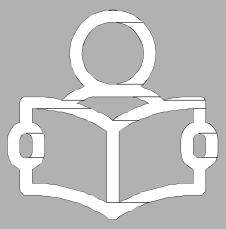
PYTHON

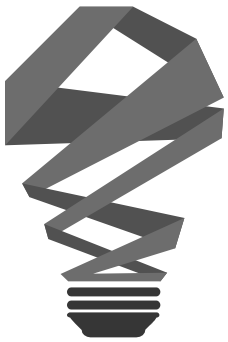
Nesne Tabanlı Programlama

Kapsam



- Nesne Tabanlı Programlama (NTP) Nedir?
- Sınıflar ve Nesneler
- Sınıf Metotları
- Kapsülleme
- Kalıtım (Miras Alma)

Nesne Tabanlı Programlama (NTP) Nedir?



Nesne Tabanlı Programlama bir programlama paradigmasıdır. Bu yaklaşımda programlama mantığı gerçek hayattaki nesneler gibi tasarlanmıştır.

Gerçek hayatta bir nesneye ihtiyacınız olduğunda onu alır ve kullanırsınız. Her nesnenin kendine ait özellikleri ve işlevleri vardır.

Nesneler tanımlanırken onların özelliklerinden ve işlevlerinden yararlanılır.

Sınıflar ve Nesneler

Python'da tüm yapılar sınıflar içinde oluşturulur. Python'da kullanılan veri türlerinin her biri bir sınıftır.

- NTP, bir nesne olarak yapının tüm özelliklerinin ve işlevlerinin bir yerde toplanmasına ve kullanılmasına olanak verir.
- Bir sınıf tanımlandığı zaman bir model oluşturulur. Bu model o sınıfa ait tüm nesneleri temsil eden bir şablondur.
- Bu modeli kullanarak o sınıftaki tüm nesneler türetilebilir. Bir sınıftan bir nesne türetildiğinde yapıya ilişkin modelin bir kopyası oluşur.
- Program akışına göre nesneler diğer nesnelerle etkileşime girerek özelliklerini ve işlevlerini o nesnelerin erişimine açabilir.
- Başka bir nesne, eriştiği nesneden aldığı değeri kullanarak yeni işlemler yapabilir.

Bir Sınıf Tanımlama

Bir sınıf şöyle tanımlanır; (jargon olarak sınıf adlarının ilk harfi büyük yazılır)
class SinifAdi: veya class SinifAdi():
Blok yapısı içinde yer alır.
Özellikler ve metotlar sınıfın içinde tanımlanır.

Kod

```
class OrnekSinif:
    def __init__(self):
        print ("Merhaba Sınıf")
```

Sınıftan Bir Nesne Oluşturma

Sınıflardan nesne türetme (instance) örnek oluşturma olarak adlandırılır.
OrnekSinif sinifindan bir nesne türetelim.

Kod

```
nesnem=OrnekSinif()
print(nesnem)
```

Nesnenizi oluşturdunuz. print( ) kullandığınızda nesnenin ait olduğu sınıfın adı ve hafızadaki yeri yazdırılacaktır.

Yapıcı "__init__()" Fonksiyonu

Bir sınıftan bir nesne oluşturulduğunda otomatik olarak çalışan bir fonksiyondur. NTP'de yapıcı (constructor) fonksiyon olarak adlandırılan bu fonksiyonlar nesne için zorunlu parametreleri işler ve nesnenin oluşturulmasını sağlar.
Yapıcı "__init__()" fonksiyonu self olmadan tanımlanamaz.

Kod

```
class MetinSinifi:
    def __init__(self, ifade, tekrarSayisi, kacisKarakteri):
        print((ifade+kacisKarakteri)*tekrarSayisi)

nesnem=MetinSinifi("Sınıf oluşturduk.", 5, "\n")
```

Verilen metin ifadelerini istenilen kadar yazan bir sınıf tanımladık.
Sınıf için bazı zorunlu parametreler oluştu.

Varsayılan Değerler

NTP'de bir fonksiyon farklı parametrelerle tanımlanabilir.
Bazı parametreler için varsayılan değerler atanabilir.
Aşağıdaki kodda kacisKarakteri varsayılan olarak '\n' verilmiştir.

Kod

```
class MetinSinifi:
    def __init__(self, ifade, tekrarSayisi, kacisKarakteri="\n"):
        print((ifade+kacisKarakteri)*tekrarSayisi)

nesnem=MetinSinifi("Sınıf oluşturduk.", 5)
```

NOT

Argümanlarınızı tanımlarken varsayılan değerli argümanlar, zorunlu argümanlardan sonra gelmelidir.

Argümanlara Erişim

Sınıf oluştururken verdiğiniz argümanlara erişmek için aşağıdaki kodları kullanabilirsiniz.

Kod

```
class MetinSinifi:
    def __init__(self, ifade, tekrarSayisi, kacisKarakteri='\n'):
        self.ifade=ifade
        self.tekrarSayisi=tekrarSayisi
        print ((ifade+kacisKarakteri)*tekrarSayisi)
```

Örnek 1

Öğrenci verilerini ve öğrencilere ait işlemlerin yapıldığı bir sınıf oluşturalım.

Kod

```
class Ogrenci():
    ogrenciler=[]
    def __init__(self, AdıSoyadı):
        self.AdıSoyadı = AdıSoyadı
        self.dersleri = []
        self.ogrenciEkle(self.AdıSoyadı)
    def ogrenciEkle(self, adSoyad):
        self.ogrenciler.append(adSoyad)
        print('{} adlı kişi öğrencilere eklendi.'.format(adSoyad))
    def ogrenciListesiYazdir(self):
        print('Öğrenci listesi')
        for ogrenci in self.ogrenciler:
            print(ogrenci)
    def dersEkle(self, dersAdi):
        self.dersleri.append(dersAdi)
    def ogrencininDersleri(self):
        print('{} adlı kişinin dersleri:'.format(self.AdıSoyadı))
        for ders in self.dersleri:
            print(ders)
```

Örnek 1

Kod

```
ogrenciNesnesi=Ogrenci('Murat Çalışkan')#Bir nesne türeterek bir öğrenci ekleyebilirsiniz.
print('Öğrenci Sayısı:', len(ogrenciNesnesi.ogrenciler)) #Öğrenci sayısına bakabilirsiniz.
ogrenciNesnesi.dersEkle('Türkçe')
ogrenciNesnesi.dersEkle('Matematik')
ogrenciNesnesi.dersEkle('Fen Bilgisi') #Öğrencilere ders ekleyebilirsiniz.
ogrenciNesnesi.ogrencininDersleri()#Öğrencinin derslerini listeleyebilirsiniz.

ogrenciNesnesi.ogrenciEkle('Ahmet Birinci') #Yeni bir öğrenci ekleyebilirsiniz.
ogrenciNesnesi.ogrenciListesiYazdir() #Öğrenci listesini görebilirsiniz.
```

Kapsülleme

Sınıf içindeki metotlara ve özelliklere erişimleri kısıtlamak için kapsülleme yöntemi kullanılır.
Bir sınıf tanımlanırken kapsülleme ile erişilmesi, değiştirilmesi istenmeyen veya sadece kısıtlı erişim verilmek istenilen özellikler ve/veya metotlar tanımlanabilir.
Bunun için istenilen metodun veya özelliğin başına "__" eklenmelidir.

Kod

```
class Ogrenciler:
    def __init__(self,adi,yasi,TCNo):
        self.adi=adi
        self.yasi=yasi
        self.__TCNo=TCNo
    def ogrenciYaz(self):
        print('Öğrenci Bilgileri')
        print(self.adi)
        print(self.yasi)
        print(self.__TCNo)
ogrenci=Ogrenciler('Murat Çalışkan',17,54639876211)
ogrenci.ogrenciYaz()
print(ogrenci.__TCNo) #hata verir
```

Örnekte self.__TCNo=TCNo kullanımı öğrencinin kimlik numarasına dışarıdan erişimi engeller.

Kapsülleme

Sınıf içinde özel olarak tanımlanmış bu tür özelliklere erişmek için "set" ve "get" metodları bulunur.
"get" özellikleri okumak "set" ise değiştirmek için kullanılır.

Kod

```
class Ogrenciler:
    def __init__(self,adi,yasi,TCNo):
        self.adi=adi
        self.yasi=yasi
        self.__TCNo=TCNo
    def ogrenciYaz(self):
        print('Öğrenci Bilgileri')
        print(self.adi)
        print(self.yasi)
        print(self.__TCNo)
    def get_TCNo(self):
        return str(self.__TCNo)[7:]

ogrenci=Ogrenciler('Sezai', 23, 12345678901)
print(ogrenci.get_TCNo())
```

Yukarıda tanımlanan def get_TCNo(self): ile kimlik numarasının son 4 hanesi döndürülmüştür.
Kapsülleme sınıfın içindeki özellikleri ve metotları korumayı sağlamaktadır. Kapsülleme aynı zamanda verinin güvenliğini sağlar.

Kapsülleme

"get" metodu ile yazma özelliği verebilir, "set" metodu ile de değişkene yeni değer atayabilirsiniz.

Kod

```
class Ogrenciler:
    def __init__(self,adi,yasi,TCNo):
        self.adi=adi
        self.yasi=yasi
        self.__TCNo=TCNo
    def ogrenciYaz(self):
        print('Öğrenci Bilgileri')
        print(self.adi)
        print(self.yasi)
        print(self.__TCNo)
    def get_TCNo(self):
        return str(self.__TCNo)[7:]
    def set_TCNo(self,TcNoDuzelt):
        self.__TCNo=TcNoDuzelt
#Yanlış girilen kimlik numarasının düzeltildiğin varsayınız.
ogrenci=Ogrenciler('Kaan Emre', 33, 23142343655) #Öncelikle bir öğrenci tanımlayalım.
print('Öğrenci TC No son 4 hane: ', ogrenci.get_TCNo())
ogrenci.set_TCNo(12309768687)
print ('Öğrenci TC No son 4 hane: ', ogrenci.get_TCNo())
```

Kalıtım (Miras Alma)

Kalıtım bir sınıfın başka bir sınıfın metotlarını ve özelliklerini kendi bünyesine almasıdır. Bu metot ve özelliklerin devralındığı (Miras alma da denir.) sınıfa üst sınıf, devralan sınıfa ise alt sınıf denir. Alt sınıflar, üst sınıfın tüm metotlarını ve özelliklerini devralır, ancak farklı özellikler ve metotlar da ekleyebilir. En temel sınıf türü, genellikle diğer tüm sınıfların ebeveynleri olarak miras aldığı bir nesnedir.

Örneğin bir "kedi" sınıfının metotları ve özellikleri neler olabilir?
Her kedinin bir adı, yaşı ve cinsi vardır. Buna ek olarak her kedi cinsinin farklı özellikleri de vardır. Her kedi cinsi, üst sınıf olan kedi sınıfının alt sınıfıdır. Üst sınıflara "ebeveyn (parent) sınıf" alt sınıflara ise "çocuk (child) sınıf" denir.

Kalıtım

Öncelikle bir kedi üst sınıfı tanımlayalım.

Kod

```
#Üst Sınıf
class Kedi:
    def __init__(self, adi, yasi):
        self.adi = adi
        self.yasi = yasi
    def ozellikler(self):
        return "{} {} yaşında".format(self.adi, self.yasi)
```

Kalıtım

Kedi üst sınıfından farklı türlerdeki kediler için alt sınıflar tanımlayalım.

Kod

```
# Kedi üst sınıfından miras alan alt sınıflar
class PersKedisi(Kedi):
    def tuyleri(self, tuyuNasil):
        return "{} tüyleri {}".format(self.adi, tuyuNasil)
class VanKedisi(Kedi):
    def gozRengi(self, gozRengiNe):
        return "{} gözleri {}".format(self.adi, gozRengiNe)
```

Kalıtım

Alt sınıftan “pisi” ve "persia" adlarında kediler türetelim.

Kod

```
pisi = VanKedisi('Pisi', 4)
print(pisi.ozellikler())
print(pisi.gozRengi('Mavi ve kahverengi'))

persia= PersKedisi('Persia', 4)
print(persia.ozellikler())
print(persia.tuyleri('Yumuşak'))
```

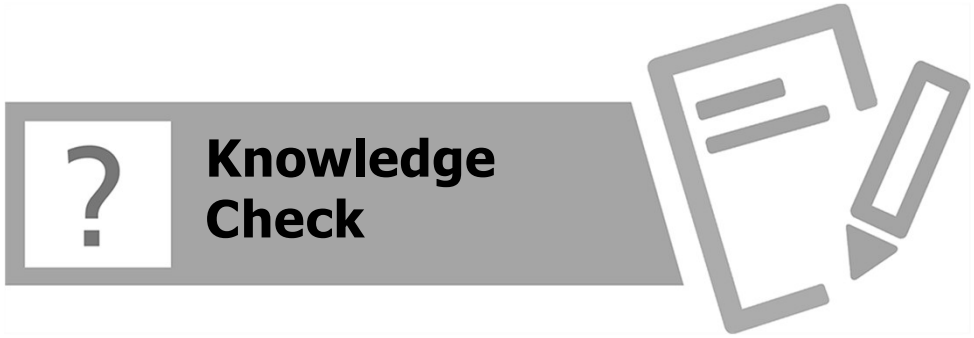
Overridding

Diğer alt sınıfın da kendine ait özelliklerini ve üst sınıftan devraldığı özellik ve metotları çağırabilirsiniz.
Üst sınıftan devralınan metotları ve özellikleri geçersiz (overridding) kılabilirsiniz.

Kod

```
class VanKedisi(Kedi):
    def gozRengi(self, gozRengiNe):
        return "{} gözleri {}".format(self.adi, gozRengiNe)
    def ozellikler(self):
        return "Türü: Van kedisi adı:{} ve {} yaşında".format(self.adi, self.yasi)
#overridding
pisi = VanKedisi('Pisi', 4)
print(pisi.ozellikler())
print(pisi.gozRengi('Mavi ve kahverengi'))
```

Yukarıdaki örnekte özellikler metodunu geçersiz kılınarak üst sınıftan farklı bir değer döndürülmüştür.



1	**Bir araba sınıfı tanımlayınız. Özellikler: marka, model, kapı sayısı, rengi, fiyatı, motor seri numarası (gizli özellik). Marka ve model dışındaki tüm özelliklerin varsayılan değeri yok (None), kapı sayısı varsayılan değeri=2 olarak tanımlanacaktır. Arabanın özelliklerini yazdıran bir metot tanımlayınız.**

1	**Bir araba sınıfı tanımlayınız. Özellikler: marka, model, kapı sayısı, rengi, fiyatı, motor seri numarası (gizli özellik). Marka ve model dışındaki tüm özelliklerin varsayılan değeri yok (None), kapı sayısı varsayılan değeri=2 olarak tanımlanacaktır. Arabanın özelliklerini yazdıran bir metot tanımlayınız.**

```
class Araba():
    def __init__(self, marka, model, kapıSayısı=2, rengi=None, fiyatı=None, motorSeriNumarası=None):
        self.marka=marka
        self.model=model
        self.kapıSayısı=kapıSayısı
        self.rengi=rengi
        self.fiyatı=fiyatı
        self.__motorSeriNumarası=motorSeriNumarası
    def özellikleri(self):
        print("{} marka {} model {} kapılı {} renginde {} fiyatlı arabam var.". format(self.marka, self.model, self.kapıSayısı, self.rengi, self.fiyatı))
```

Doğru Cevap

2	**Bu araba sınıfından 90.000 TL. değerinde sarı renkli 2 kapılı 1993 model bir "Marka 00" marka araba türetiniz. Motor şase numarası 123456789 olsun. Daha sonra bu arabanın özelliklerini yazdırınız.**

2	**Bu araba sınıfından 90.000 TL. değerinde sarı renkli 2 kapılı 1993 model bir "Marka 00" marka araba türetiniz. Motor şase numarası 123456789 olsun. Daha sonra bu arabanın özelliklerini yazdırınız.**

```
arabam=Araba("Marka 00", 1993, 2, "sarı", "90.000 TL.", motorSeriNumarası=123456789)
arabam.özellikleri()
```

Doğru Cevap

3	**Motor şase numarasını okumak ve değiştirmek için sınıfı düzenleyiniz. Şase numarasını değiştirerek ekrana yazdırınız.**

3

Motor şase numarasını okumak ve değiştirmek için sınıfı düzenleyiniz. Şase numarasını değiştirerek ekrana yazdırınız.

```
class Araba():
    def __init__(self, marka, model, kapıSayısı=2, rengi=None, fiyatı=None, motorSeriNumarası=None):
        self.marka=marka
        self.model=model
        self.kapıSayısı=kapıSayısı
        self.rengi=rengi
        self.fiyatı=fiyatı
        self.__motorSeriNumarası=motorSeriNumarası
    def özellikleri(self):
        print("{} marka {} model {} kapılı {} renginde {} fiyatlı arabam var.". format(self.marka,
self.model, self.kapıSayısı, self.rengi, self.fiyatı))
    def get_motorSeriNumarası(self):
        return self.__motorSeriNumarası
    def set_motorSeriNumarası(self, motorSeriNoDuzeltme):
        self.__motorSeriNumarası=motorSeriNoDuzeltme
```

Doğru Cevap

```
arabam=Araba('Marka 123', 1981)
arabam.özellikleri()
arabam.set_motorSeriNumarası(123456) #yeni seri numara değerini verdik
print(arabam.get_motorSeriNumarası())
```

4

Araba sınıfından kalıtım yoluyla bir sınıf üretiniz ve gazVer metodunda ekrana 'Gaza basıldı' yazmasını sağlayınız. Bu alt sınıfa çelik jant özelliği ekleyiniz.

4

Araba sınıfından kalıtım yoluyla bir sınıf üretiniz ve gazVer metodunda ekrana 'Gaza basıldı' yazmasını sağlayınız. Bu alt sınıfa çelik jant özelliği ekleyiniz.

```
class ArabaSinifim(Araba):
    çelikJant=True
    def gazVer(self):
        print (self.marka, 'Durum: Gaza basıldı')

yeniAraba=ArabaSinifim('Marka 456', 1992)
yeniAraba.gazVer()
yeniAraba.özellikleri()
```

Doğru Cevap

Teşekkürler