OOP Kuralları

Nesne Nedir?

Nesne ait olduğu **sınıftan gelen özelliklerin değerlerinin belli olduğu, sınıfı için tanımlı olan davranışları nasıl sergilediğinin bilindiği, somut olarak var olan, biricik bir kimliği olan** varlıktır. Örnek olarak bir Araba düşünürsek markası, modeli, rengi vs. belli olan bir araç olmalı.

Sınıf Nedir?

Sınıf, **aynı özellik ve davranışları sergileyen varlıkların ortak kümesini belirleyen tanımdır**. Örnek olarak hiç otomobil görmemiş birisine otomobilin ne olduğunu anlatmaya kalktığımız zaman kişinin bir otomobil gördüğünde tanımasını sağlayacak şekilde anlatmalıyız. Tekerlek, kapı, motor, far vs.

Nesneye Yönelik Programlama ve Yaklaşımı Nedir?

Nesneye yönelik programlama, **nesnelerin birbirlerine ileti göndermeleri ilkesine dayanır.** Bir nesne, başka bir nesneye bir ileti(message) göndererek, o nesneden bir davranış(behaviour) sergilemesini ister**. İletiyi alan nesne, bu iletiye göre davranışını gerçekleştirir ve kendi durum(state) bilgisini değiştirir**. Yaklaşımına gelirsek, gerçek hayattan alınmış problemi çözmek üzere oluşturulacak modelin, gene gerçek hayatta var olan nesneler ve bu nesneler arasındaki ilişkilerden faydalanılarak oluşturulmasını ilke edinmiştir. Problem aynen gerçek hayatta görüldüğü şekli ile sanal ortama aktarılabilirse, **günlük yaşamında nesneler ve nesneler arası ilişkiler ile etkileşimde olan programcı, nesneye yönelik model sayesinde, üzerinde çalıştığı problemi aynen gerçek hayatta olduğu şekliyle görebilecektir.**

Constructor Nedir?

Sınıf adı ile aynı adı taşır, dışarıdan parametre alabilirler, geriye değer döndürmezler, overload yapılabilirler, birden fazla constructor olabilir. Nesneyi her ürettiğimiz yerde tekrar bu marka değişkenine değer atamak zorunda kalmalıyız. İstisnai durumlarda ise overload yapılabilsin diye, ikinci constructor parametre alarak marka değişkenine değer atanması sağlanır. Kod yazarken bu değerlere tiplerine ilk değerlerini vermeden kullanmak istediğimizde hata alınır. Oluşturduğumuz nesnenin, bellekten silinmeden

önceki gerçekleştireceği son işlemi gerçekleştiren metottur. Örnek olarak (__init__) , (__del__) constructor yapıları verilebilir.

Class Variable Nedir?

Bu değişken **sınıftan türetilmiş tüm örnekler tarafından kullanılan ortak değişkendir. Class 'ın içerisinde tanımlanır.** Class 'ın dışında onun metodları (alt fonksiyonları) tarafından tanımlanamaz. Class değişkenleri ondan türetilmiş örnekler (instance) tarafından kullanılamaz.

İnstance Variable Nedir?

Bu değişken, bir metodun içinde tanımlanan ve sadece sınıfın o anki örneğine ait olan değişkendir.

Decorator Nedir?

Temelde **alınan parametreyi iç fonksiyona (wrapper) gönderen ve geriye de bu fonksiyonu döndüren fonksiyonlardır.** İç fonksiyonları dekoratör yazmadığınız durumlarda da kullanabilirsiniz.

Encapsulation (Sarmalama) Nedir?

Bu yapı bir sınıf içerisindeki değişkenlere **"kontrollü bir şekilde erişimi sağlamak / denetlemek"** için kullanılan yapıdır. **Class içerisindeki değişken private yapılarak dışarıdan direkt erişilmesi engellenir,** bu değişken içerisine değer atıp değer okumak için **get** ve **set** adı verilen metodlar kullanılır. **Yani direkt değişkene değil, bu değişkene erişmek(işlem yapmak) için bu metodlar ile çalışılır.** Set değer akıp değişkene atar, get 'de değeri geri döndürür, tabii bu işlem sizin belirlediğiniz, olması gereken kurallar çerçevesinde gerçekleşir.

Property nedir?

Bu fonk "property()" gömülü olarak Python da vardır. İşlevi biraz önce gördüğümüz encapsulation özelliği için kullanılır. **Özel olarak atadığımız değişkeni eğer getirmek istiyorsak direk getiremediğimiz için** property.getter metodunu kullanmak lazım, eğer düzeltmek istiyorsak property.setter metodunu kullanmamız lazım.

Self, cls nedir?

Self: Bir sınıf için yöntem tanımladığımızda **self, her durumda ilk parametre olarak kullanılır.** Self anahtarı belirli bir sınıfın bir örneği (nesne) temsil etmek için kullanılır. Self her nesnenin özniteliklerine ve yöntemlerine erişime izin

verir. Bu, her nesnenin kendi niteliklerine ve yöntemlerine sahip olmasını sağlar. Bu nedenle, bu **nesneleri oluşturmadan çok önce bile, nesnelere self sınıfı tanımlarken başvururuz.**

Cls: Sınıf metotları ise, **otomatik olarak, kendisini çağıran sınıfa veya örneğin sınıfına bir referans alır.** Bu argümana da **geleneksel olarak cls adı** verilir.

Class method nedir?

Parametre olarak sınıfın kendisini alır. Bu sayede sınıfla ilgili tüm bilgiye sahip olmaktadır. Eğer classmethod ile çalışıyorsanız sınıfın kendisi ya da nesne adı ile çağırılabilir. Classmethodlar **ilk parametreyi cls olarak sınıfın kendisi şeklinde almaktadır.** Bunları tanımlamak için @classmethod dekoratörü kullanılır.

Instance method nedir?

Her bir nesne derken burada bahsettiğimiz instance metottur ve instance metotlar **ilk parametre olarak self parametresini almak zorundadırlar** ki hizmet ettiği nesnenin diğer elemanlarına ulaşabilsin.

Static method nedir?

Kendisini hangi sınıf veya örneğin çağırdığını bilemez. Sadece **kendine verilen argümanları bilir,** örnek veya sınıf metotları gibi, gizli bir ilk argüman almazlar. Bu yönden bakıldığında, bu fonksiyonun **sınıf içinde yazılmasıyla, sınıf dışında yazılması arasında, hiçbir fark yoktur.** Ancak, **aynı modül içerisindeki birçok fonksiyonu anlamsal bütünler içinde toplamak gerektiğinde kullanılabilir.** Bunları tanımlamak için @staticmethod dekoratörü kullanılır.

import ve from anahtarları nasıl kullanılır?

main.py ile mixture.py diye iki tane dosyamız olsun. Main.py ‘da nesne oluşturma, komutsal işlemler yapılırken, mixture.py ‘da komutsal işlemleri oluştururuz. From ve import anahtarları nesnelerin oluşturulduğu yani main.py da kullanılır. **Eğer main.py ‘dan mixture.py ‘nin bilgilerini çağırmak istersek import dosyanın adı(mixture) import sınıf adı(Mixture) olarak yazmamız yeterli.**

Operator Overloading in Python with Special Methods?

Dunder methods da denilebilir.

+ __add__(self, other)

- __sub__(self,other)

*	__mul__(self,other)
/	__truediv__(self, other)
//	__floordiv__(self,other)
%	__mod__(self,other)
**	__pow__(self,other)
>>	__rshift__(self,other)
<<	__lshift__(self,other)
&	__and__(self,other)
|	__or__(self,other)
^	__xor__(self,other)

Kalıtım:

Miras alma olarak da bahsedilir. Bir kez yazılan kodların farklı yerlerde kullanılabilmesini sağlayan, bir kez yazılan kodların farklı yerlerde kullanılabilmesini sağlayan, bu bakımdan da programcıyı kod tekrarına düşmekten kurtaran oldukça faydalı bir araçtır.

Miras alma mekanizmasının işleyişi bakımından kabaca üç ihtimalden söz edebiliriz:

1- Miras alınan sınıfın bütün nitelik ve metotları alt sınıfa olduğu gibi devredilir.

2- Miras alınan sınıfın bazı nitelik ve metotları alt sınıfta yeniden tanımlanır.

3- Miras alınan sınıfın bazı nitelik ve metotları alt sınıfta değişikliğe uğratılır.

Taban sınıf(base class) birkaç farklı sınıfta ortak olan nitelik ve metotları barındıran sınıf türüdür. Üst sınıf(super class) diye de hitap edilir.
Alt sınıf(subclass) bir tabandan türeyen bütün sınıflar, o taban sınıfın alt sınıflarıdır. Alt sınıflar, kendilerinden türedikleri taban sınıfların metot ve niteliklerini miras yoluyla devralır.

Super(): Üst sınıfın değiştirmek istediğimiz niteliklerine yeni değerler atarken, değiştirmek istemediğimiz nitelikleri ise aynı şekilde muhafaza ettik.(eğer üst sınıfın __init__() metodundaki parametre listesini alt sınıfta da tek tek tekrar etmek sizi rahatsız ediyorsa şu şekilde de yazılabilir.(*arglar)

Eğer süper kullanmak istemiyorsak sınıfın kendi adını yazarız ve self ekleriz sonrasında ise süper de yaptığımız adımları yaparız.

Not: Yıldız parametreleri önceki derslerimizden hatırlıyor olmalısınız. Bildiğiniz gibi, tek yıldızlı parametreler bir fonksiyonun bütün konumlu(positional) argümanlarını, parametrelerin parantez içinde geçtiği sırayı dikkate alarak bir demet içinde toplar. İşte yukarıda da bu özellikten faydalanıyoruz. Eğer taban sınıfta isimli (keyword) argümanlar da olsaydı, o zaman da çift yıldızlı argümanları kullanabilirdik. Böylece konumlu argümanları bir demet içinde, isimli argümanları ise bir sözlük içinde toplamış oluyoruz. Bu da bizi üst (ya da taban) sınıfın parametre listesini alt sınıflarda tekrar etme derdinden kurtarıyor.

Çoklu Kalıtım:

Normal kalıtımda yaptığımız işlemler geçerlidir tek farkı bir alt sınıfa birden fazla üst sınıfı bağlamaktır. Ancak super() fonksiyonun kullanılmaması önerilir çünkü birden fazla sınıf olduğu için karışıklık olabilir.

ABC Method(abstract base class):

Bir işlemi yapan tek bir sınıf oluşturarak, bu işlemi yapmak isteyen diğer sınıfların kendisinden miras alınmasını ister. Oluşturulan bu tek sınıf, diğerlerine hangi methodları oluşturması konusunda bir yol haritası çizer, herhangi bir implementasyon barındırmaz. Tüm işlemler, miras aldıktan sonra yeni oluşturulan sınıf içerisinde yeniden yapılmalıdır.

OOP Kuralları

Abstraction = Olması gerekeni gösterir ama nasıl bir şey olduğunu göstermez. Evde mutfak var ama nasıl bir mutfak olduğu belli değil.

Duck typing = **Değişkenin türünü ayarlamaya gerek kalmaz.** Python da mesela değişkeni int, char gibi tanımlamalar yapmaya gerek yok. Otomatikmen kendisi algılıyor zaten.

Encapsulation = Bunun içi kendi içini ilgilendirir, bizi ilgilendirmez. Sadece ne için kullanacağımızı bilmemiz yeterli. Bir diğer adıyla kapsülleme. Bazı birimleri başkalarının kullanmayacağı şekilde saklayabiliriz, herkesin kullanımına açmayabiliriz. Gizli değişkenler kullanabiliriz (çift alt çizgi __secret)

Design Patterns = Yazılım geliştirme teknikleri.

Software Development

1-Tasarlama

2-İmplement(uygulama)

3-Test ve debug(ayıklama)

Pseudo-Code: Tasarım esnasında kod yazmak yanlış bir süreçtir. **Adım adım gitmek lazım; class, method, functions oluşturduktan sonra yorum satırı ile yanına o kod bloğunun görevini açıkla.** İlk olarak ana hatlarını yazmalıyız. Kod yazarak tasarlamak yanlış. Kağıt kalem al şu şunu yapar şunun işlevi şu diye yaz. İyice işi baştan sonra bitir. Zaman geçince aynı dökümanı tekrar oku bak eksik var mı. Tasarım desenlerini öğrenmek çok önemli.

Coding Style and Documantation = Bir anlaşma var herkes bu kurala uyarsa sıkıntı çıkmaz **(Green code).** Süreyi idareli kullanmak için. Her şeyi olması gerektiği yapacaksın senden önceki ve sonrakiler de kurallara uyacak ki kodu düzenlerken sıkıntı çıkmasın. **Yazılı kurallar değil ama uyulması lazım.** Mesela Python da ya tab ile boşluk bırakılır ya da 4 space ile boşluk bırakılır.

İdentifiers: Bir değişkene isim verirken onun işlevini tarif eden bir isim koyun. Mesela x,y,z diye değişken koymak yerine hesaplama fonk. yapan bir modüle calculator yazmak daha mantıklı. Sınıfın adı kişi olsun kişiler veya kişilerim olmasın(singular noun). Baş harfi büyük olsun. Birden fazla kelime birleşecek ise ilk harfleri büyük iki tane 3 tane kelime olabilir.

Member function(class 'ın içinde fonk) küçük harf ile başlamalı. Anlamı doğru olmalı, kelimelerin arasına alt çizgi konulmalı.

Bir nesne tanımlanırken küçük harf ile yazılır.

Documentation:

```
def scale(data, factor):
    """Multiply all entries of numeric data list by the given factor."""
    for j in range(len(data)): data[j] = factor
```

Constructor: Bir nesneyi ürettiğimiz zaman otomatik olarak çalışan ilk fonksiyon. Mesela yeni bir kredi kartı oluşturduğumuzda (__init__) fonk otomatik olarak çalışır. İçerisine self alır, yanına ise tanımlayacağımız nesnelerin ortak özelliklerini tanımlarız.