

## Python 3 – Nesne Yönelimli Python

### Programlama Paradigmaları (kısaca özet)

- «**Fonksiyonel programlama (FP)**», hesaplamayı matematiksel işlevlerin değerlendirilmesi olarak ele alır ve değişen durum ve değişken veriden kaçınır. Kaynak: [https://en.wikipedia.org/wiki/Functional\\_programming](https://en.wikipedia.org/wiki/Functional_programming), Erişim: Haziran 2022.
- «**Nesne yönelimli programlama (OOP)**», genellikle nitelikler olarak bilinen alanlar biçiminde veri içerebilen “nesneler” kavramına dayanan bir programlama paradigmasıdır; ve genellikle **metot** olarak bilinen prosedürler biçiminde kodlar. Kaynak: [https://en.wikipedia.org/wiki/Object-oriented\\_programming](https://en.wikipedia.org/wiki/Object-oriented_programming), Erişim: Haziran 2022.
- «**Yapısal (Prosedürel) programlama**», prosedür çağrısı kavramına dayalı olarak yapılandırılmış programlamadan türetilen bir programlama paradigmasıdır. Rutinler, alt rutinler veya fonksiyonlar olarak da bilinen prosedürler, basitçe gerçekleştirilecek bir dizi hesaplama adımını içerir. Kaynak: [https://en.wikipedia.org/wiki/Procedural\\_programming](https://en.wikipedia.org/wiki/Procedural_programming), Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

261

261

## Python 3 – Nesne Yönelimli Python

- Python, genel amaçlı programlama için tasarlanmış üst düzey bir programlama dili olduğu için **hem Nesne Yönelimli hem de Yapısal Programlamayı destekler**. «Çoklu paradigma» kavramıyla kastedilen budur.
- Bu bölüme kadar Yapısal Programlamayı öğrendik.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

262

262

## Python 3 – Nesne Yönelimli Python

- Nesne yönelimli programlamanın yapı taşları şunları içerir:
  - **Class (Sınıf)**: bireysel nesneler, nitelikler ve yöntemler için şablon görevi gören kullanıcı tanımlı veri türleridir.
  - **Object (Nesne)**: özel olarak tanımlanmış veri ile oluşturulan bir sınıfın örnekleridir. Nesneler, gerçek dünyadaki nesnelere veya soyut bir varlığa karşılık gelebilir.
  - **Method (Metot/Yöntem)**: bir nesnenin davranışlarını tanımlayan bir sınıf içinde tanımlanan fonksiyonlardır. Sınıf tanımlarında yer alan her metot, bir örnek nesneye başvuruyla başlar. Ek olarak, bir nesnede bulunan alt rutinlere örnek metotlar denir.
  - **Attribute (Nitelik, Özellik)**: sınıf şablonunda tanımlanır ve bir nesnenin durumunu temsil eder. Nesneler, nitelikler alanında depolanan veriye sahip olacaktır. Sınıf nitelikleri sınıfın kendisine aittir.

Kaynak: <https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

263

263

## Python 3 – Nesne Yönelimli Python

- Nesne yönelimli programlamanın yapısı veya yapı taşları şunları içerir:
- Örnek: [https://hub-courses.pages.pasteur.fr/python\\_one\\_week\\_4\\_biotologists\\_solutions/Object\\_Oriented\\_Programming.html](https://hub-courses.pages.pasteur.fr/python_one_week_4_biotologists_solutions/Object_Oriented_Programming.html), Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

264

264

## Python 3 – Nesne Yönelimli Python

Nesne yönelimli programlama aşağıdaki ilkelere dayanmaktadır:

- **Encapsulation (Kapsülleme)**: Bu ilke, tüm önemli bilgilerin bir nesnenin içinde bulunduğunu ve yalnızca belirli bilgilerin açığa çıktığını belirtir. Her nesnenin uygulaması ve durumu, tanımlanmış bir sınıf içinde özel olarak tutulur. Diğer nesnelerin bu sınıfa erişimi veya değişiklik yapma yetkisi yoktur. Yalnızca genel işlevlerin veya yöntemlerin bir listesini çağırabilirler. Veri gizlemenin bu özelliği, daha fazla program güvenliği sağlar ve istenmeyen veri bozulmalarını önler.
- **Abstraction (Soyutlama)**: Nesneler, yalnızca diğer nesnelerin kullanımıyla ilgili iç mekanizmaları ortaya çıkararak gereksiz uygulama kodlarını gizler. Türetilmiş sınıfın işlevselliği genişletilmiş olabilir. Bu konsept, geliştiricilerin zaman içinde ek değişiklikler veya eklemeler yapmasına daha kolay yardımcı olabilir.
- **Inheritance (Kalıtım)**: Sınıflar, diğer sınıflardan gelen kodları yeniden kullanabilir. Nesneler arasındaki ilişkiler ve alt sınıflar atanabilir, bu da geliştiricilerin benzersiz bir hiyerarşiyi korurken ortak mantığı yeniden kullanmalarını sağlar. OOP'nin bu özelliği, daha kapsamlı bir veri analizini zorlar, geliştirme süresini azaltır ve daha yüksek düzeyde doğruluk sağlar.
- **Polymorphism (Polimorfizm)**: Nesneler, davranışları paylaşmak üzere tasarlanmıştır ve birden fazla biçim alabilirler. Program, bir üst sınıftan o nesnenin her yürütülmesi için hangi anlamın veya kullanımın gerekli olduğunu belirleyerek, kodu çoğaltma ihtiyacını azaltacaktır. Daha sonra, üst sınıfın işlevselliğini genişleten bir alt sınıf oluşturulur. Polimorfizm, farklı türdeki nesnelerin aynı arayüzden geçmesine izin verir.

Kaynak: <https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

265

265

## Python 3 – class

- Örnek: x adlı bir özelliğe (nitelik) sahip MyClass adlı bir sınıf:

```
class MyClass:
    x = 5

p1 = MyClass()
print(p1.x)
```

5

- Tüm sınıfların, sınıf başlatılırken her zaman yürütülen `__init__()` adlı bir fonksiyonu vardır (constructor). Yukarıda olduğu gibi, `__init__` belirtilmezse varsayılan olarak boştur.
- Nesne özelliklerine veya nesne oluşturulurken yapılması gereken diğer işlemlere değer atamak için `__init__()` işlevini kullanın:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

John  
36

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

266

266

## Python 3 – class

- **self** parametresi, sınıfın mevcut örneğine bir referanstır ve sınıfa ait değişkenlere erişmek için kullanılır. Sınıftaki herhangi bir fonksiyonun ilk parametresi olmalıdır:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def myfunc(self):
        print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Hello my name is John

```
# Nesnenin bir özelliğini değiştirmek
p1.age = 40
print(p1.age)
```

40

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

267

267

## Python 3 – class: Kalıtım (Inheritance)

### Python Kalıtımı

- Kalıtım, tüm yöntemleri ve özellikleri başka bir sınıftan miras alan bir sınıf tanımlamamızı sağlar.
- **Parent** sınıf, temel sınıf olarak da adlandırılan, miras alınan sınıftır.
- **Child** sınıf, türetilmiş sınıf olarak da adlandırılan başka bir sınıftan miras alan sınıftır.

PARENT

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname
    def printname(self):
        print(self.firstname, self.lastname)

x = Person("John", "Doe")
x.printname()
```

John Doe

CHILD

```
class Student(Person):
    pass
```

```
x = Student("Mike", "Olsen")
x.printname()
```

Mike Olsen

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

268

268

## Python 3 – class: Kalıtım (Inheritance)

### Python Kalıtımı

- \* `__init__()` fonksiyonunu eklediğinizde, alt sınıf artık ebeveynin `__init__()` işlevini devralmaz.
- Yani, child sınıfın `__init__()` fonksiyonu, parent'ın `__init__()` işlevini **override** eder.
- Child sınıfa, parent sınıftaki bir fonksiyonla aynı ada sahip bir fonksiyon eklerseniz, parent'tan kalıtımla gelen fonksiyon **override** olur.
- Python ayrıca, alt sınıfın ebeveyninden tüm metotları ve özellikleri miras almasını sağlayacak bir **super()** fonksiyonuna sahiptir.
- `super()` fonksiyonunu kullanarak, parent öğenin adını kullanmanız gerekmez, parent ögesinden yöntemleri ve özellikleri otomatik olarak devralınır.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

269

269

## Python 3 – class: Kalıtım (Inheritance)

### Python Kalıtımı

```
class Person:
    * def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

class Student(Person):
    * def __init__(self, fname, lname, year):
        super().__init__(fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

x = Student("Mike", "Olsen", 2019)
x.welcome()
```

```
Welcome Mike Olsen to the class of 2019
```

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

270

270


# Python 3 – class: Kalıtım (Inheritance)

Python Kalıtımı: 5 Tip Kalıtım

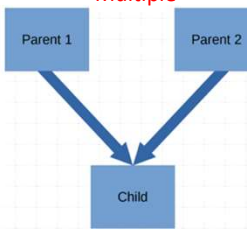
Single inheritance, Multiple inheritance, Multilevel inheritance

Hierarchical inheritance, Hybrid inheritance

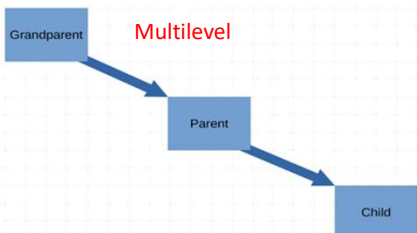
**Single**



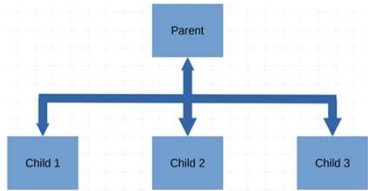
**Multiple**



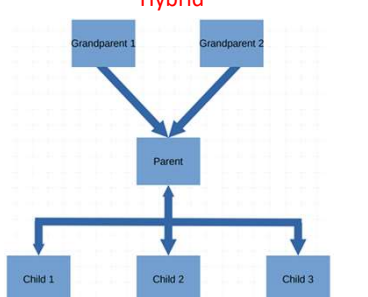
**Multilevel**



**Hierarchical**



**Hybrid**



29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

271

# Python 3 – class: Kalıtım (Inheritance)

Python Kalıtımı: 5 Tip Kalıtım

Single inheritance

PyCharm ipuçları

is subclassed by: Fish

Overrides method in: Animal

is overriddden in: Fish

is overriddden in: Fish

```
1 class Animal:
2     def __init__(self):
3         self.num_eyes = 2
4
5     def breathe(self):
6         print("inhale, exhale.")
7
8
9 class Fish(Animal):
10     def __init__(self):
11         super().__init__()
12
13     def breathe(self):
14         super().breathe()
15         print("doing this underwater.")
16
17     def swim(self):
18         print("moving in water.")
19
20
21 nemo = Fish()
22 nemo.breathe()
```

29.06.2022

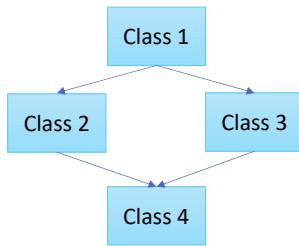
Python Eğitimi - Dr. Fatma GÜMÜŞ

272

272

## Python 3 – class: Kalıtım (Inheritance)

- Python Kalıtımı: 5 Tip Kalıtım
- Multiple inheritance
  - Diamond Problem



**Diamond Problem:** Class2 ve Class3'den iki sınıfı bir üst sınıf Class1'dan miras aldığı ve Sınıf4 sınıfı hem Class2 hem de Class3'den miras aldığı ortaya çıkan bir belirsizliği ifade eder.

Class2 ve Class3'den birinde veya her ikisinde de geçersiz kılınan (override) bir metod olan bir "m" metodu varsa, Class4'ün hangisini devralması gerektiği hakkında belirsizlik ortaya çıkar.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

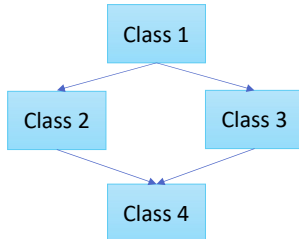
Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

273

273

## Python 3 – class: Kalıtım (Inheritance)

- Python Kalıtımı: 5 Tip Kalıtım
- Multiple inheritance
  - Diamond Problem: Case 1) Her iki sınıfta da metod override edildiğinde



```

4 class Class1:
5     def m(self):
6         print("In Class1")
7
8
9 class Class2(Class1):
10     def m(self):
11         print("In Class2")
12
13
14 class Class3(Class1):
15     def m(self):
16         print("In Class3")
17
18
19 class Class4(Class2, Class3):
20     pass
21
22
23 obj = Class4()
24 obj.m()
  
```

Çıktı?

obj.m()'yi çağırdığınızda çıktı «In Class2» olur.  
Class4, **Class4(Class3, Class2)** olarak bildirilirse, obj.m()'nin çıktısı «In Class3» olacaktır.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

274

274

## Python 3 – class: Kalıtım (Inheritance)

Python Kalıtımı: 5 Tip Kalıtım

Multiple inheritance

- Diamond Problem: Case 2) Sınıflardan birinde metod override edildiğinde

```
class Class1:
    def m(self):
        print("In Class1")

class Class2(Class1):
    pass

class Class3(Class1):
    def m(self):
        print("In Class3")

class Class4(Class2, Class3):
    pass

obj = Class4()
obj.m()
```

Çıktı?  
obj.m()'yi çağırdığınızda çıktı «In Class3» olur.

29.06.2022 Python Eğitimi - Dr. Fatma GÜMÜŞ  
Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

275

## Python 3 – class: Kalıtım (Inheritance)

Python Kalıtımı: 5 Tip Kalıtım

Multiple inheritance

- Diamond Problem: Case 3) Her sınıf aynı metodu override ederse

```
class Class1:
    def m(self):
        print("In Class1")

class Class2(Class1):
    def m(self):
        print("In Class2")

class Class3(Class1):
    def m(self):
        print("In Class3")

class Class4(Class2, Class3):
    def m(self):
        print("In Class4")

obj = Class4()
obj.m()

Class2.m(obj)
Class3.m(obj)
Class1.m(obj)
```

In Class4  
In Class2  
In Class3  
In Class1

29.06.2022 Python Eğitimi - Dr. Fatma GÜMÜŞ  
Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

276



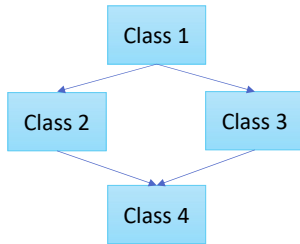
## Python 3 – class: Kalıtım (Inh

### Python Kalıtımı: 5 Tip Kalıtım



Multiple inheritance

- **Diamond Problem:** Case 4) Class4'ün m metodu içinde; Class1, Class2, Class3 için m metodunu çağırmaya çalıştığımızda



In Class4  
In Class2  
In Class3  
In Class1

```

5 class Class1:
6     def m(self):
7         print("In Class1")
8
9
10 class Class2(Class1):
11     def m(self):
12         print("In Class2")
13
14
15 class Class3(Class1):
16     def m(self):
17         print("In Class3")
18
19
20 class Class4(Class2, Class3):
21     def m(self):
22         print("In Class4")
23         Class2.m(self)
24         Class3.m(self)
25         Class1.m(self)
26
27
28 obj = Class4()
29 obj.m()
  
```

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

277

Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

277

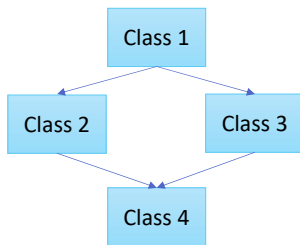
## Python 3 – class: Kalıtım (Inhe

### Python Kalıtımı: 5 Tip Kalıtım



Multiple inheritance

- **Diamond Problem:** Case 5) , Sınıf2'de hem de Sınıf3'te override edildikten sonra Sınıf1'i çağırmaya çalıştığımızda



In Class4  
In Class2  
In Class1  
In Class3  
In Class1

Sorun: Sınıf1'in m yöntemi iki kez  
çağrılır.  
Python, super() fonksiyonu  
yardımıyla yukarıdaki soruna bir  
çözüm sunar.

```

4 class Class1:
5     def m(self):
6         print("In Class1")
7
8
9 class Class2(Class1):
10     def m(self):
11         print("In Class2")
12         Class1.m(self)
13
14
15 class Class3(Class1):
16     def m(self):
17         print("In Class3")
18         Class1.m(self)
19
20
21 class Class4(Class2, Class3):
22     def m(self):
23         print("In Class4")
24         Class2.m(self)
25         Class3.m(self)
26
27
28 obj = Class4()
29 obj.m()
  
```

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

278

Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

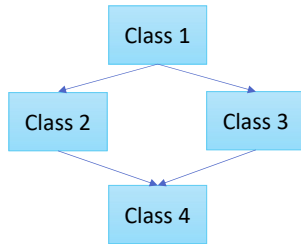
278

## Python 3 – class: Kalıtım (Inh

### Python Kalıtımı: 5 Tip Kalıtım

#### Multiple inheritance

- Diamond Problem: Case 6) super()



Super(), örnekler başlatıldığında genellikle \_\_init\_\_ fonksiyonuyla birlikte kullanılır. Süper fonksiyonu, metod çözümleme sırası (method resolution order, MRO) yardımıyla hangi metodun çağrılacağı konusunda bir sonuca varır.

In Class4  
In Class2  
In Class3  
In Class1

```

3 class Class1:
4     def m(self):
5         print("In Class1")
6
7
8 class Class2(Class1):
9     def m(self):
10        print("In Class2")
11        super().m()
12
13
14 class Class3(Class1):
15     def m(self):
16        print("In Class3")
17        super().m()
18
19
20 class Class4(Class2, Class3):
21     def m(self):
22        print("In Class4")
23        super().m()
24
25
26 obj = Class4()
27 obj.m()
  
```

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

279

Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

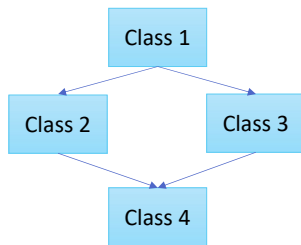
279

## Python 3 – class: Kalıtım (Inheritance)

### Python Kalıtımı: 5 Tip Kalıtım

#### Multiple inheritance

- Diamond Problem: MRO



Python'da, yerleşik veya kullanıcı tanımlı her sınıf, **Object** sınıfından türetilir ve tüm nesneler, **Object** örnekleridir. Bu nedenle, **Object** sınıfı, diğer tüm sınıflar için temel sınıftır.

Çoklu kalıtım durumunda, belirli bir öznitelik bulunamazsa önce mevcut sınıfta aranır, ardından üst sınıflarda aranır. Ebeveyn sınıflar sol-sağ şeklinde aranır ve her sınıf bir kez aranır.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

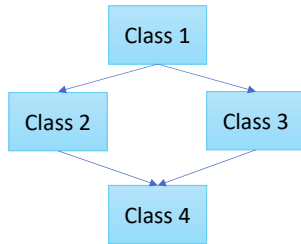
280

Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

280

## Python 3 – class: Kalıtım (Inheritance)

Python Kalıtımı: 5 Tip Kalıtım  
Multiple inheritance  
• Diamond Problem: MRO



Önceki örnekte, öznelikler arama sırası:

- Derived (Türeyen, Class4),
  - Base1 (Class2),
  - Base2 (Class3),
  - Object
- olacaktır.

İzlenen sıra, Derived sınıfının doğrusallaştırılması olarak bilinir ve bu sıra, Yöntem Çözünürlük Sırası (MRO) adı verilen bir dizi kural kullanılarak bulunur.

In Class4  
In Class2  
In Class3  
In Class1

```

3 class Class1:
4     def m(self):
5         print("In Class1")
6
7
8 class Class2(Class1):
9     def m(self):
10        print("In Class2")
11        super().m()
12
13
14 class Class3(Class1):
15     def m(self):
16        print("In Class3")
17        super().m()
18
19
20 class Class4(Class2, Class3):
21     def m(self):
22        print("In Class4")
23        super().m()
24
25
26 obj = Class4()
27 obj.m()
  
```

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

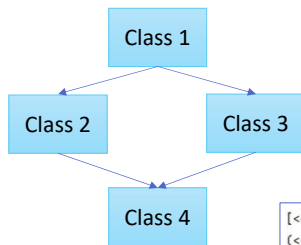
Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

281

281

## Python 3 – class: Kalıtım (Inheritance)

Python Kalıtımı: 5 Tip Kalıtım  
Multiple inheritance  
• Diamond Problem: MRO



Bir sınıfın MRO'sunu görüntülemek için:

- mro() metodu, bir liste döndürür
  - Örn: Class4.mro()
- \_mro\_ niteliği, bir tuple döndürür
  - Örn: Class4.\_mro\_

```

[<class '__main__.Class4'>, <class '__main__.Class2'>, <class '__main__.Class3'>, <class '__main__.Class1'>, <class 'object'>]
(<class '__main__.Class4'>, <class '__main__.Class2'>, <class '__main__.Class3'>, <class '__main__.Class1'>, <class 'object'>)
  
```

```

3 class Class1:
4     def m(self):
5         print("In Class1")
6
7
8 class Class2(Class1):
9     def m(self):
10        print("In Class2")
11        super().m()
12
13
14 class Class3(Class1):
15     def m(self):
16        print("In Class3")
17        super().m()
18
19
20 class Class4(Class2, Class3):
21     def m(self):
22        print("In Class4")
23        super().m()
24
25
26 print(Class4.mro()) # print list
27 print(Class4._mro_) # print tuple
  
```

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

Kaynak: <https://www.geeksforgeeks.org/multiple-inheritance-in-python/>, Haziran 2022.

282

282

## Python 3 – class: Encapsulation

Kapsülleme, verinin ve bu veri üzerinde çalışan fonksiyonların tek bir nesne içinde paketlenmesidir. Bunu yaparak, nesnenin iç durumunu dışarıdan gizleyebilirsiniz. Bu bilgi gizleme olarak bilinir.

Sınıf, bir kapsülleme örneğidir. Bir sınıf, veriyi ve metotları tek bir birimde toplar.

Bilgi gizleme fikri, dışarıdan görünmeyen bir niteliğiniz varsa, nesnenizin her zaman geçerli bir duruma sahip olduğundan emin olmak için değerine erişimi kontrol edebilirsiniz.

Kaynak: <https://www.pythontutorial.net/python-oop/python-private-attributes/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

283

283

## Python 3 – class: Encapsulation

```
class Counter:
    def __init__(self):
        self.current = 0

    def increment(self):
        self.current += 1

    def value(self):
        return self.current

    def reset(self):
        self.current = 0
```

```
counter = Counter()

counter.increment()
counter.increment()
counter.increment()

print(counter.value())
```

Output:

3

Kod çalışıyor ama bir sorunu var: Counter sınıfının dışından yine de mevcut özneliğe erişebilir ve onu istediğiniz gibi değiştirebilirsiniz.

Örneğin:

```
counter = Counter()

counter.increment()
counter.increment()
counter.current = -999

print(counter.value())
```

Output:

-999

Bu örnekte, Counter sınıfının bir örneğini oluşturuyoruz, increment() yöntemini iki kez çağırıyoruz ve mevcut özneliğin değerini geçersiz bir -999 değerine ayarladık. Peki, mevcut özneliğin Counter sınıfının dışında değişmesini nasıl engellersiniz?

Kaynak: <https://www.pythontutorial.net/python-oop/python-private-attributes/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

284

284

## Python 3 – class: Encapsulation

### Private attributes (Özel nitelikler)

**private** niteliklere yalnızca sınıfın metotları ile erişilebilir. Başka bir deyişle, sınıf dışından erişilemezler.

**Python'un private nitelik kavramı yoktur.** Diğer bir deyişle, tüm niteliklere bir sınıfın dışından erişilebilir.

Geleneksel olarak, tek bir alt çizgi (\_) öneki ekleyerek özel bir öznelik tanımlayabilirsiniz:

`_attribute`

Bu, `_attribute`'in manipüle edilmemesi gerektiği ve gelecekte bir kırılma değişikliği olabileceği anlamına gelir.

Kaynak: <https://www.pythontutorial.net/python-oop/python-private-attributes/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

285

```
class Counter:
    def __init__(self):
        self._current = 0

    def increment(self):
        self._current += 1

    def value(self):
        return self._current

    def reset(self):
        self._current = 0
```

285

## Python 3 – class: Encapsulation

### Private attributes (Özel nitelikler)

**private** niteliklere yalnızca sınıfın metotları ile erişilebilir. Başka bir deyişle, sınıf dışından erişilemezler.

**Python'un private nitelik kavramı yoktur.** Diğer bir deyişle, tüm niteliklere bir sınıfın dışından erişilebilir.

Geleneksel olarak, tek bir alt çizgi (\_) öneki ekleyerek özel bir öznelik tanımlayabilirsiniz:

`_attribute`

Bu, `_attribute`'in manipüle edilmemesi gerektiği ve gelecekte bir kırılma değişikliği olabileceği anlamına gelir.

Kaynak: <https://www.pythontutorial.net/python-oop/python-private-attributes/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

286

```
class Counter:
    def __init__(self):
        self._current = 0

    def increment(self):
        self._current += 1

    def value(self):
        return self._current

    def reset(self):
        self._current = 0
```

286

## Python 3 – class: Encapsulation

### Private attributes (Özel nitelikler)

Bir öznelik adının önüne aşağıdaki gibi çift alt çizgi (\_\_) eklerseniz, Python \_\_attribute adını otomatik olarak \_\_class\_\_attribute olarak değiştirir:

`__attribute`

Buna Python'da **name mangling** denir.

Bunu yaparak, \_\_attribute öğesine, örneğin: `instance.__attribute` gibi bir sınıfın dışından doğrudan erişemezsiniz.

Ancak yine de buna `__class__attribute` adını kullanarak erişebilirsiniz:

`instance.__class__attribute`

```
class Counter:
    def __init__(self):
        self.__current = 0

    def increment(self):
        self.__current += 1

    def value(self):
        return self.__current

    def reset(self):
        self.__current = 0
```

```
counter = Counter()
print(counter.__current)
```

Output:

```
AttributeError: 'Counter' object has no attribute '__current'
```

Ancak, şu şekilde erişebilirsiniz:

```
counter = Counter()
print(counter._Counter__current)
```

Kaynak: <https://www.pythontutorial.net/python-oop/python-private-attributes/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

287

287

## Python 3 – class: Polymorphism

Python'da polimorfizmi kullanmanın farklı yöntemleri vardır. Polimorfizmi tanımlamak için farklı fonksiyon, sınıf metotları veya nesneler kullanabilirsiniz.

### 1- Fonksiyon ve Nesnelerle Polimorfizm

Herhangi bir nesneyi alabilen, polimorfizme olanak tanıyan bir fonksiyon oluşturabilirsiniz.

```
class Tomato():
    def type(self):
        print("Vegetable")
    def color(self):
        print("Red")
class Apple():
    def type(self):
        print("Fruit")
    def color(self):
        print("Red")

def func(obj):
    obj.type()
    obj.color()

obj_tomato = Tomato()
obj_apple = Apple()
func(obj_tomato)
func(obj_apple)
```

Output:

```
Vegetable
Red
Fruit
Red
```

Kaynak: <https://www.edureka.co/blog/polymorphism-in-python/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

288

288

## Python 3 – class: Polymorphism

Python'da polimorfizmi kullanmanın farklı yöntemleri vardır. Polimorfizmi tanımlamak için farklı fonksiyon, sınıf metotları veya nesneler kullanabilirsiniz.

### 2- Sınıf Metotları ile Polimorfizm

Python aynı şekilde iki farklı sınıf tipi kullanır. Burada, bir grup nesne boyunca yinelenen bir **for** döngüsü oluşturmanız gerekir. Ardından, her bir nesnenin hangi sınıf türü olduğuyla ilgilenmeden metotları çağırmanız gerekir. Bu metotları aslında her sınıfta var olduğunu varsayıyoruz.

```
class India():
    def capital(self):
        print("New Delhi")

    def language(self):
        print("Hindi and English")

class USA():
    def capital(self):
        print("Washington, D.C.")

    def language(self):
        print("English")

obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):
    country.capital()
    country.language()
```

```
New Delhi
Hindi and English
Washington, D.C.
English
```

Kaynak: <https://www.edureka.co/blog/polymorphism-in-python/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

289

289

## Python 3 – class: Polymorphism

Python'da polimorfizmi kullanmanın farklı yöntemleri vardır. Polimorfizmi tanımlamak için farklı fonksiyon, sınıf metotları veya nesneler kullanabilirsiniz.

### 3- Kalıtım ile Polimorfizm

Bu, çoğunlukla üst sınıftan miras alınan metodun alt sınıfa uymadığı durumlarda kullanılır. Alt sınıfta bir metodu yeniden uygulama işlemi, metot **override** olarak bilinir.

```
class Bird:
    def intro(self):
        print("There are different types of birds")

    def flight(self):
        print("Most of the birds can fly but some cannot")

class parrot(Bird):
    def flight(self):
        print("Parrots can fly")

class penguin(Bird):
    def flight(self):
        print("Penguins do not fly")

obj_bird = Bird()
obj_parr = parrot()
obj_peng = penguin()

obj_bird.intro()
obj_bird.flight()

obj_parr.intro()
obj_parr.flight()

obj_peng.intro()
obj_peng.flight()
```

```
There are different types of birds
Most of the birds can fly but some cannot
There are different types of bird
Parrots can fly
There are many types of birds
Penguins do not fly
```

Kaynak: <https://www.edureka.co/blog/polymorphism-in-python/>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

290

290

## Python 3 – class: Abstraction

Python'da soyut sınıflar (abstract class) ve interface kullanılarak soyutlama yapılabilir.

Bir veya daha fazla soyut metottan oluşan **abstract class** sınıf denir.

Soyut metotlar implementasyon içermez.

Soyut sınıf, alt sınıf tarafından miras alınabilir ve soyut metot, tanımını alt sınıfta alır.

Soyutlama sınıfları, diğer sınıfın planı olmak içindir.

Soyut bir sınıf, büyük fonksiyonlar tasarlarlarken faydalı olabilir. Soyut bir sınıf, bileşenlerin farklı uygulamaları için standart arabirim sağlamak için de yararlıdır.

Kaynak: <https://www.javatpoint.com/abstraction-in-python>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

291

291

## Python 3 – class: Abstraction

Python programında soyutlamayı kullanmak için **abc** modülü kullanılır.

```
from abc import ABC
class ClassName(ABC):
```

Kaynak: <https://www.javatpoint.com/abstraction-in-python>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

292

292



## Python 3 – class: Abstraction

Python programında soyutlamayı kullanmak için `abc` modülü kullanılır.

zorunlu  
değil

```
from abc import ABC, abstractmethod

class Car(ABC):
    @abstractmethod
    def mileage(self):
        pass

class Tesla(Car):
    def mileage(self):
        print("The mileage is 30kmph")

class Suzuki(Car):
    def mileage(self):
        print("The mileage is 25kmph ")

class Duster(Car):
    def mileage(self):
        print("The mileage is 24kmph ")

class Renault(Car):
    def mileage(self):
        print("The mileage is 27kmph ")
```

Kaynak: <https://www.geogebra.org/m/abc>

29.06.2022

Python Eğitimi - Dr. Fatma Gümü

293

```
# Driver code

t= Tesla ()
t.mileage()

r = Renault ()

r.mileage()

s = Suzuki ()
s.mileage()

d = Duster ()
d.mileage()
```

```
The mileage is 30kmph
The mileage is 27kmph
The mileage is 25kmph
The mileage is 24kmph
```

293

## Python 3 – class: Abstraction

Python programında soyutlamayı kullanmak için `abc` modülü kullanılır.

```
from abc import ABC

class Polygon(ABC):

    @abstractmethod
    def sides(self):
        pass

class Triangle(Polygon):
    def sides(self):
        print("Triangle has 3 sides")

class Pentagon(Polygon):
    def sides(self):
        print("Pentagon has 5 sides")

class Hexagon(Polygon):
    def sides(self):
        print("Hexagon has 6 sides")

class square(Polygon):
    def sides(self):
        print("I have 4 sides")
```

Kaynak: <https://www.geogebra.org/m/abc>

29.06.2022

Python Eğitimi - Dr. Fatma Gümü

294

```
t = Triangle()
t.sides()

s = square()
s.sides()

p = Pentagon()
p.sides()

k = Hexagon()
k.sides()
```

```
Triangle has 3 sides
Square has 4 sides
Pentagon has 5 sides
Hexagon has 6 sides
```

294

## Python 3 – class: Abstraction

Python programında soyutlamayı kullanmak için `abc` modülü kullanılır.

Özet:

- Bir Soyut sınıf, hem normal hem de abstract metot içerebilir.
- Bir soyut sınıf için nesne oluşturulamaz.

Kaynak: <https://www.javatpoint.com/abstraction-in-python>, Erişim: Haziran 2022.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

295

295

## Python 3 – Nesne Yönelimli Python

### Alıştırma Ödevi

- Python standart kütüphanesinde yer alan `turtle` modülünü inceleyin: <https://docs.python.org/3/library/turtle.html>, Erişim: Haziran 2022.
- Başlangıç kodunu kullanın, her satırın görevini anlamak için yukarıdaki dokümantasyona başvurun.
- Gereksinimler:
  - Kaplumbağanın şeklini (shape) değiştirin.
  - Kaplumbağanın rengini (color) değiştirin.
  - Kaplumbağanın 100 piksel ilerlemesini (forward) sağlayın.
  - Kaplumbağanın ilerleme hızını (speed) görüntüleyin ve yavaşlatın.
  - `for` döngüsü kullanarak «kare» çizdirin. Her bir kenar bittiğinde ekran arkaplan rengi değişsin.

29.06.2022

Python Eğitimi - Dr. Fatma GÜMÜŞ

296

296

# Python 3 – Nesne Yönelimli Python

## Alıştırma Ödevi

- `prettytable` paketi kurulu değilse, kurun.
- Dokümantasyonu inceleyin: <https://pypi.org/project/prettytable/>, Erişim: Haziran 2022.
- Aşağıdaki çıktıyı, bu kütüphaneyi kullanarak elde edin:

```
+-----+-----+
| Pokemon name | Type   |
+-----+-----+
| Pikachu      | Elektrik |
| Charmander   | Fire     |
| Squirtle     | Water    |
+-----+-----+
```