**Nesne Tabanlı Programlama (Object Oriented Programming)**

**Neden Nesne Tabanlı Programlama?**

Nesne tabanlı programlama bilmek zorunda değilsiniz. Özellikle Python da nesne tabanlı programlama bilmeden de çok güzel işler yapabilirsiniz. Python sizi nesne tabanlı programlama kullanmaya zorlamaz.

Peki, o zaman neden nesne tabanlı programlama öğreniyorum diyeceksiniz. Şimdi size nesne tabanlı programlama neden kullanmanız gerektiğini anlattığım da o zaman aslında nesne tabanlı programlamayı bilmenin bir zorunluluk olduğunu göreceksiniz. Hemen moralimizi bozmayalım. Çünkü Python da nesne tabanlı programlama çok basit hele ki diğer programlama dilleri ile karşılaştırdığınız zaman. İlk defa nesne tabanlı programlamayı öğreniyorsanız Python ile öğrendiğiniz için çok şanslısınız. O zaman neden nesne tabanlı programlama sorusuna? Cevap verelim.

·       Nesne tabanlı programlama ile yazılmış bir kodu okumak ve ne olduğunu anlamak için öğrenmelisiniz.

·       Çok büyük projelerle karşılaştığınız zaman işleri bölmek zorunda kaldığınız zaman kullanmak zorunda olacaksınız.

·       Son olarak iyi bir yazılımcı olmak istiyorsanız kesinlikle bilmek zorundasınız. Çünkü kod yazmak kadar okunabilir ve verimli kod yazmak da bir o kadar önemli.

Bu sayede nesne tabanlı programlama ile yazdığınız kodlar;

·       Okunabilir kodlar

·       Modülerlik iş parçacıklarına bölme

·       Kodları değiştirmede kolaylık

·       Projeyi daha kolay geliştirme ve genişletme

·       Kodların bakımı çok daha kolay

·       Modüllere ayırdığınız için kodları tekrar başka bir projede ekleme ve kullanabilme kolaylığı.

**Nesne Tabanlı Programlama Nedir?**

Nesne tabanlı programlama bir yazılım geliştirme yöntemidir. Nesne tabanlı programlama gerçek hayatın yazılıma uyarlanmış halidir. Nasıl mı? Şöyle bir etrafımıza baktığımız zaman etrafımız da birçok nesne var. Çokta uzağa gitmeye gerek yok şu an bu yazıyı okurken kullandığınız; bilgisayar, telefon veya tablet bir nesne. Bu nesnelerin bazı özellikleri ve işlevleri var. Örneğin bilgisayarınızın rengi, işlemci hızı ve markası bilgisayarın bazı özellikleri. Aynı zamanda bilgisayarın açma/kapatma düğmesi, tuşları vs. bu bilgisayarın bazı işleri yerine getirmesini sağlayan işlevlerdir.

İşte bizde bilgisayar nesnesi gibi nesneler oluşturacağız. Bu nesnelere bazı özellikler vereceğiz. Aynı zamanda bu nesnelere bazı işlemler yaptıracağız. Şimdi gerçek hayatta ki nesneler ile yazılım da oluşturduğumuz nesneleri karşılaştıralım.

Biz bir nesnenin özellikleri olduğunu söylemiştik peki bizim oluşturduğumuz nesnelerin özellikleri nedir? Bizim oluşturduğumuz nesnenin veri tipleri, operatörler vb. bizim nesnenin özellikleri olacak. Peki, işlevlerini nasıl oluşturacağız? Nesnenin işlevlerini ise; fonksiyon veya metotlar yardımıyla oluşturacağız. Tamam, çok güzel nesnenin ne olduğunu öğrendik peki nesneleri nasıl oluşturacağız?

[Python Dersleri #25 | Class Yapısı ve OOP (Object Oriented Programming) Giriş](https://www.youtube.com/watch?v=d2v_e2qKO44)

**Sınıf (Class) Nedir?**

Nesnelerimizi sınıflar yardımı ile oluşturacağız. Peki sınıf nedir? Daha önce ki derslerimizde bazı veri tiplerini görmüştük. Örneğin listeler, demetler, sözlükler vb. işte sınıflarda bu veri türleri gibi Python da bir veri türüdür.

**Sınıf**, bir nesnenin (object) özelliklerini (nitelikler) ve davranışlarını (metodlar) tanımlayan bir şablondur. Örneğin, bir Araba sınıfı, arabaların özelliklerini (renk, marka, model) ve davranışlarını (hızlanma, frenleme) tanımlayabilir.

**Sınıf Nitelikleri**

Örneğin renk, marka, model vb. bir sınıfın nitelikleridir.

**class Araba:**

**marka = ""**

**renk = ""**

**plaka = ""**

**Nesne (Object) Nedir?**

**Nesne**, bir sınıfın örneğidir. Bir sınıftan oluşturulan her nesne, sınıf tarafından tanımlanan özelliklere ve davranışlara sahip olur. Örneğin, Araba sınıfından oluşturulan bir nesne, belirli bir renge, markaya ve modele sahip olabilir.

**class Araba:**

**marka = ""**

**renk = ""**

**plaka =""**

**#sınıftan bir nesne türetiyoruz**

**araba1 = Araba()**

**araba1.marka ="FORD"**

**araba1.renk ="Siyah"**

**araba1.plaka ="34 P 148"**

**Sınıf ile Nesne(Object) arasında ki fark nedir?**

Buraya çok iyi dikkat edin. Bizim oluşturduğumuz Araba() sınıfı bir genel kavram. Şöyle anlatalım bütün arabaların bir markası, modeli veya bir rengi var. Yani genel bir sınıflandırma yapıyoruz. Ama siz bu arabanın rengini, markasını, modelini bilemezsiniz. Çünkü Araba() sınıfı bir soyut kavram. Bu yüzden soyut bir kavramın özelliklerini bilemezsiniz. Ama eğer rengi beyaz, modeli ford gibi özelliklerini belirttiğiniz zaman siz o sınıfı somutlaştırmış bir nesne(object) oluşturmuş olursunuz.

Şimdi oluşturduğumuz sınıftan bir nesne(object) daha oluşturalım.

**class Araba:**

**marka = ""**

**renk = ""**

**plaka =""**

**araba1 = Araba()**

**araba1.marka ="FORD"**

**araba1.renk ="Siyah"**

**araba1.plaka ="34 P 148"**

**araba2 = Araba()**

**araba2.marka ="BMW"**

**araba2.renk ="Beyaz"**

**araba2.plaka ="04 GH 48"**

**print("-------ARABA 1------")**

**print(f"Marka: { araba1.marka } \nRenk :{ araba1.renk }\nPlaka :{ araba1.plaka }")**

**print("-------ARABA 2------")**

**print(f"Marka: { araba2.marka } \nRenk :{ araba2.renk }\nPlaka :{ araba2.plaka }")**

Şimdi çıktıyı görelim.

**-------ARABA 1------       -------ARABA 2------**

**Marka: FORD                Marka: BMW**

**Renk :Siyah                    Renk :Beyaz**

**Plaka :34 P 148             Plaka :04 GH 48**

**Örnek (Instance) Nedir?**

Bir sınıftan oluşturulan her nesne, o sınıfın bir örneğidir ve bir örnek olarak adlandırılır. Örneğin, "araba1" ve "araba2" gibi farklı isimlerle oluşturulan iki araba nesnesi, Araba sınıfının iki ayrı örneğidir.

Gördüğünüz gibi arkadaşlar bir sınıftan birden fazla nesne oluşturabilirsiniz. Zaten Nesne tabanlı programlamanın amacı da bu, bir defa yaz çok defa kullan.

Sınıfı oluşturduk ve bu sınıfa bazı özellikler verdik. Dersin başında nesne kavramını anlatırken belli özellikleri ve işlevleri olduğunu söylemiştik. Şimdi sınıflarımıza bazı işlevler verelim örneğin; arabanın hızlanması, çalışması vs. o nesnenin bazı işlevleridir.

O zaman bizde arabamıza hız işlevlerini ekleyelim.

**def hizArttir(self):**

**self.hiz +=10**

**return self.hiz**

Nesnenin hız işlevini oluşturduk. Burada daha önce hiç bahsetmediğimiz **self** anahtar kelimesi var.

**Self Parametresi Neden Kullanılır?**

"**self**" parametresi, sınıf içindeki metotlarda kullanılan bir gelenektir. Bu parametre, bir örneği temsil eder ve o örneğin niteliklerine ve metodlarına erişmek için kullanılır. Self, siz dışarıdan bir fonksiyona veya özelliğe ulaşmaya çalıştığınız zaman bu anahtar kelime sayesinde ulaşabiliyorsunuz. Sınıf içinde oluşturduğunuz her fonksiyonun ilk parametresi self anahtar kelimesi olmak zorundadır. Bir metot çağrıldığında self Python tarafından otomatik olarak geçirilir. Şimdi oluşturduğumuz fonksiyonu çağıralım.

**araba.hizArttir()**

Şimdi diyeceksiniz self parametresini başta oluştururken fonksiyona eklediniz ama çağırırken vermediniz. Bir önceki derslerde fonksiyonları anlatırken bu şekilde anlatmamıştınız diyeceksiniz. Çağırırken self parametresi vermedik çünkü Python otomatik ekliyor. Sizin eklemenize gerek yok.

Kodlarımızın tamamını yazalım.

**class Araba:**

**marka = ""**

**renk = ""**

**plaka =""**

**hiz = 0**

**def hizArttir(self):**

**self.hiz +=10**

**return self.hiz**

**araba = Araba()**

**araba.marka ="FORD"**

**araba.renk ="Siyah"**

**araba.plaka ="34 P 148"**

**print("-------ARABA ------")**

**print(f"Marka: { araba.marka } \nRenk :{ araba.renk }\nPlaka :{ araba.plaka }")**

**araba.hizArttir()**

**print("Hız :",araba.hiz)**

Çıktımız şu şekilde olacaktır;

**-------ARABA ------**

**Marka: FORD**

**Renk :Siyah**

**Plaka :34 P 148**

**Hız : 10**

Eğer fonksiyonu ne kadar çağırsak fonksiyonu o kadar çalıştırmış oluruz.

**araba.hizArttir()**

**araba.hizArttir()**

**araba.hizArttir()**

**araba.hizArttir()**

**araba.hizArttir()**

Çıktımız;

**-------ARABA ------**

**Marka: FORD**

**Renk :Siyah**

**Plaka :34 P 148**

**Hız : 50**

Çok güzel artık kendi nesnemizi oluşturabiliyoruz, nesnelerimize özellik ve işlev verebiliyoruz. Şimdi nesnelerimizi biraz daha düzenleyelim. Örneğin biz sonradan nesnelere özellik verdik. Nesnemizi oluştururken özellik verebiliyor muyuz? Ona bakalım;

c**lass Araba:**

**marka = "FORD"**

**renk = "KIRMIZI"**

**plaka ="34 GH 100"**

**hiz = 20**

**araba = Araba()**

**print("-------ARABA ------")**

**print(f"Marka: { araba.marka } \nRenk :{ araba.renk }\nPlaka :{ araba.plaka }")**

Hemen çıktıya bakalım;

**-------ARABA ------**

**Marka: FORD**

**Renk :KIRMIZI**

**Plaka :34 GH 100**

Gördüğünüz gibi nesne tanımladığımız yani oluşturduğumuz zamanda nesneye özellik verebiliyoruz. Yalnız bir sorunumuz var. Hemen gösterelim;

**class Araba:**

**marka = "FORD"**

**renk = "KIRMIZI"**

**plaka ="34 GH 100"**

**hiz = 20**

**araba1 = Araba()**

**araba2 = Araba()**

**print("-------ARABA 1 ------")**

**print(f"Marka: { araba1.marka } \nRenk :{ araba1.renk }\nPlaka :{ araba1.plaka }")**

**print("-------ARABA 2 ------")**

**print(f"Marka: { araba2.marka } \nRenk :{ araba2.renk }\nPlaka :{ araba2.plaka }")**

Çıktıya bakalım;

**-------ARABA 1 ------       -------ARABA 2 ------**

**Marka: FORD                 Marka: FORD**

**Renk :KIRMIZI                Renk :KIRMIZI**

**Plaka :34 GH 100          Plaka :34 GH 100**

Gördüğünüz gibi iki nesne de aynı özelliklere sahip peki farklı türde nesne nasıl oluşturabiliriz?

**\_\_init\_\_() Fonksiyonu Nedir?**

**\_\_init\_\_()** fonksiyonu, bir sınıftan bir örnek oluşturulduğunda otomatik olarak çağrılan özel bir yöntemdir, Python yapıcı fonksiyonudur. Yapıcı fonksiyon nedir? Yapıcı fonksiyon Python nesne oluşturulduğunda default olarak çalıştırılan fonksiyondur. Az önce size self anahtar kelimesini anlatırken şu şekilde bir tanım yapmıştık: “Bu anahtar kelime oluşturduğumuz, fonksiyonun veya özelliğin referans olarak gösterir. ” aynı şekilde siz \_\_init\_\_() fonksiyonunu içinde akılda kalması için Python’nun referans anahtar kelimesi olarak aklınızda tutabilirsiniz. Bu yöntem, örneğin başlangıç durumunu (özelliklerin değerlerini) ayarlamak için kullanılır. \_\_init\_\_() fonksiyonunun kendisi, bir örneği temsil eden "self" parametresini alır ve sınıfın diğer niteliklerini bu fonksiyon içinde tanımlayabiliriz.

Hemen bir örnekle gösterelim o zaman çok daha iyi anlayacaksınız.

c**lass Araba:**

**def \_\_init\_\_(self):**

**print("Yapıcı fonksiyon \_\_init\_\_() fonksiyonu çalıştı...")**

**araba = Araba()**

Çıktıya bakalım:

**Yapıcı fonksiyon \_\_init\_\_() fonksiyonu çalıştı...**

Gördüğünüz gibi biz fonksiyonu çağırmadık. Hatırlarsanız biz hizArttir() fonksiyonu oluşturmuştuk ve fonksiyonu çağırmak için şöyle yapmıştık:

**araba.hizArttir()**

Ama  \_\_init\_\_() fonksiyonu için yapmadık çünkü biz tanımlarken fonksiyonu otomatik olarak çalıştırılan fonksiyon demiştik. Bu yüzden programı çalıştırdığımız zaman otomatik olarak bu fonksiyon çalıştı.

Şimdi \_\_init\_\_() fonksiyonunu kullanalım;

**class Araba:**

**def \_\_init\_\_(self, marka, renk, plaka, hiz):**

**self.marka = marka**

**self.renk = renk**

**self.plaka = plaka**

**self.hiz = hiz**

**araba1 = Araba("Fiyat", "Mavi", "25 P 45", 30)**

**araba2 = Araba("Ford", "Beyaz", "14 YH 566", 100)**

**print("------Araba 1 ------")**

**print(araba1.marka)**

**print(araba1.renk)**

**print(araba1.plaka)**

**print(araba1.hiz)**

**print()**

**print()**

**print("-----Araba 2 -------")**

**print(araba2.marka)**

**print(araba2.renk)**

**print(araba2.plaka)**

**print(araba2.hiz)**

Çıktıya bakalım;

**------Araba 1 ------       -----Araba 2 -------**

**Fiyat                              Ford**

**Mavi                              Beyaz**

**25 P 45                         14 YH 566**

**30                                  100**

Güzel peki burada default olarak değer verebiliyor muyuz? Hemen bir örnekle gösterelim;

**class Araba:**

**def \_\_init\_\_(self, marka="Fiyat", renk="Mavi", plaka="25 P 45", hiz=30):**

**self.marka = marka**

**self.renk = renk**

**self.plaka = plaka**

**self.hiz = hiz**

**araba1 = Araba()**

**print("------Araba 1 ------")**

**print(araba1.marka)**

**print(araba1.renk)**

**print(araba1.plaka)**

**print(araba1.hiz)**

Çıktıya bakalım ;

**------Araba 1 ------**

**Fiyat**

**Mavi**

**25 P 45**

**30**

Peki ikinci bir nesne oluşturduğumuzda yine aynı özelliklere mi sahip olacak diyorsanız. Şöyle bir örnek verelim:

**class Araba:**

**def \_\_init\_\_(self, marka="Fiyat", renk="Beyaz", plaka="25 P 45", hiz=30):**

**self.marka = marka**

**self.renk = renk**

**self.plaka = plaka**

**self.hiz = hiz**

**araba1 = Araba()**

**araba2 = Araba("Ford","Beyaz","14 YH 566",100)**

**print("------Araba 1 ------")**

**print(araba1.marka)**

**print(araba1.renk)**

**print(araba1.plaka)**

**print(araba1.hiz)**

**print()**

**print()**

**print("-----Araba 2 -------")**

**print(araba2.marka)**

**print(araba2.renk)**

**print(araba2.plaka)**

**print(araba2.hiz)**

Çıktıya bakalım;

**------Araba 1 ------       -----Araba 2 -------**

**Fiyat                              Ford**

**Beyaz                            Beyaz**

**25 P 45                         14 YH 566**

**30                                  100**

Gördüğünüz gibi iki objede farklı.

Default olarak değer verebiliyoruz. Peki hepsine default değer vermek istemiyorum ben istediğim değeri default değer vermek istiyorum diyorsan o zaman şöyle bir örnek gösterelim.

**class Araba:**

**def \_\_init\_\_(self, marka="Fiyat", renk=None, plaka="25 P 45", hiz=30):**

**self.marka = marka**

**self.renk = renk**

**self.plaka = plaka**

**self.hiz = hiz**

**araba1 = Araba(renk="Beyaz")**

**araba2 = Araba("Ford","Beyaz","14 YH 566",100)**

**print("------Araba 1 ------")**

**print(araba1.marka)**

**print(araba1.renk)**

**print(araba1.plaka)**

**print(araba1.hiz)**

Çıktıya bakalım;

**------Araba 1 ------**

**Fiyat**

**Beyaz**

**25 P 45**

**30**

Gördüğünüz gibi default değer vermek istemediğimiz özellikler için None anahtar değerini veriyoruz. Biz burada None ile boş değer verdiğimizi sonradan değer ekleyeceğimizi söylüyoruz.

İşte bu konseptleri daha iyi anlamak için bir örnek:

**# Araba sınıfı örneği**

**class Araba:**

**def \_\_init\_\_(self, renk, marka, model):**

**self.renk = renk**

**self.marka = marka**

**self.model = model**

**def hizlan(self, hiz\_miktari):**

**print(f"{self.marka} {self.model}, {hiz\_miktari} km/s hızlandı.")**

**def frenle(self):**

**print(f"{self.marka} {self.model}, durdu.")**

**# Araba sınıfından nesne örneği oluşturma**

**araba1 = Araba("Kırmızı", "Toyota", "Corolla")**

**print(araba1.renk)  # Çıktı: Kırmızı**

**araba1.hizlan(50)  # Çıktı: Toyota Corolla, 50 km/s hızlandı.**

**araba1.frenle()    # Çıktı: Toyota Corolla, durdu.**

[Python Dersleri #26 | Class'larda Method'lar](https://www.youtube.com/watch?v=1F7JeGveCKE)

\*\*\*\*\*\*\*\*\*

### Nesne Tabanlı Programlama Metodolojileri

1.    Miras Alma (Inheritance)

2.    Çok Biçimlilik (Polymorphism)

3.    Soyutlama (Abstraction)

4.    Kapsülleme (Encapsulation)