    def grunt():
        return "*grunt*"


    @property
    def age(self):
        return self._age


    @age.setter
    def age(self, age):
        self._age = age


    @age.deleter
    def age(self):
        del self._age
```

ویژگی کلاس (*class attribute*):

این متغیر متعلق به کلاس است، نه یک نمونه (*instance*) خاص از کلاس و بین تمام نمونه‌های این کلاس مشترک است.

```
>>> first = Human("first")
>>> second = Human("second")
>>> Human.species ="something"
>>> print(second.species)
something
>>> print(first.species)
something
```

در کد بالا می‌بینید که با تغییر این ویژگی از طریق کلاس، این ویژگی برای همه تغییر کرده اما اگر از طریق یک شی عوض می‌شد، فقط برای آن شی تغییر می‌کرد.

- **نکته**: دقت کنید توابع و اشیایی که قبل و بعد از نام آنها دو زیر خط می‌آید (به صورت `__name__`) توسط پایتون استفاده می‌شوند اما در فضای تحت کنترل کاربر هستند. برای مثال `__init__` , `__str__` , `__repr__` از جمله‌ی آنها هستند. سعی کنید در نامگذاری‌های خود از چنین الگویی استفاده نکنید.

تابع سازنده (*initializer*) :

وقتی نمونه‌ای از کلاس ساخته می‌شود، متد `__init__` فراخوانی می‌شود. می‌تواند آرگومان‌های دیگری غیر از `self` هم داشته باشد و معمولا در آن ویژگی‌های کلاس را مقداردهی می‌کنند یا کارهایی برای آماده‌سازی آن نمونه انجام می‌دهند.

```
>>> i = Human(name="Ian")
>>> i.say("hi")
Ian: hi
>>> j = Human("Joel")
>>> j.say("hello")
Joel: hello
```

متد متعلق به نمونه (*instance method*):

به متدهایی گفته می‌شود که `self` را به عنوان آرگومان اول خود می‌گیرند و به این شی از کلاس، از این طریق دسترسی دارند.

متد متعلق به کلاس (*class method*):

بین تمامی نمونه‌های کلاس مشترک است و می‌تواند از طریق نمونه یا خود کلاس فراخوانی شود. کلاس را به عنوان آرگومان اول خود می‌گیرد. (این ویژگی در جایی به درد می‌خورد که از اینکه این متد توسط کدام کلاس در توالی وراثت فراخوانی شده، اطلاعی نداریم) و حتما باید قبل از این متد از `@classmethod` استفاده شود.

```
>>> i = Human(name="Ian")
>>> j = Human("Joel")
>>> i.say(i.get_species())
Ian: H. sapiens
>>> Human.species = "H. neanderthalensis"
>>> i.say(i.get_species())
Ian: H. neanderthalensis
```

```
>>> j.say(j.get_species())
Joel: H. neanderthalensis
```

متد استاتیک (*static method*) :

فقط می‌تواند توسط کلاس فراخوانی شود نه نمونه‌های کلاس. `self` را به عنوان آرگومان ندارد. بیشتر به عنوان ابزار توابع دیگر از آن استفاده می‌شود. باید حتما قبل از آن از `@staticmethod` استفاده شود.

```
>>> print(Human.grunt())
*grunt*
```

متد *setter*

به عنوان `setter` برای یک متغیر عمل می‌کند. قبل از آن باید از `@varible_name.setter` استفاده شود.

```
>>> i.age = 42
```

متد *property*:

کاملا مشابه `getter` برای یک متغیر عمل می‌کند. باید قبل از آن از `@property` استفاده شود.

```
>>> i.say(i.age)
Ian: 42
```

متد *deleter*

مثل دوتای بالایی هنگام پاک کردن متغیر استفاده می‌شود. قبل از آن باید از `@varible_name.deleter` استفاده شود.

```
>>> del i.age
```