# Python ile Nesne Tabanlı Programlama(Object Oriented Programming)

Mehmet TEK · Follow
Published in Rexven · 4 min read · May 21, 2020




Bu yazımda sizlere Python programlama dilinde Nesne Tabanlı Programlama nasıl yapılır anlatmaya çalışacağım

> *Object oriented programming (OOP) ifadesini Türkçe'ye Nesne Yönelimli Programlama ya da Nesne Tabanlı Programlama olarak çevirebiliriz.*

Aslında gerçek hayattan örnekler ile ilerlemek daha faydalı olacaktır. Gerçek hayatta kullanılan herşey birer **nesne (obje)** dir. Mesela telefonumuz bir nesnedir. Bu nesnenin belirli özellikleri ve yaptığı işlevler vardır. Telefonunuzun markası , şarj süresi ve depolama kapasitesi onun özellikleridir. Telefonunuz ile fotoğraf çekmeniz ve arama yapmanızda bu nesnenin davranışları ve faaliyetleridir. Bizim Python'da oluşturacağımız nesnelerin de **özellikleri (attributes)** ve **davranışları (methods)** olacak.

. . .

> ***Nesne Tabanlı Programlama ne işimize yarayacak?***

Bir okulda çalışan insanların ve öğrencilerin verilerini tutup üzerinde işlem yapacağınızı farz edelim. Nasıl bir kod yazmamız gerekirdi?

In [157]:
```
bir_isim = "mehmet"
bir_soyisim = "tek"
bir_yaş = 19
bir_not_ort = 85

iki_isim = "ali"
iki_soyisim = "yılmaz"
iki_yaş = 18
iki_not_ort = 75

üç_isim = "kemal"
üç_soyisim = "çelik"
üç_yaş = 20
üç_not_ort = 95
```

Örneğin üç öğrencinin özelliklerini yazmak bile bu kadar sürdü. Bir okulda 1000 öğrenci olduğunu düşünün. Ayrıca bir okulda sadece öğrenciler bulunmuyor. Aslında bir okulu 4 sınıfa ayırabiliriz. Bir okulda öğrenciler, öğretmenler, yönetim ve hizmetliler olmak üzere 4 sınıf insan vardır. Bu 4 sınıfın aynı ve ayrı özellikleri olacak elbette. Mesela isim, soyisim ve yaş gibi benzer özellikleri olacak. Ama öğrencilerin not ortalaması gibi özellikleri olacakken, öğretmenlerin maaşları gibi özellikleri olacak. Okulda çalışan bütün insanlara bu şekilde tek tek değişken atayarak veriler tanımlarsak 1000 kişi olduğunda işin içinden çıkılamaz hale gelecektir. İşte burada Nesne Tabanlı Programlama imdadımıza yetişiyor ve işlerimizi kolaylaştırıyor.

. . .

> ***Daha iyi anlaşılması için hadi hemen bir öğrenci sınıfı (class) tanımlayalım***

```
In [1]: class Ogrenci:
            isim = "mehmet"
            soyisim = "tek"
            yaş = 29
            not_ort = 85
```

```
In [4]: Ogrenci.isim
Out[4]: 'mehmet'
```

```
In [5]: Ogrenci.yaş
Out[5]: 29
```

Class kelimesini kullanarak bir öğrenci sınıfı tanımladık. Her öğrencide bulunmasını belirlediğimiz 4 tane özelliğini girdik (isim, soyisim, yaş ve not ortalaması). Resimde gördüğünüz şekilde öğrencinin özelliklerine erişebildik.

. . .

> ***Şimdi isimleri Mehmet ve Ali olacak şekilde iki tane öğrenci nesnesi oluşturalım***

```
In [7]: birinci = Ogrenci()
        birinci
Out[7]: <__main__.Ogrenci at 0x1119ac0d0>

In [8]: birinci.isim
Out[8]: 'mehmet'

In [9]: ikinci = Ogrenci()

In [10]: ikinci.isim
Out[10]: 'mehmet'

In [11]: ikinci.isim = "ali"

In [12]: birinci.isim
Out[12]: 'mehmet'

In [13]: ikinci.isim
Out[13]: 'ali'
```

Burada iki öğrenci nesnesi oluşturduk. Bu şekilde bir çok öğrenci nesnesi oluşturabiliriz ama hepsinin değerleri aynı oluyor. Bizim her öğrencinin özelliklerini farklı şekilde yapabiliyor olmamız lazım. Aşağıdaki şekilde yaptığımızda oluşturduğumuz her nesnenin farklı özellikleri olabilecek.

## Initializer or Constructor

```
Bizim sınıfımızın ana yapısını oluşturan bir metod ==>  __init__()
```

```
In [47]: class Ogrenci:

             def __init__(self,isim,soyisim,yaş,not_ort):
                 self.nesnenin_ismi = isim
                 self.nesnenin_soyismi = soyisim
                 self.nesnenin_yaşı = yaş
                 self.nesnenin_not_ort = not_ort


In [48]: bir = Ogrenci("mehmet","tek",29,85)

In [49]: iki = Ogrenci("ali","yılmaz",27,90)

In [51]: bir.nesnenin_ismi
Out[51]: 'mehmet'

In [52]: iki.nesnenin_ismi
Out[52]: 'ali'
```

Bakın burada öğrenci sınıfımızdan iki tane nesne oluşturduk. İstediğimiz kadar nesneyi kolaylıkla bu yöntemi kullanarak oluşturabiliriz. Şimdiye kadar görmediğiniz ***___init___()*** isiminde bir metot eklendi. Bu metot bizim sınıfımızın ana yapısını oluşturuyor. Ayrıca daha önce görmediğiniz ***self*** kelimesini kullanmaya başladık. Bu ***self*** kelimesi bizim nesnelerimizi tanımlıyor.

*Aslında* ***self.nesnenin_ismi = isim*** şeklinde yazılmasına gerek yok. Anlaşılması için böyle yazdım.Burada ***self.isim*** zaten nesnenin ismi demek. Biz şu ana kadar sadece nesnelerin özelliklerini (attributes) belirledik. Yazının başında belirttiğim gibi nesnelerin faaliyetleri ve işlevleri (methods) de olacak.

```
In [158]: class Ogrenci:

              def __init__(self,isim,soyisim,yaş,not_ort):
                  self.isim = isim
                  self.soyisim = soyisim
                  self.yaş = yaş
                  self.not_ort = not_ort

              def info(self):
                  print("{} {} {} yaşında ve {} notu olan bir öğrencidir".format(self.isim,self.soyisim,self.yaş,self.not_ort)

In [159]: bir = Ogrenci("mehmet","tek",29,85)

In [160]: bir.info()
          mehmet tek 29 yaşında ve 85 notu olan bir öğrencidir

In [161]: iki = Ogrenci("ali","yılmaz",27,90)

In [162]: iki.info()
          ali yılmaz 27 yaşında ve 90 notu olan bir öğrencidir
```

Burada iki öğrenci nesnesi oluşturduk. Öğrencilerin bilgilerini gösteren ***info*** isminde bir metot tanımladık. Metotlar aslında sınıflar içinde tanımlanan fonksiyonlardır. Bir fonksiyon sınıf içinde tanımlanmış ise metot ismini alır.

. . .

> ***Metotları daha iyi anlamanız için yeni bir öğretmen sınıfı oluşturalım ve öğretmenlerin maaşına %30 zam yapacak bir metot yazalım***

```
In [180]: class Ogretmen:
              def __init__(self,isim,soyisim,yaş,maaş,uzmanlık):
                  self.isim = isim
                  self.soyisim = soyisim
                  self.yaş = yaş
                  self.maaş = maaş
                  self.uzmanlık = uzmanlık
              def info(self):
                  print(
                  """
                  isim = {}
                  soyisim = {}
                  yaş = {}
                  maaş = {}
                  uzmanlık = {}
                  """.format(self.isim,self.soyisim,self.yaş,self.maaş,self.uzmanlık))
              def zam(self):
                  return self.maaş * 1.3

In [181]: bir = Ogretmen("mustafa","tek",35,5000,"Fizik")

In [182]: bir.info()

                  isim = mustafa
                  soyisim = tek
                  yaş = 35
                  maaş = 5000
                  uzmanlık = Fizik

In [183]: bir.zam()
Out[183]: 6500.0
```

Evet Öğretmen sınıfı oluşturup bir nesne oluşturduk. Zam isminde bir metot tanımladık. Bu metot öğretmenin maaşını %30 oranında artırıyor. Biraz yukarıda metot ve fonksiyonların farkından bahsettik.Daha iyi anlaşılması için şimdi onunla ilgili bir örnek yapalım

```
Metodlar aslında sınıflar içinde tanımlanan fonksiyonlardır. Yani bir fonsiyon sınıf içinde tanımlanıyorsa metod
oluyor.

In [43]: class Ogretmen:

             def __init__(self,isim,soyisim,yaş,maaş,uzmanlık):
                 self.isim = isim
                 self.soyisim = soyisim
                 self.yaş = yaş
                 self.maaş = maaş
                 self.uzmanlık = uzmanlık

             def info(self):
                 print("{} {} {} yaşında ve maaşı {} olan bir {} öğretmenidir ".format(self.isim,self.soyisim,self.yaş,self.m

             def zam(self):
                 return self.maaş * 1.3

         def info(isim,soyisim,yaş,maaş,uzmanlık):
             print("{} {} {} yaşında ve maaşı {} olan bir {} öğretmenidir ".format(isim,soyisim,yaş,maaş,uzmanlık))

In [44]: info("Ahmet","durmaz",37,6000,"Matematik")
         Ahmet durmaz 37 yaşında ve maaşı 6000 olan bir Matematik öğretmenidir

In [45]: bir = Ogretmen("mustafa","tek",35,5000,"Fizik")

In [46]: bir.info()
         mustafa tek 35 yaşında ve maaşı 5000 olan bir Fizik öğretmenidir

         Metodlar ile fonksiyonlar farklı, dışarıdaki fonksiyonun sınıf ile bir ilgisi kalmıyor
```

Örnekte de görüldüğü gibi eğer sınıf içinde bir fonksiyon tanımlarsak bu metot oluyor. Sınıfın dışında tanımladığımız fonksiyonun sınıf ile bir ilişkisi olmuyor. Gördüğünüz gibi fonksiyon ve metotların çağırılması da birbirinden farklı.

Evet bu yazımızda Nesne Tabanlı Programlamaya giriş yapmış olduk.Bir sonraki yazımızda da ;

- *Encapsulation — Kapsülleme (Erişimi Engelleme)*
- *Inheritance (Kalıtım)*
- *Abstract Class — Soyut Sınıflar*
- *Overriding (Üzerine Yazma-Öncekini İptal Etme)*
- *Polymorphism — Çok Şekillilik*

Konularını anlatıyor olacağım.Bir sonraki yazıda görüşmek üzere …

. . .

Udemy platformunda şu an 18'den fazla İngilizce ve Türkçe kursum bulunmakta.Dünya genelinde 100.000'den fazla öğrenciye programlama, yazılım ve e-ticaret alanlarında eğitimler vermekteyim.

Udemy eğitimlerime **Mehmet TEK — Udemy** adresinden ulaşabilirsiniz.Görüşmek üzere ...

Python | Nesne Tabanlı Programlama | Object Oriented | Python3 | Python Programming