**Python Nedir?**

Python, [nesne yönelimli](https://tr.wikipedia.org/wiki/Nesne_y%C3%B6nelimli), [yorumlamalı](https://tr.wikipedia.org/wiki/Yorumlanan_programlama_dili), birimsel (modüler) ve etkileşimli [yüksek seviyeli](https://tr.wikipedia.org/w/index.php?title=Y%C3%BCksek_seviyeli_programlama_dili&action=edit&redlink=1) bir [programlama dilidir](https://tr.wikipedia.org/wiki/Programlama_dilleri).

**Tarihçe**

Geliştirilmeye 1990 yılında [Guido van Rossum](https://tr.wikipedia.org/wiki/Guido_van_Rossum" \o "Guido van Rossum) tarafından [Amsterdam](https://tr.wikipedia.org/wiki/Amsterdam)'da başlanmıştır. Adını sanılanın aksine bir yılandan değil Guido van Rossum’un çok sevdiği, [Monty Python](https://tr.wikipedia.org/wiki/Monty_Python" \o "Monty Python) adlı altı kişilik bir İngiliz komedi grubunun Monty Python’s Flying Circus adlı gösterisinden almıştır. Günümüzde [Python Yazılım Vakfı](https://tr.wikipedia.org/wiki/Python_Yaz%C4%B1l%C4%B1m_Vakf%C4%B1) çevresinde toplanan gönüllülerin çabalarıyla sürdürülmektedir.

**Python hangi amaçlar için kullanılır?**

Python yazılım dili; veri analizi, veri manipülasyonu, dijital oyun geliştirme, yapay zeka uygulamaları gibi alanlarda kullanılan, kullanımı basit, nesne yönelimli bir programlama dilidir.

**Hangi Tür Uygulamalar İçin Uygundur?**

Modüler yapısı, sınıf dizgesini (sistem) ve her türlü veri alanı girişini destekler. Hemen hemen her türlü platformda çalışabilir ([Unix](https://tr.wikipedia.org/wiki/Unix), [Linux](https://tr.wikipedia.org/wiki/Linux), [Mac](https://tr.wikipedia.org/wiki/Mac), [Windows](https://tr.wikipedia.org/wiki/Microsoft_Windows), [Amiga](https://tr.wikipedia.org/wiki/Amiga" \o "Amiga), [Symbian](https://tr.wikipedia.org/wiki/Symbian" \o "Symbian)).

**Python Koşul (If ... Else)**

Python, matematiğin olağan mantıksal koşullarını destekler:

* Eşittir: a == b
* Eşit Değil: a != b
* Daha az: a < b
* Küçük veya eşittir: a <= b
* Büyüktür: a > b
* Büyük veya eşittir: a >= b

***If ifadesi:***

a = 33

b = 200

if b > a:

print("b, a'dan büyüktür ")

***\*Not****: if ifadesi, girintisiz (bir hataya neden olur)*

***Else :***

a = 200

b = 33

if b > a:

print(“b, a'dan büyüktür”)

else:

print(“b, a'dan büyük değil")

**Python Döngüleri**

Python’un iki ilkel döngü komutu vardır:

* while döngüler
* for döngüler

**While**

While döngüsü ile, bir koşul doğru olduğu sürece bir dizi ifade çalıştırabiliriz.

*İ 6’dan küçük olduğu sürece i yazdır:*

i = 1

while i < 6:

print(i)

i += 1

***Break:***

Break deyimi ile while koşulu doğru olsa bile döngüyü durdurabiliriz:

i 3 olduğunda döngüden çık:

i = 1

while i < 6:

print(i)

if i == 3:

break

i += 1

***continue:***

continue ile mevcut yinelemeyi durdurabilir ve bir sonraki ile devam edebiliriz.

İ 3 ise bir sonraki yinelemeye devam edin:

i = 0

while i < 6:

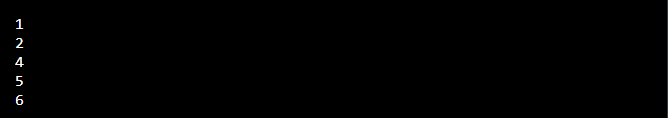
i += 1

if i == 3:

continue

print(i)

***Çıktı:***



Şekil 1 ‘3 ‘sayısını atlıyor.

***Else:***

else deyimiyle , koşul artık doğru olmadığında bir kod bloğunu bir kez çalıştırabiliriz.

Koşul yanlış olduğunda bir mesaj yazdırın:

i = 1

while i < 6:

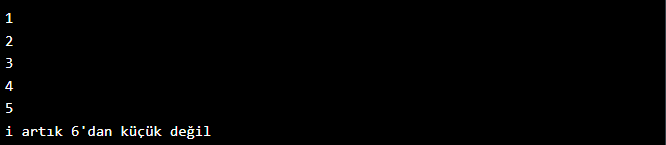
print(i)

i += 1

else:

print(“i artık 6’dan küçük değil”)

***Çıktı:***



Şekil 2 doğru olmayan kod.

**For**

Bir for döngüsü, bir dizi üzerinde yineleme yapmak için kullanılır (yani bir liste, bir demet, bir sözlük, bir küme veya bir dizedir).

Bu, diğer programlama dillerindeki for anahtar sözcüğüne daha az benzer ve diğer nesne yönelimli programlama dillerinde bulunan bir yineleme yöntemi gibi çalışır.

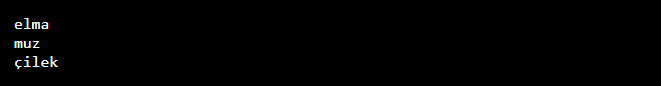
Her meyveyi bir meyve listesinde yazdırın:

meyve = [“elma”, “muz”, “çilek”]

for x in meyve:

print(x)

***Çıktı:***



Şekil 3 for döngüsü

**Bir Dizide Döngü Yapmak**

Dizeler bile yinelenebilir nesnelerdir, bir dizi karakter içerirler.

“Süleyman” kelimesindeki harfler arasında dolaşın:

for x in “Süleyman”:

print(x,end= “ “)

***Çıktı:***



Şekil 4 Yan yana ve harf harf ayrı yazıldı.

***Break İfadesi:***

Break deyimi ile, tüm öğeler arasında döngüye girmeden önce döngüyü durdurabiliriz.

x “muz” olduğunda döngüden çıkın :

fruits = [“elma”, “muz”, “çilek”]

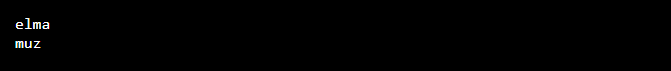
for x in fruits:

print(x)

if x == “muz”:

break

***Çıktı:***



Şekil 5 Döngüden çıkış

**range() Function**

Bir kod kümesinde belirli sayıda döngü yapmak için range() işlevini kullanabiliriz,

range() fonksiyon , varsayılan olarak 0’dan başlayan ve (varsayılan olarak) 1 artan bir sayı dizisi döndürür ve belirtilen bir sayıda biter.

Range() fonksiyonu kullanarak:

for x in range(6):

print(x,end=” ”)

***Çıktı:***



Şekil 6 range() fonksiyon

**Python ile Nesneye Yönelik Programlama**

Bu bölümde NYP (OOP)kavramları üzerinde duracak, bu kavramların Java’da nasıl gerçekleştirildiğine bakacağız.

Python Sınıflar/Nesneler

Python, nesne tabanlı bir programlama dilidir. Python’da neredeyse her şey bir sınıftır.  
  
Bir sınıf (class), kendisinden örnek oluşturulabilen bir nesnedir (object). Bu sınıflara özellik ve yöntemler atanabilir.

**Sınıf Kavaramı**

Özellikle nesne yönelimli programlama geliştirilirken en önemli kavramlardan birisi sınıf(class) olarak düşünülebilinir.

Python üzerinde sınıf kavramından bahsedeceksek özet olarak fonksiyonlar yerine yöntem(method), değişken yerine de nitelik(attribute) kullanacağınızı düşünebilirsiniz.

***Örnek:***

class Sinifim:

x = 5

**Yöntem ve Nitelik**

Python üzerinde sınıf tanımlayarak aşağıdaki gibi yöntem ve nitelikler tanımlayabilirsiniz:

class personel:

tipi=”insan”

def \_\_init\_\_(self,ad,no):

self.ad=ad

self.no=no

def adinNe(self):

return(“ad: “+self.ad)

def numaraNe(self):

return(“no: “+str(self.no))

**\_\_init\_\_ Yöntemi**

Bu tanımlamalarda en farklı gördüğümüz kısım \_\_init\_\_ isimli yöntem(method) olacaktır. Bu yöntem özetle sınıfın tanımlanması, tanımlanırken neler yapılması gerektiğini anlatır. Yani birazdan nesne oluştururken kullanılacak yöntem tam olarak burasıdır.

**Self Yapısı Nedir?**

Sınıfın içerisindeki herhangi bir nesnenin, “global” bir yapı kazanması yani bulunduğu yapının bir üst katmanında da kullanılabilmesini sağlayan yapıdır.

İnit yönteminde dışarıdan aldığımız 2 değişkeni, içeride çeşitli yöntemlerin içinde de kullanabilmek için “self” yapısı kullanılmıştır. Eğer adinNe ve numaraNe yöntemlerinde self kullanılmaadan değişkenler kullanılsaydı bu değişkenler diğer yöntemlerde kullanılamazdı.

**Nesne Yöntemleri/Metotları**

Nesneler için sınıfa özgü yöntemler/metotlar geliştirilebilir. Nesne yaratıldıktan sonra bu yöntemler .yontemAdi() şeklinde çalıştırılabilir.

Class Kisi:

def \_\_init\_\_(self, isim, yas):

self.isim = isim

self.yas = yas

def yazdir(self):

print(“Merhaba, adım “, self.isim, “, yaşım “, self.yas);

p1 = Kisi(“Murat”, 36)

p1.yazdir()

**Sınıf Kurucular (Constructors)**

[Nesne yönelimli programlamada](https://pynative.com/python/object-oriented-programming/) , yapıcı , bir [sınıfın](https://pynative.com/python-classes-and-objects/) nesnesini oluşturmak ve başlatmak için kullanılan özel bir yöntemdir . Bu metot sınıfta tanımlanmıştır.

* Yapıcı, nesne oluşturma sırasında otomatik olarak yürütülür.
* Bir kurucunun birincil kullanımı, bir sınıfın veri üyesi/ [örnek değişkenlerini](https://pynative.com/python-instance-variables/) bildirmek ve başlatmaktır . Yapıcı, bir nesnenin özniteliklerini başlatmak için nesne oluşturma sırasında yürütülen bir ifadeler koleksiyonu (yani talimatlar) içerir.

***Örnek:***

class Ogrenci:

# yapıcı

# örnek değişkenini başlat

def \_\_init\_\_(self, name):

print(‘İç Yapıcı’)

self.name = name

print(‘Tüm değişkenler başlatıldı’)

# örnek Metod

def show(self):

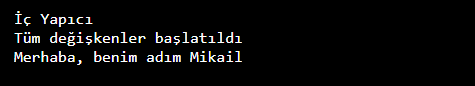
print(‘Merhaba, benim adım’, self.name)

# Oluşturucuyu kullanarak nesne oluştur

s1 = Ogrenci(‘Mikail’)

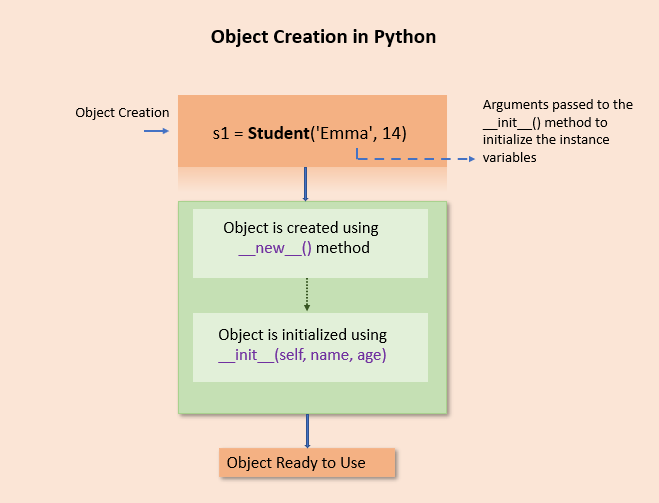
s1.show()

***Çıktı:***



Şekil 7 Kurucu örnek

* Yukarıdaki örnekte, s1yapıcı kullanılarak bir nesne yaratılmıştır.
* Bir Öğrenci nesnesi oluşturulurken , nesneyi başlatma yöntemine nameargüman olarak iletilir .\_\_init\_\_()
* Benzer şekilde, Student sınıfının çeşitli nesneleri, farklı isimler argüman olarak geçirilerek oluşturulabilir.



Şekil 8 Bir yapıcı kullanarak Python’da bir nesne oluşturun

***\*Not :***

Her nesne için yapıcı yalnızca bir kez yürütülür. Örneğin, dört nesne oluşturursak, kurucu dört kez çağrılır.

Python'da her sınıfın bir yapıcısı vardır, ancak bunu açıkça tanımlamanız gerekmez. Sınıfta kurucuları tanımlamak isteğe bağlıdır.

Python, herhangi bir kurucu tanımlanmamışsa varsayılan bir kurucu sağlayacaktır.

**Kurucu Aşırı Yükleme (Constructor Overloading)**

Yapıcı aşırı yüklemesi, aynı ada sahip ancak farklı bir bağımsız değişkene (parametre) sahip bir sınıfta birden fazla kurucu anlamına gelir. Python, Yapıcı aşırı yüklenmesini desteklemez; hiçbir işlevi yoktur.

Python'da Yöntemler yalnızca adlarıyla tanımlanır ve belirli bir ada sahip sınıf başına yalnızca bir yöntem olabilir.

***Örnek:***

class Ornek:

# arg'lere dayalı yapıcı aşırı yüklemesi

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

# eğer arg'ler arg'lerin 1 toplamından fazlaysa

if len(args) > 1:

self.answer = 0

for i in args:

self.answer += i

# arg bir tamsayının karesiyse arg

elif isinstance(args[0], int):

self.answer = args[0] \* args[0]

e1 = Ornek(1, 2, 3, 6, 8)

print("Toplam :", e1.answer)

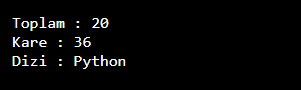
e2 = Ornek (6)

print("Kare :", e2.answer)

e3 = Ornek ("Python")

print("Dize :", e3.answer)

***Çıktı:***



Şekil 9 Constructor Overloading

**Yöntem Aşırı Yükleme (Method Overloading)**

İki veya daha fazla yöntem aynı ada ancak farklı sayıda parametreye veya farklı parametre türlerine veya her ikisine birden sahiptir. Bu yöntemlere aşırı yüklenmiş yöntemler denir.

Diğer dillerde olduğu gibi (örneğin, C++'da yöntem aşırı yüklemesi ), python da varsayılan olarak yöntem aşırı yüklemesini desteklemez. Ancak Python'da yöntem aşırı yüklemesini gerçekleştirmenin farklı yolları vardır.

***Örnek:***

def product(a, b):

p = a \* b

print(p)

def product(a, b, c):

p = a \* b\*c

print(p)

product(4, 5, 5)

***Çıktı:***



Şekil 10 Method Overloading

**Python Diziler**

Diziler, birden çok değeri tek bir değişkende depolamak için kullanılır.

Bir öğe listeniz varsa (örneğin bir araba adları listesi), arabaları tek değişkenlerde depolamak şöyle görünebilir:

cars = ["Ford", "Volvo", "BMW"]

**Dizi Metodları**

Python, listelerde/dizilerde kullanabileceğiniz bir dizi yerleşik yönteme sahiptir.

|  |  |
| --- | --- |
| Method | Description |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

Şekil 11 dizi yöntemleri

**Sarmalama/Kapsülleme (Encapsulation)**

Kapsülleme, sınıfınız içindeki özellik ve fonksiyonlara diğer sınıflar içinden yetkisiz erişimden korunması işlemidir. Bunun için erişim yetkilerini siz belirlersiniz. Bir sınıftaki özellikleri(değişkenleri) özel(private) olarak belirleyerek dış erişimden koruyabilirsiniz.

Python’da bir özelliği(değişkeni) özel(private) yapmak için başına iki tane alt çizgi ekleriz. Bunu yaptığımızda o sınıftan oluşturulan nesne ile bu özelliğe direkt olarak erişilemez.

***Örnek:***

class Personel:

def \_\_init\_\_(kisi,ad,soyad, programlamaDili, deneyimSuresi):

kisi.ad=ad

kisi.soyad=soyad

kisi.programlamaDili=programlamaDili

kisi.\_\_deneyimSuresi=deneyimSuresi

personel1 = Personel("Oben","Seven",("Python","PHP","Javascript"),18)

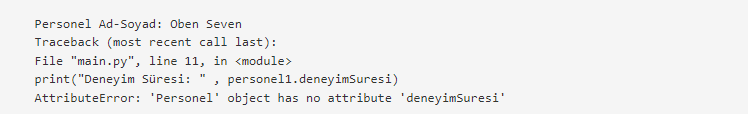
print("Personel Ad-Soyad: " + personel1.ad + " " + personel1.soyad)

print("Deneyim Süresi: " , personel1.deneyimSuresi)

print("Kullandığı Programlama Dilleri: ")

personel1.dilYazdir()

***Çıktı:***



Şekil 12 Sarmalama

Gördüğünüz gibi, deneyimSuresi verisini ekrana yazdırmak istediğimizde Personel() sınıfından oluşturulan nesnenin böyle bir özelliği olmadığı hakkında bir hata alıyoruz.

Bu veriye erişmenin tek yolu, alt çizgi ile birlikte özelliği içeren Sınıfın adını birlikte kullanmaktır;

print("Deneyim Süresi: " , personel1.\_Personel\_\_deneyimSuresi)

***Çıktı:***



Şekil 13 alt çizgi

**Kalıtım/Miras (Inheritance)**

Kalıtım, mevcut bir sınıftan yeni bir sınıf oluşturmamızı sağlar.

Oluşturulan yeni sınıf, alt sınıf (alt sınıf veya türetilmiş sınıf) olarak bilinir ve **alt** sınıfın türetildiği mevcut sınıf, **üst sınıf** (ebeveyn veya temel sınıf) olarak bilinir.

***Örnek:***

class Hayvan:

# üst sınıfın özniteliği ve yöntemi

isim = ""

def yemek(self):

print("Yiyebilirim")

# Hayvandan devral

class Köpek(Hayvan):

# alt sınıfta yeni metod

def display(self):

# kendini kullanarak üst sınıfın erişim adı özniteliği

print("Benim adım ", self.isim)

# alt sınıfın bir nesnesini oluştur

labrador = Köpek()

# üst sınıf özniteliğine ve metoduna erişin

labrador.isim = " Max"

labrador.yemek()

# alt sınıf metod çağır

labrador.display()

***Çıktı:***



Şekil 14 Bir üst sınıf Hayvandan bir alt sınıf Köpek türettik

**Erişim Belirleyicisi(protected)**

protected erişim belirleyicisi kullanılan metod yada özellik

* sınıf içinden erişilebilir,
* sınıf dışından erişilemez,
* ve türetilen sınıflardan erişilebilir.

olmaktadır.

Yine private özellikteki metod yada özelliğe sınıf dışından erişebilmek için başka bir public metod yada özellik kullanmalıdır

**Kalıtımındaki super() Metodu**

Daha önce alt sınıftaki aynı yöntemin üst sınıftaki yöntemi geçersiz kıldığını görmüştük. Ancak, alt sınıftan üst sınıf yöntemine erişmemiz gerekirse, super() yöntemini kullanırız.

Örnek:

class Hayvan:

isim = ""

def yemek(self):

print("Yiyebilirim.")

# Hayvandan devral

class Kopek(Hayvan):

# yemek() yöntemini geçersiz kıl

def yemek(self):

# super() kullanarak üst sınıfın yemek() yöntemini çağırın

super().yemek()

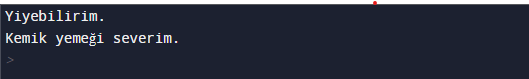
print("Kemik yemeği severim.")

# alt sınıfın bir nesnesini oluştur

labrador = Kopek()

labrador.yemek()

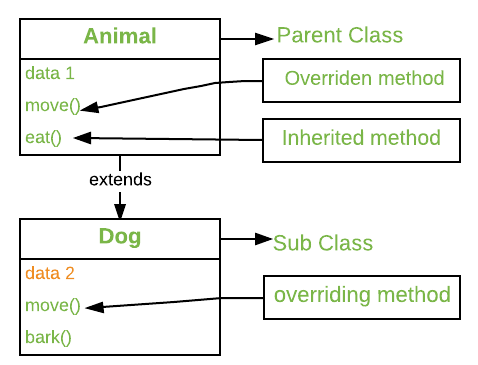
***Çıktı:***



*Şekil 15 Köpek alt sınıfının yemek() yöntemi, Hayvan üst sınıfının aynı yöntemini geçersiz kılar.*

**Yöntemi Geçersiz Kılma (Method Overriding)**

Yöntem geçersiz kılma, herhangi bir nesne yönelimli programlama dilinin, bir alt sınıfın veya alt sınıfın, üst sınıflarından veya üst sınıflarından biri tarafından zaten sağlanan bir yöntemin belirli bir uygulamasını sağlamasına izin veren bir yeteneğidir. Bir alt sınıftaki bir yöntem, super-sınıfındaki bir yöntemle aynı ada, aynı parametrelere veya imzaya ve aynı dönüş türüne (veya alt türe) sahip olduğunda, alt sınıftaki yöntemin super-sınıftaki yöntemi geçersiz kıldığı söylenir.



Şekil 16 diyagram

Geçersiz kılınan bir yöntemin hangi sürümünün yürütüleceğini belirleyen, başvurulan nesnenin türüdür (başvuru değişkeninin türü değil).

Örnek:

class Parent():

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

self.value = "Inside Parent"

def show(self):

print(self.value)

class Child(Parent):

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

self.value = "Inside Child"

def show(self):

print(self.value)

obj1 = Parent()

obj2 = Child()

obj1.show()

obj2.show()

***Çıktı:***



Şekil 17 referans değişkeninin türü değil.

**Çokbiçimlilik (Polymorphism)**

Bilindiği üzere Python, farklı sınıfların aynı ada sahip metotlara izin verdiğinden, [**sınıf**](https://1kodum.com/pythonda-sinif/) metotları oluştururken polimorfizm kavramını kullanara farklı çıktılar alınmasını sağlamak için kullanılabilir.

Yani ebeveyn sınıftan çocuk sınıfına [kalıtım](https://1kodum.com/pythonda-miras-alma/) (inheritance) yoluyla aktarılan ama çocuk sınıfında farklı bir şekilde kullanılan metotların yaptığı iş polimorfizmdir. Kısaca çok biçimlilik adı üzerine aynı metodu kullanıp farklı çıktılar almak için kullanılır.

***Örnek :***

class Personel:

def zam(self):

zam\_oran = 0.1

hesap = 100 + 100 \* zam\_oran

print("Eleman: ", hesap, "TL")

class Doktor(Personel):

def zam(self):

zam\_oran = 0.2

hesap = 100 + 100 \* zam\_oran

print("Doktor: ", hesap, "TL")

class Hakim(Personel):

def zam(self):

zam\_oran = 0.3

hesap = 100 + 100

***Çıktı:***



Şekil 18 personel

**Soyut Sınıflar (Abstract Classes)**

Her ne kadar Python varsayılan olarak soyut sınıfları sunmasa da açık kaynak kodlu bir dil olduğu için **ABC**(Abstract Base Classes) modülü ile soyut sınıfları tanımlayabiliriz.

Soyut sınıflarda tanımlanan herhangi bir soyut metot alt sınıflarda kullanılmak zorundadır. Ayrıca bazı diğer nesne tabanlı programlama dillerinden farklı olarak Python’da soyut bir yöntemin “tamamen soyut” olması gerekmez:

#Python'da Soyut Sınıflar (Abstract Classes)

from abc import ABC, abstractmethod

class Hayvan (ABC): # parent class (üst sınıf, ebeveyn sınıf)

@abstractmethod

def yurumek(self): # alt sınıfta bu metot kullanılmakt zorunda

pass

def kosmak(self): #alt sınıfta bu metot kullanılmak zorunda değil

pass

class Kus(Hayvan): # sub class (alt sınıf, çocuk sınıf)

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

print("Kuş")

def yurumek(self):

print("Yürür")

kus1 = Kus()

***Çıktı:***



Şekil 19 soyut

--------------------------------------------------------------------------------------------------------------------------------------

**Kaynakça**

https://www.sadikturan.com/python-nesne-tabanli-programlama/oop-nedir/1397

https://www.mobilhanem.com/python-nesne-tabanli-programlama-oop/

https://medium.com/@alper\_guven/pythonda-class-s%C4%B1n%C4%B1f-ve-object-nesne-yap%C4%B1s%C4%B1-c1deb10c4edc

https://mertmekatronik.com/nesne-tabanli-programlama-ve-siniflar-mert-mekatronik-youtube-python-programlama-ders-29

https://www.w3schools.com/python/python\_classes.asp

https://www.freecodecamp.org/news/object-oriented-programming-in-python/

https://pynative.com/python-constructors/

https://realpython.com/python-class-constructor/

https://www.bilgigunlugum.net/prog/python/python\_keywords

https://tutorial.eyehunts.com/python/constructor-overloading-in-python-example-code/

https://kodlamavakti.com/java/encapsulation/

https://www.programiz.com/python-programming/inheritance

https://www.geeksforgeeks.org/method-overriding-in-python/