



**T.C.
İSTANBUL MEDENİYET ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMERİ FAKÜLTESİ**

BİLGİSAYAR MÜHENDİSLİĞİ

VERİ BİLİMİNE GİRİŞ DERSİ

**DERSİN SORUMLUSU
DR. ÖGR. ÜYESİ MUHAMMET SİNAN BAŞARSLAN**

İÇİNDEKİLER

1. Python ve Giriş	2
2. Sınıflar/Classes	9
1. Nitelikler/Attributes veya properties.....	10
2. Metotlar-Fonksiyonlar	14
3. Yapıcı /Constructor/Initializer	18
3.1 OOP ile Basit Bir Hesap Makinesi Proje.....	19
4. Kapsülleme/Encapsulation.....	20
5. Miras Alma/Inheritance	22
6. Soyut Sınıf/Abstract Class	24
7. Overriding.....	25
8. Polymorphism/Çok biçimlilik.....	25
9. Jupyter	29
9.1 Online IDE	29
9.2 SciPy, Scikit-learn.....	30
10. Referanslar	30

1. Python ve Giriş

Colin McCulloch'un 2013 yılında söylediği gibi, biyolojik bilimler diğer birçok bilimsel alan gibi şu anda bir "büyük veri" devrimi yaşamaktadır [1]. En özet haliyle, insan tarafından analiz edilip anlamlandırılması çok uzun zaman alacak ya da anlamlandırılma imkânı olmayan vereleri büyük veri olarak tanımlayabiliriz. Ancak bir bilgisayar tarafından analiz edilip saklanması mümkün olan büyük veriler; sağlık alanında genomik, proteomiks gibi omik verilerden, sağlık ve medikal kayıt verilerinden, akıllı saat, tablet ve telefon bilgilerinden, laboratuvar ve görüntü cihazları verileri gibi birçok analiz edilmeyi bekleyen verilerdir. Teknik olarak bir veriyi ancak doğru şekilde işlenerek analiz edildikten sonra bilgi haline gelebilmektedir. Bir veriden bilgi elde etmek istiyorsak, ham veriyi iyi tanımalıyız. Verinin depolanmasından, güvenliğinden, doğruluğundan, temizlenmesinden sonra ancak bilgiye erişip paylaşabilen taraf olabiliyoruz [1,2].

1991'de ortaya çıkan Python programlama dili, çeşitli tarihsel ve kültürel nedenlerle yorumlanan diller arasında geniş ve aktif bir bilimsel hesaplama ve veri analizi topluluğu geliştirmiştir. Python'un kütüphaneleri (pandas, numpy ve scikit-learn gibi), onu veri analizi için popüler bir seçim haline getirmiştir. Yorumlanmış bir programlama dili olması, genel olarak çoğu Python kodunun Java ve C++ gibi derlenmiş bir dille yazılmış koddan önemli ölçüde daha yavaş çalışmasını sağlar veya birçok CPU'ya bağlı iş parçacığını içeren uygulamalarda çok iş parçacığı oluşturmak için zorlu bir dil olabilir. Fakat yine de bunlar üstesinden gelinemeyecek özelliklerden değildir ve Python'un işlevselliğini değiştirmez [3].

Tüm biyolojik alanlarda standartlaştırılmış istatistiksel hesaplamalar; ortalamanın ve standart sapmaların hesaplanması (t-testleri, ANOVA'lar, regresyon, vb.) günümüzde birçok laboratuvar ortamındaki lisans öğrencisi tarafından yapılabilmektedir. İstatistiksel hesaplamaların yapılacağı veri analizleri için en çok tercih edilen Python gibi R dili de açık kaynaklı bir programlama dilidir. Ellerindeki verilerden ileri seviyede analiz etmek isteyen veri bilimine meraklı kişiler içinde Python aranan becerileri sağlamaktadır. Veri analizinde Python, çok çeşitli istatistiksel analizleri (pandas, numpy, scipy vb.) ve sonucunun sayısız şekilde görselleştirilmesini (matplotlib, seaborn, plotly vb.) sağlayan kütüphanelere sahiptir [1,4,5].

Python For Data Science Cheat Sheet

Python Basics

Learn More Python for Data Science [Interactively at www.datacamp.com](https://www.datacamp.com)



Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
5
```

Calculations With Variables

>>> x+2	Sum of two variables
7	
>>> x-2	Subtraction of two variables
3	
>>> x*2	Multiplication of two variables
10	
>>> x**2	Exponentiation of a variable
25	
>>> x%2	Remainder of a variable
1	
>>> x/float(2)	Division of a variable
2.5	

Types and Type Conversion

str()	'5', '3.45', 'True'	Variables to strings
int()	5, 3, 1	Variables to integers
float()	5.0, 1.0	Variables to floats
bool()	True, True, True	Variables to booleans

Asking For Help

```
>>> help(str)
```

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> 'm' in my_string
True
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset

```
>>> my_list[1]
>>> my_list[-3]
```

Select item at index 1
Select 3rd last item

Slice

```
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]
```

Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list

Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][1:2]
```

my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

>>> my_list.index(a)	Get the index of an item
>>> my_list.count(a)	Count an item
>>> my_list.append('!')	Append an item at a time
>>> my_list.remove('!')	Remove an item
>>> del(my_list[0:1])	Remove an item
>>> my_list.reverse()	Reverse the list
>>> my_list.extend('!')	Append an item
>>> my_list.pop(-1)	Remove an item
>>> my_list.insert(0, '!')	Insert an item
>>> my_list.sort()	Sort the list

String Operations

Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

>>> my_string.upper()	String to uppercase
>>> my_string.lower()	String to lowercase
>>> my_string.count('w')	Count String elements
>>> my_string.replace('e', 'i')	Replace String elements
>>> my_string.strip()	Strip whitespaces

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
Selective import
>>> from math import pi
```

pandas
Data analysis

scikit-learn
Machine learning

NumPy
Scientific computing

matplotlib
2D plotting

Install Python

ANACONDA
Leading open data science platform
powered by Python

spyder
Free IDE that is included
with Anaconda

jupyter
Create and share
documents with live code,
visualizations, text, ...

NumPy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset

```
>>> my_array[1]
2
```

Select item at index 1

Slice

```
>>> my_array[0:2]
array([1, 2])
```

Select items at index 0 and 1

Subset 2D Numpy arrays

```
>>> my_2darray[:,0]
array([1, 4])
```

my_2darray[rows, columns]

Numpy Array Operations

```
>>> my_array > 3
array([False, False, False,  True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([6, 8, 10, 12])
```

Numpy Array Functions

>>> my_array.shape	Get the dimensions of the array
>>> np.append(other_array)	Append items to an array
>>> np.insert(my_array, 1, 5)	Insert items in an array
>>> np.delete(my_array, [1])	Delete items in an array
>>> np.mean(my_array)	Mean of the array
>>> np.median(my_array)	Median of the array
>>> my_array.corrcoef()	Correlation coefficient
>>> np.std(my_array)	Standard deviation

DataCamp
Learn Python for Data Science [Interactively](https://www.datacamp.com)



Şekil 1: Python.zip içindeki python.ipynb

Print fonksiyonu ve built in function:	1 # Veri yapıları 2
<pre>print("sinan") print("sinan","başarlan","T","B","M",sep='!')</pre> <p>Değişkenler ve built in function:</p> <pre>a="ali" b=5 type(a) type(b)</pre> <p># tek yorum satırı, "string tanımlama"</p> <p>""" çoklu Yorum satırı """</p> <p>#veri tipleri a= 5 type(a) #integer veri tipi b="sinan" #string c= 3.5 #float d='k' #karakter veri tipi</p>	<ul style="list-style-type: none"> • liste, (list) elemanları değiştirilebilir (mutable) • tuple (demet) değiştirilemez (immutable) • dictionary (key,value) • set (aynı elemanı yazamazsın) <pre>liste1=[1,3,3,5,4,5,4,"ali","canan"] liste1[0] #indis,indexler 0 dan başlar len(liste1) #sayma sayıları 1 den başlar dir(liste1) #built in functions</pre> <pre>liste2=[1,3,3,5,4,5,4] liste2.sort() liste2 liste2[0]=100000 #mutable değişir liste2[0] liste2</pre> <pre>liste2.pop() liste2.append(25) liste2.insert(7,"25") liste2 liste2[7]</pre> <pre>demet=("ali","ayşe") demet len(demet) demet[0] dir(demet)</pre> <pre>demet.count("ali")#sayma sayıları 1 den demet.index("ali")</pre>

<p>3</p> <pre>x="sinan" type(x) #string ilkerl veri x.capitalize()</pre>	<p>4</p> <pre>def topla(deger,deger2): print("{} , {} degerlerini alır".format(deger,deger2)) def topla(parametre): print("alınan degerleri: ",parametre) topla(5) topla(3,5) selam() selam("sinan")</pre>
<p>5 Fonksiyon Tanımlama</p> <pre>def selam(): print("selam") help(print) # ?print</pre> <hr/> <p>selam() #parametre almayan, değer döndürmeyen function</p> <hr/> <pre>def selamGotur(isim): print("selam",isim) selamGotur("sinan") #parametre alan değer döndürmeyen</pre> <hr/> <pre>def topla(): print("toplama yapan fonks.") def selam():</pre>	<p>6</p> <p>deger döndüren metotlar/fonksiyonlar</p> <pre>def topla(a,b): c=a+b return c</pre> <hr/> <p>topla(3,5)</p> <hr/> <pre>def kareal(a): return a*a</pre> <hr/> <p>sonuc=kareal(4)</p> <p>print("istenilen değerin karesi {} : ".format(sonuc))</p> <p>sonuc=kareal(8)</p> <p>print("istenilen değerin karesi ",sonuc)</p>

<p>7</p> <pre> print("sinan") def selam(d): print("grilen deger",d) help(format) #?format a= int(input("1. sayıyı giriniz: ")) b= int(input("2. sayıyı giriniz: ")) def cıkar(x,y): print("sonuc :",a-b) return a-b def kareal(x): #print("a nın karesi:",a*a) return a*a sonuc=cıkar(a,b) kareal(a) cıkar(5,5) </pre>	<p>8</p> <pre> def ceyrek(x): return x/4 def küpal(x): return x*x*x küpal(3) ceyrek(4) küpal(ceyrek(12)) print(küpal(ceyrek(20))) liste1=[1,2,3,4,5] liste1 #tek boyutlu dizi #liste tek artırma liste2=[i*2 for i in liste1] liste2 kare=lambda x:x*x #kare al fonksiyonu yerine def toplama(x,y): return x+y toplama(5,5) #lambda ile toplama=lambda x,y: x+y toplama(5,5) </pre>
---	---

<pre> isim="sinan" pass="sinan" # # modül kavramı =>library, dll import math #include,import..... """ pandas,numpy scikit learn (veri madenciliği) opencv (görüntü işleme) matplotlib,seaborn=>veri görselleştirme math=> matematik işlemleri genelde her prog. dilinde var """ help(math) #?math </pre>	<pre> import math as matematik matematik.pow(2,2) 2**2 matematik.cos(60) matematik.sin(60) matematik.sqrt(4) from math import * #tüm math library getirir import time import random rastgele=random.randint(1,100) rastgele </pre>
---	--


```
import random
import time

rastgele=random.randint(1,100)
tahmin=5

while True:

    tahmin=int(input("tahmin gir 1-100 arası :"))
    print("kontrol ediliyor")

    if( tahmin==rastgele):
        print("kontrol ediliyor")

        time.sleep(1)

        print("girilen sayi",tahmin)
        break

        print("tebrikler")
    elif( tahmin<rastgele):
        tahmin=tahmin-1
        print("sayıyı büyült")
    elif(tahmin>rastgele)
        tahmin=tahmin-1
        print("sayıyı küşçült")
    else:
        print("1-100 de değer gir")

    if(tahmin==0):
        print("hak bitti")
        break
```

2. Sınıflar/Classes

Nesneye Dayalı Programlama da asıl amaç, gerçek hayatta var olan olguları programlamaya aktarma yaklaşımıdır. Bu yaklaşım temelde hiyerarşi düzenini baz alır. Bu hiyerarşiyi de sınıflar ve nesneler oluşturur.

Mesela bir proje üzerine çalışan bir ekip düşünelim. Bu ekip nesneye dayalı programlama mantığında sınıfa karşılık gelir. Bu ekipteki her bir üye ise nesneye karşılık gelir. Bu ekipteki her bir üyenin yani nesnenin, diğer nesneler ile etkileşimi ve organize çalışması sonucunda Nesneye Dayalı Programlama kavramı ortaya çıkmaktadır.



Resim 1: OOP-Object Oriented Programming-Nesneye Dayalı programlama

Resim 1’de OOP kavramıyla ilişkili yapıları görülmektedir. Bu kavramlardan dört tanesi Nesneye Dayalı Programlamanın temel prensiplerini oluşturmaktadır. Nesneye yönelik programlama,

- Inheritance (Kalıtım/Miras alma)
- Encapsulation (Kapsülleme)
- Abstraction (Soyutlama)
- Polymorphism (Çok Biçimlilik)

bu dört temel esastan meydana gelmektedir. Fakat Nesneye Dayalı Programlama prensiplerini daha iyi anlamak için bunlardan önce Sınıf ve Nesne kavramlarını iyi bir şekilde anlamamız gereklidir.

Sınıf.py dosyasının içeriği Spyder editörde yazıldı.
Pycharm'da çalışır. Fakat # %% ifade anlamsız görünür.

```
"""
Spyder Editor

aracılığıyla sınıf yapısına giriş-1
"""

# %%
a=5
string="dogus"
#ctrl+enter run
# %% Class-sınıflar

isci_ismi="ali"
isci_yas= 50
isci_adres="Üsküdar"
isci_kurum="dogus"

"""her çalışan için
tanımlama yapmayız
sınıf oluştururuz
"""

#class ortak nesnelere ait özellikleri bir arada tutan yapıdır.

class Calisan():
    #özellikler; yaş,adres,tc,ad,soyad.....vs
    #metot/davranışlar; çalışma,terfi alma.....vss

    pass #içeriği sonra oluşturacağım

calisan1=Calisan() #nesne türetildi.
```

1. Nitelikler/Attributes veya properties

Etrafınızda gördüğünüz her şey aslında birer nesnedir. Şuan kullandığınız bir elektronik cihaz veya çalıştığınız masa, evinizin penceresi, dışarıdaki arabalar, sokaktaki kedi vs. nesne kavramına örnektir. Nesnelerin önemli iki özelliği vardır. Durum ve davranış.

Mesela bir telefon düşünelim. Bu telefonun rengi, fiyatı, markası, modeli, boyutları, işlemcisi, hafızası vs. gibi özellikleri onun durumunu belirtir. Buna attribute/property (özellik) de denir.

Yine bu telefonun çağrı yapmak, resim çekmek, ses kaydı almak, parlaklık ayarını değiştirmek gibi eylemleri ise onun davranışını belirtir. Dolayısıyla buradan şöyle bir tanım çıkartabiliriz.

Telefon örneğimizin durumlarında belirttiğimiz özellikler, Nesneye Dayalı Programlama tarafında, değişkenler sayesinde hafızada saklanır.

Yani biz nesnemin durumlarını değişkenler ile tanımlayacağız.

Nesneye yönelik programlama da davranışların karşılığı ise metotlardır. Yani nesnemize ait bir çeşit fonksiyon olarak da tanımlayabiliriz.

Sınıf2.py dosyasının içeriği---- Spyder editörde yazıldı
Pycharmda çalışır. Fakat # %% ifade anlamsız görünür

```
"""
sınıf yapısı 2
"""

# %% öznelik-property veya attribute

class Arac():

    renk="sarı"
    model="jeep"
    marka="kia"

a1=Arac()

print(a1) #a1 bir objedir çıktısı veriyor.

print(a1.marka)
print(a1.model)
print(a1.renk)

a1.marka="BMW"

print(a1.marka)
# %% Metotlar

class Kare():
    kenar=10 #metre

    def alan(self):
        alan=self.kenar*self.kenar
        print("Alan",alan)

k1=Kare()

print(k1)
print(k1.kenar)

print(k1.alan())
```

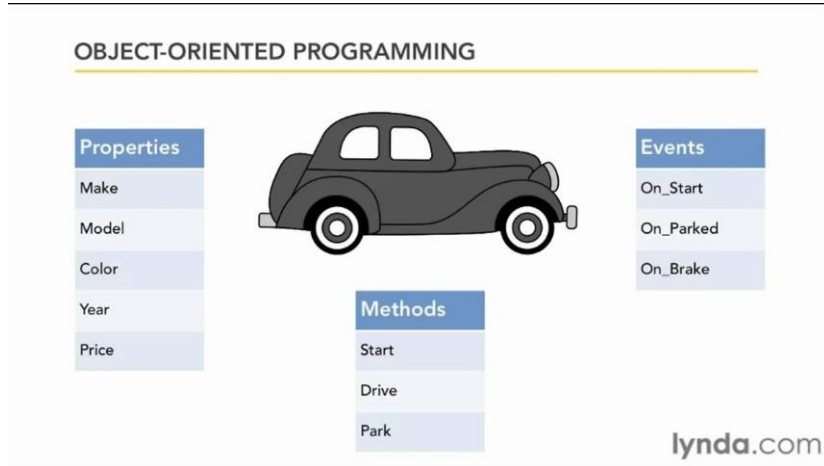
bir araba nesnesi üzerinden gidelim. İlk olarak bu nesnemizin durumları nelerdir, onları belirleyelim.

- Rengi,
- Fiyatı,
- Markası,
- Modeli,
- Beygir gücü,
- Yakıt tipi,
- Ağırlığı

gibi özellikler araba nesnemizin durumlarını yani programlama karşılığı olarak da, nesnenin değişkenlerini temsil eder. Bu arabanın,

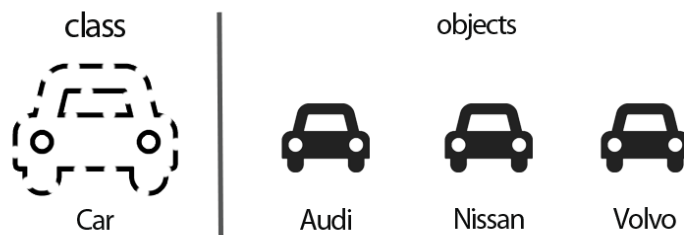
- Hızlanması,
- Yavaşlaması,
- Vites değiştirmesi,
- Komuta edilmesi

gibi özellikleri de araba nesnemizin **davranışlarını** yani nesneye dayalı programlama karşılığı olan **metotlarımızı** temsil eder. İşte **sınıf** kavramı da tam bu noktada ortaya çıkmaktadır. Bu **durumlar** ve **davranışların** beraber tutulduğu yer **sınıftır**.



Resim 2: Araba Sınıfı

Bu sınıfa araba sınıfı diyebiliriz ve bu araba sınıfından istediğimiz kadar araba üretebiliriz. Bu ürettiğimiz yeni nesneleri biz programlamada **instance (örnek)** olarak adlandırırız. Yine bu araba sınıfından oluşturduğumuz her **araba nesnesinin**, değerleri (rengi,modeli gibi) farklı olsa da, yukarıda yazdığımız **durum** ve **davranışlara** sahiptir.



Resim 3: Sınıf ve nesne kavramı

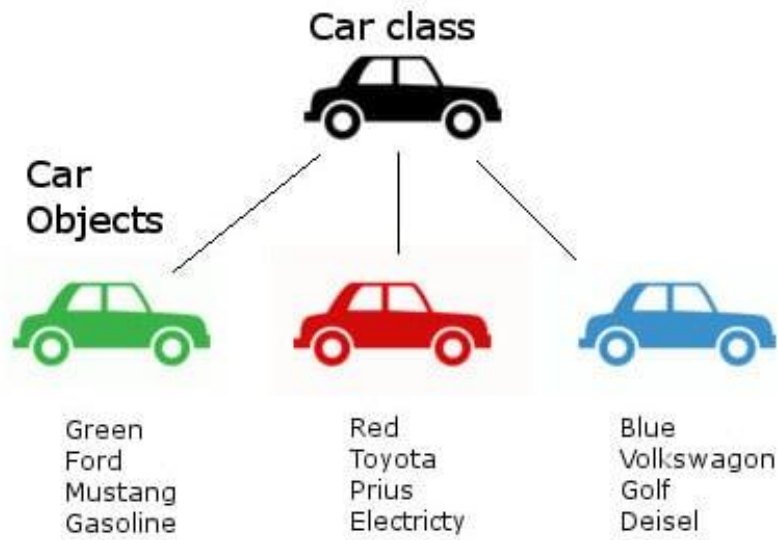
Yani aslında **sınıf**, soyut bir veri tipidir. **Nesne (object)** ise onun somutlaşan bir varlığıdır. Sınıflarda, nesnelerde olması gereken **özellikler**, yani **durum** ve **davranışlar** bulundurulur. Bu sınıftan bir **instance (örnek)** oluşturduğumuzda ise buna **nesne** denir. Dolayısıyla bir sınıftan istediğimiz kadar **nesne** üretebiliriz.

Fakat sadece **aynı sınıftan** oluşturulan nesnelerin **tipi** aynıdır. Yani bir **telefon** sınıfından üretilen bir nesne ile **araba** sınıfından üretilen nesnenin **özellikleri** farklıdır.

Karıştırılmaması gereken önemli bir nokta ise, **aynı sınıftan üretilen nesneler** aynı ortak özelliklere sahip olsalar da, bu özelliklerin **değeri farklıdır**. Örneğin bir **insan** sınıfı düşündüğümüzde, bazı insanlar zayıf bazı insanlar şişman olabilir. Ya da başka bir örnek olarak yandaki araba sınıfı örneğine bakalım.

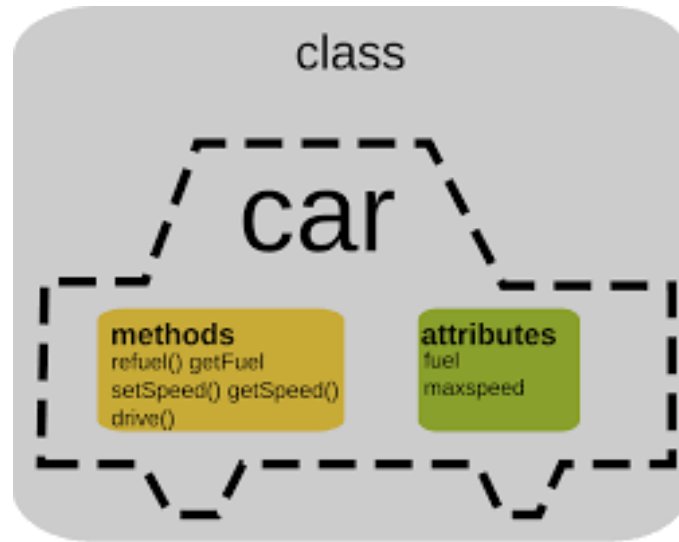
Aynı sınıftan üretilen fakat **renk, marka, model ve yakıt tipleri farklı** olan araba nesneleri verilmiş. Bu nesnelerin **ortak özellikleri** aynı olsa da, özellik değerlerinin **farklı** olduğunu görüyoruz. İşte **nesneye dayalı programlama mantığı** da, burada başlıyor aslında.

Aynı sınıftan üretilen farklı değerlere sahip araba nesneleri



Resim 4: Sınıf ve nesne kavramı 2

2. Metotlar-Fonksiyonlar



Resim 5: Sınıf ve içerisindeki metotlar ile özellikler

Pycharm'da da çalışır. Fakat # %% ifade anlamssız görünür

%% metotlar

```
class Isci():
```

```
    yas=20
    maas=1000
```

```
    def yasaGoreMaasOranla(self):
        print(self.yas/self.maas)
```

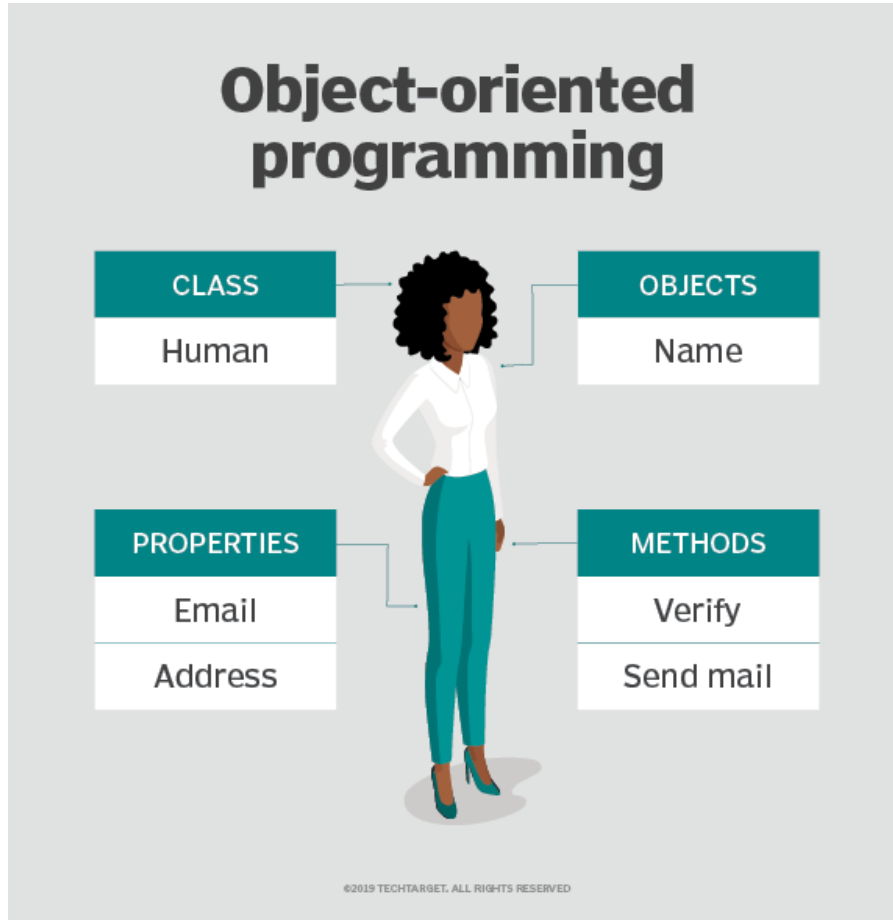
```
isci1=Isici()
```

```
isci1.yasaGoreMaasOranla()
```

```
def yasMaasOranı(yas,maas):    #class dışı metot
    a=yas/maas
    #print("oran:\t",a)
    print("oran:\t {}".format(a))
```

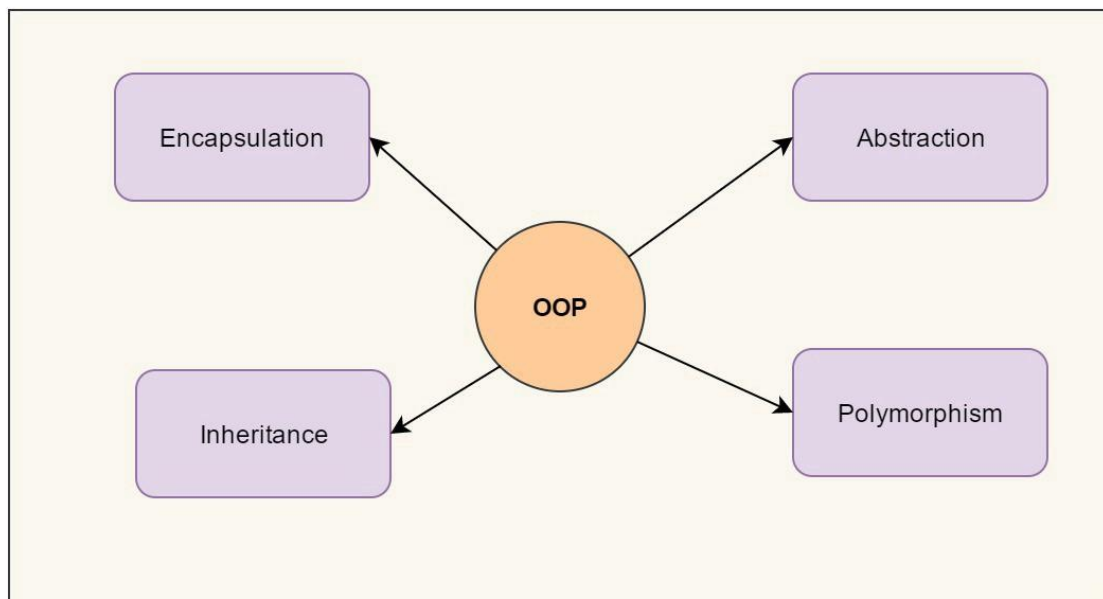
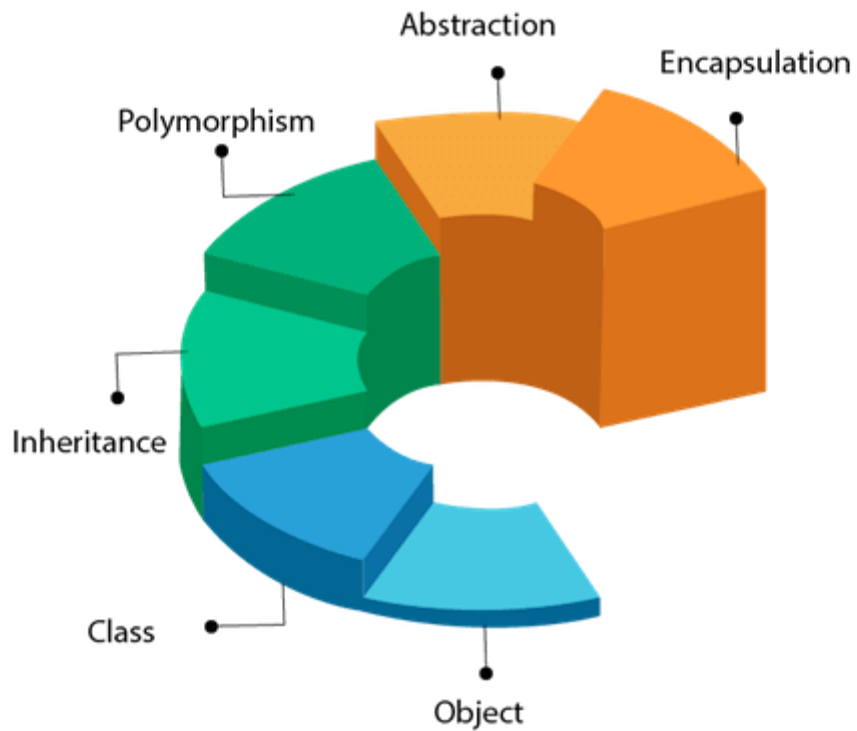
```
yasMaasOranı(20,2000)
```

OOP'nin Temellerini Özetleyen resim



Resim 6: OOP Özet Resim

OOPs (Object-Oriented Programming System)



Four Pillars of Object Oriented Programming

Sorular:	
1.Hangi keyword ile sınıf tanımlanır A. def B. class C. return D. __init E. Hiçbiri	
2.Hangisi ile python kodu yazılamaz A. Notepad++ B. jupyter C. spyder D. pycharm E. chrome.py	
3.Nesneye dayalı programlama ile ilgili değildir. A. Sınıf yapısı ile nesne türetilir B. Sınıflar içeriği olarak, nitelik,metotlar olur C. Bir sınıftan birden çok nesne türetilemez D. Metotlar sınıftan türetilen nesnelerin davranışlarıdır. E. Sınıfların özellikleri public,private,protected,internal..gibi erişimler alabilir.	
kısayoldan kod run etme jupyterde shift+enter Spyderda f5 veya ctrl+enter	bilgi
Spyderda seçili alanı çalıştırmak için F9'a basmak yeterli	
#yorum satırı <pre> """ Çoklu Yorum satırı """ </pre> #yazıyı italik yapmak için <i>*italik*</i> yıldız il başlanıp bitirmek yeterlidir.	
4.python dosya uzantısı hangisidir. A. .pyt B. .py C. .ps D. .ph E. .pt	
5.Hangisi ile fonksiyon oluşturulur. A. create fonksiyon1(): B. def fonksiyon1(): C. func fonksiyon1(): D. class fonksiyon1(): E. **init__ fonksiyon1():	
6. bir sınıftan yeni ve eşsiz bir örnek(instance) oluşturmak için hangisi gerekli A. class B. return C. constructors D. def	

3. Yapıcı /Constructor/Initializer

Yapıcı Metot

constructor veya (or) initializer metot

```
class Hayvan():
```

```
    def __init__(self,isim,yas): #yapıcı/constructor metot
        self.isim=isim
        self.yas=yas
```

```
    def getYas(self):
        return self.yas
```

```
    def getAd(self):
        return self.isim
```

```
h1=Hayvan("dog",2)
```

```
h1_yas=h1.getYas()
```

```
print("h1 in yaşı :",h1_yas)
```

```
h1_isim=h1.getAd()
```

```
print("h1 in isim :",h1_isim)
```

```
h2=Hayvan("cat",3)
```

```
h2_yas=h2.getYas()
```

```
print("h2 in yaşı :",h2_yas)
```

3.1 OOP ile Basit Bir Hesap Makinesi Proje

Pycharm veya Jupyterde açabilirsiniz—

Bunun ikinci versiyonu olan oop ile hesap makinesi-dışardan değer almalı olana da bakınız

```
# -*- coding: utf-8 -*-
"""
OOP ile Hesap makinesi
@author: msinan
"""
class Makine():
    "hesap makinesi"

    def __init__(self,a,b):
        "başlangıç değerlerini ayarlar"
        #öznitelik alır
        """ pass """

        self.deger1=a
        self.deger2=b

        # pass
        #sonra oluşturursak pass

    def topla(self):
        " toplama a+b = sonuc -> return sonucs"
        sonuc=self.deger1+self.deger2
        return sonuc

        """pass"""

    def carp(self):
        "carpma a*b= sonuc -> return sonuc"
        sonuc=self.deger1*self.deger2
        return sonuc
        """pass"""

    def cikar(self):

        return self.deger1-self.deger2

    def bol(self):

        return self.deger1/self.deger2

x=5
y=2
h: Makine=Makine(x,y)
tSonuc=h.topla()
cSonuc=h.carp()
print("toplama sonuc: {}, çarpma sonucu: {}".format(tSonuc,cSonuc))
```

4. Kapsülleme/Encapsulation

Encapsulation adı verilen yapı, bir sınıf içerisindeki değişkenlere “kontrollü bir şekilde erişimi sağlamak /denetlemek” için kullanılan bir yapıdır. Class içerisindeki değişken private yapılarak dışarıdan direkt erişilmesi engellenir, bu değişken içerisine değer atıp değer okumak için get ve set adı verilen metodlar kullanılır. Yani direkt değişkene değil, bu değişkene erişmek (işlem yapmak) için bu metodlar ile çalışılır. set değeri alıp değişkene atar, get’de değeri geri döndürür, tabii bu işlem sizin belirlediğiniz, olması gereken kurallar çerçevesi içinde gerçekleşir.

Python'un özel anahtar sözcüğü yoktur (private keyword), diğer nesne tabanlı dillerin aksine, kapsülleme yapılabilir. Bunun yerine, sözleşmeye dayanır: doğrudan erişilmesi gereken bir sınıf değişkenine altçizgi başlığı eklenmelidir.

Kapsülleme, verilerin nitelik olarak bir yerde güvenli bir şekilde depolanmasını sağlamakla ilgilidir. Kapsülleme şunları bildirir:

- Verilere yalnızca örnek yöntemlerle erişilmelidir.
- Veriler, sınıf yöntemlerinde belirlenen doğrulama gereksinimine dayalı olarak her zaman doğru olmalıdır.
- Veriler, dış işlemlerdeki değişikliklerden korunmalıdır.

Nesne özellik değerlerine erişmek için 'setter' ve 'getter' yöntemlerinin kullanılması gerekir.

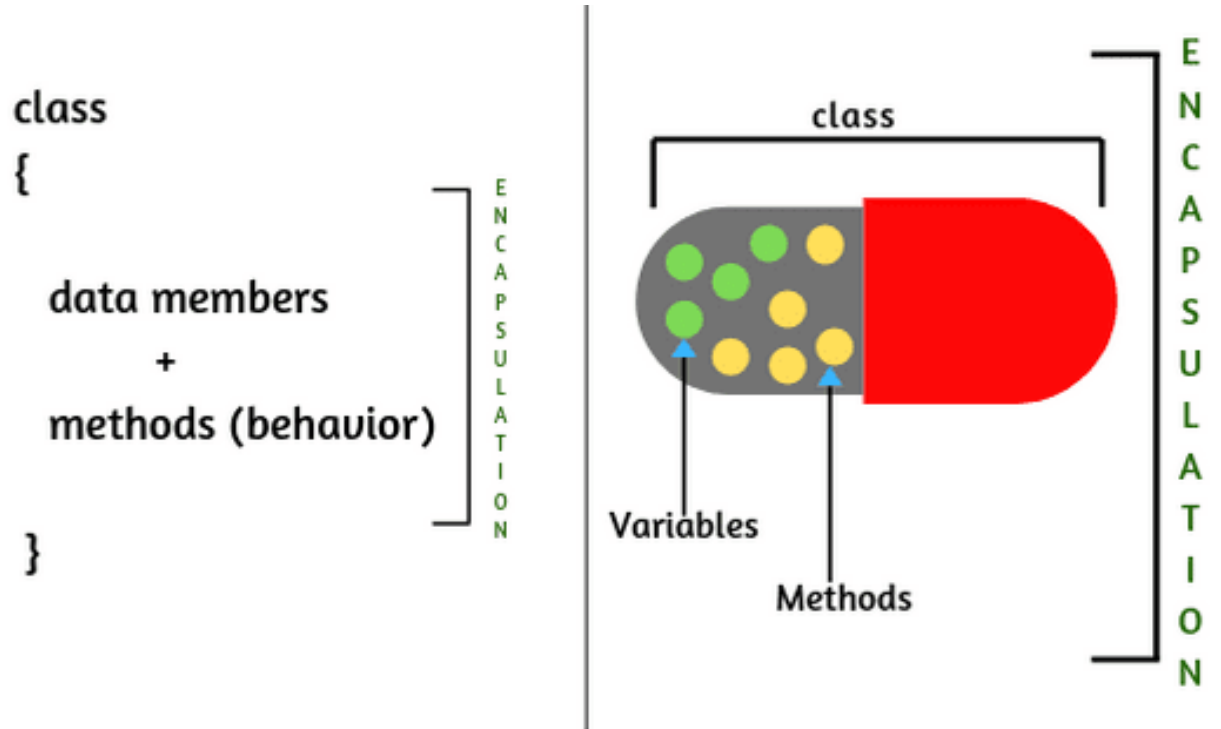
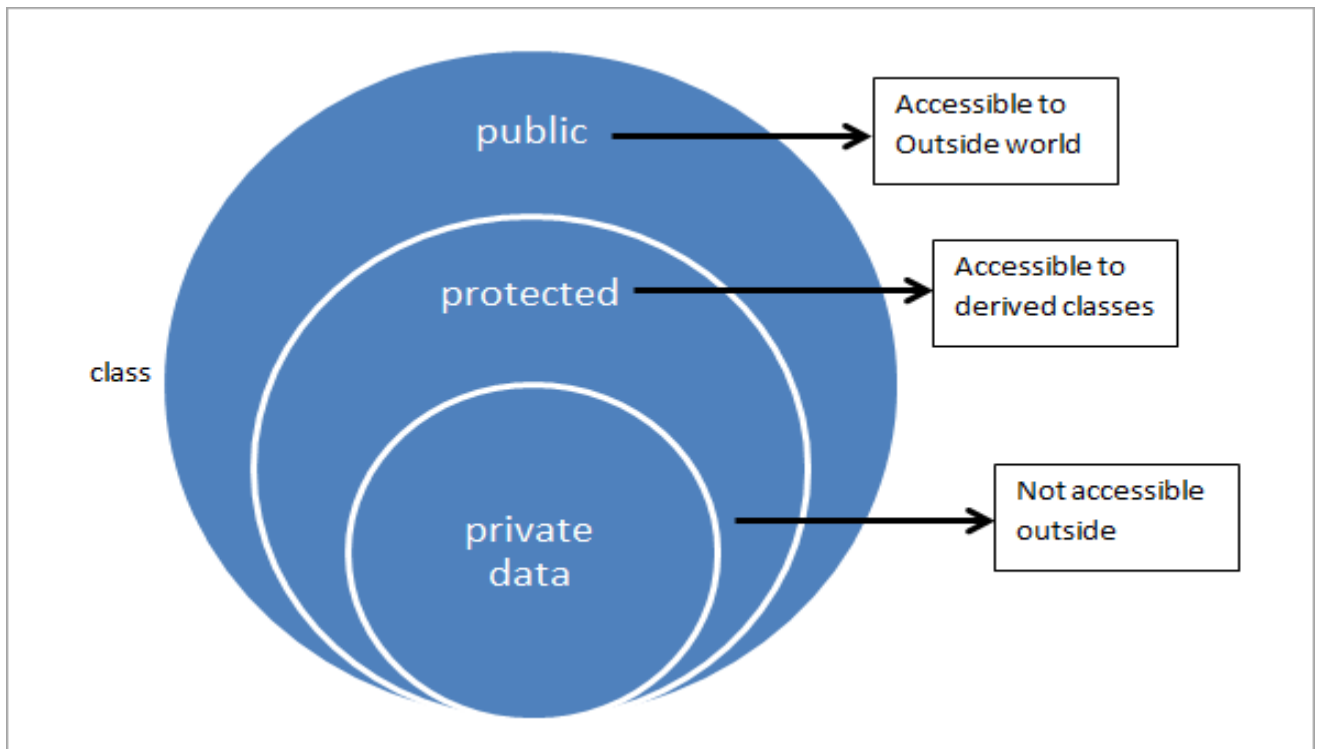


Fig: Encapsulation

Nasıl yapılır için aşağıdaki resim her şeyi açıklamaktadır.



Encapsulation Örneği

```
"""
encapsulation

@author: msinan
"""

class Banka():

    def __init__(self,isim,para,adres):
        self.isim=isim
        self.para=para
        self.adres=adres

hesap1=Banka("Sinan",1000,"istanbul")
hesap2=Banka("Ayşe",5000,"Erzurum")

hesap1.para

hesap2.para=hesap2.para+hesap1.para #sinan'ın parası kalmadı

hesap2.para
hesap1.para=0
hesap1.para

#bunu engellemek için public değişkenleri private yapmalıyız
```

```

%% encapsulation

class Banka2():

    def __init__(self,isim,para,adres):
        self.__isim=isim
        self.__para=para
        self.__adres=adres

    #get ve set metotları

    def getPara(self):
        return self.__para

    def setPara(self,miktar):
        self.__para=miktar

    def islemSayisi(self):
        self.__para=self.__para-10

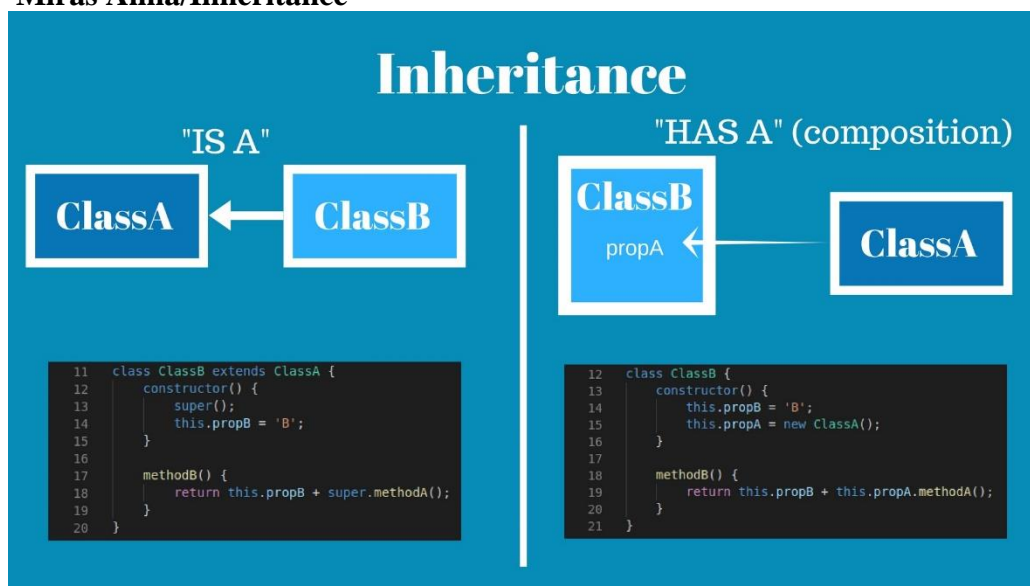
hes1=Banka2("Sinan",1000,"istanbul")
hes2=Banka2("Ayşe",5000,"Erzurum")

print("1. hesaptaki para: ",hes1.getPara())
hes1.setPara(100)
print("set işlemi sonrası 1. hesaptaki para değişimi :",hes1.getPara())

#hes1.islemSayisi()
#print("işlem ücreti :",hes1.getPara())

```

5. Miras Alma/Inheritance



Miras Alma Örneği

```
#ata sınıf --- parent
class Animal():
    def __init__(self):
        print("hayvan sınıfının yapıcı metotum")

    def sesCikar(self):
        print("hav,miyav,vak....")

    def hareket(self):
        print("zıplar,koşar,yürür..")

#cocuk sınıf ----child

class kedi(Animal):
    def __init__(self):
        super().__init__() # yazmasakta olur ata sınıfın init metodunu ezeriz metot overriding!!

        print("kedi sınıfı oluşturuldu")

    def sesCikar(self):
        print("miyav")

    def DokuzCan(self): #diğer hayvanlardan ayrılan bir fonksiyon :D
        print("Bu sevimli hayvanlar hep dört ayak üstüne düşer")

k1=kedi()

k1.sesCikar() #ata sınıfı ezer
k1.hareket()

k1.DokuzCan()

class kus(Animal):
    def __init__(self):
        print("kus sınıfı oluşturuldu")

    def ucma(self):
        print("kanatları vardır uçarlar")

kus1=kus()
kus1.ucma()
kus1.hareket()
```


6. Soyut Sınıf/Abstract Class

Nesnelerin oluş tarzını ifade eden kavramlara soyut kavram denir. Bir restorana gittiğimiz zaman, bizimle ilgilenen garsona “bize biraz besin getirebilir misin?” diyebilir miyiz?

Akliselim bir insan isek cevabımız hayır olacaktır. Biraz besin yerine “zeytinyağlı enginar” dersek daha doğru olacaktır, çünkü besin soyut bir kavramdır, zeytinyağlı enginar ise besin sınıfına ait bir nesnedir.

Abstract base class mantığındadır, sadece temel oluşturmak için kullanılır, yani bu sınıf inheritance alındıktan sonra kullanılabilir, tek başına işlevi yoktur, bu nedenle en karakteristik özelliği **kendisinden nesne üretmesine izin vermez**, abstract olmayan bir sınıfın içerisinde abstract bir elaman olamaz. Abstract bir sınıfı, miras alan bir sınıf, bu miras aldığı abstract sınıfı içerisindeki **metodu override** eder.

Örnek: abc modülü, türetilmiş bir sınıfın, o yöntem üzerinde özel bir @abstractmethod dekoratörü kullanarak belirli bir yöntemi uygulamasına izin verir. Soyut (Abstract) bir yöntem uygulanabilir ancak yalnızca türetilmiş bir sınıftan “super” ile çağrılabilir.

```
from abc import ABC, abstractmethod
```

```
class Animal(): # super clas
    pass
```

```
class kus(Animal):
    pass
```

```
a=Animal()
```

```
from abc import ABC, abstractmethod
```

```
class Animal(ABC): # super clas
    @abstractmethod
    def yurume(self): pass
```

```
    @abstractmethod
    def kosma(self): pass          #sablon oluşturup tekrar tekrar kullan
```

```
class kus(Animal):
```

```
    def __init__(self):
        print("kuş oluşturuldu")
```

```
    def yurume(self):
        print("kuslar pek yürümez")
```

```
    def kosma(self):
        print("kuslar pek kosmazda")
```

```
#a=Animal() soyut sınıflardan nesne oluşturulmaz.  
b=kus() #abstract clasın içeriği yavru sınıfta doldurulur.  
b.kosma()  
b.yurume()
```

7. Overriding

override: Inheritance alınan bir sınıftaki metodu tekrar yazarak, base class'takini ezmiş, yoksaymış yani override etmiş oluruz. Bunun gerçekleşebilmesi için base class'taki metodun virtual olarak tanımlanması gerekir, yani buna izin vermesi gerekir ki virtual'un tanımı bu şekilde yapılsın.

overload: Türkçe karşılığı ile bir metodun aşırı yüklenmesidir, yani aynı isimde birden fazla metod farklı parametreler alabilir, ya da almayabilir.

```
class Animal():  
  
    def sesVer(self):  
        print("ses çıkarırlar")  
  
class kedi(Animal):  
  
    def sesVer(self): #ata sınıfı ezdi  
        print("miyav")  
  
a=Animal()  
a.sesVer()  
  
k=kedi()  
k.sesVer()
```

8. Polymorphism/Çok biçimlilik

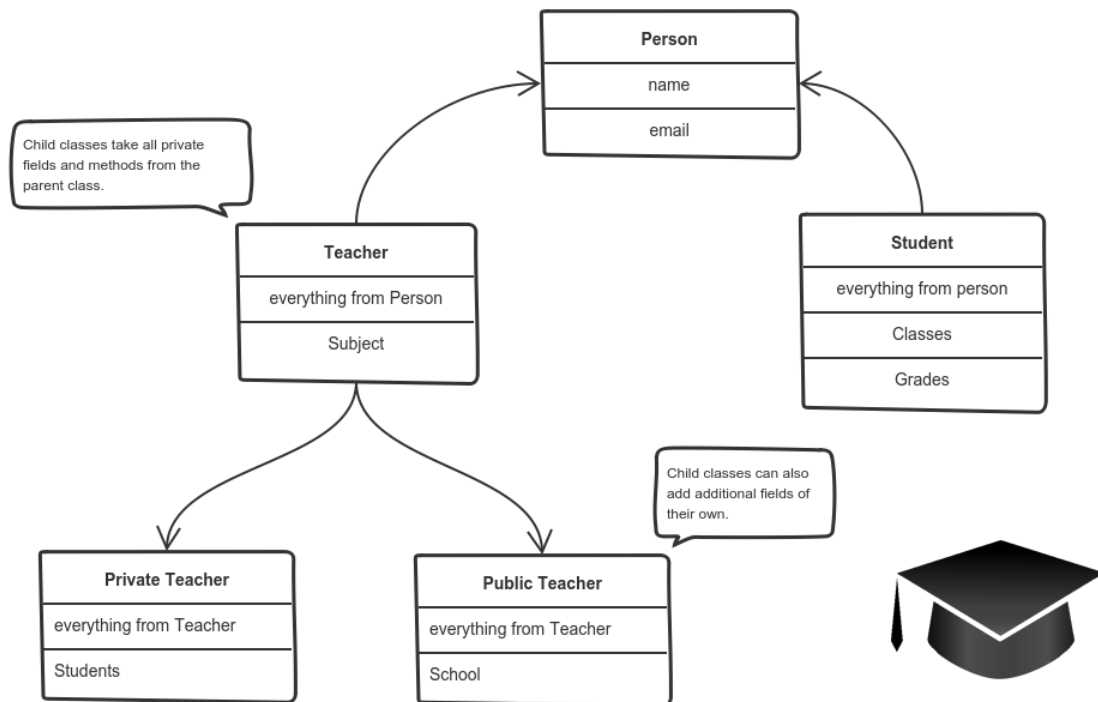
Nesneye yönelik programlamanın en önemli özelliklerinden biri de çok biçimliliktir. Çok biçimlilik, bir sınıf içinde aynı isimde fakat farklı içerikte metotlar kullanabilme özelliğidir. Çok biçimlilik, aynı metodun değişik sınıflar içinde yer alabilme imkanındır.

```
class Hayvanlar:  
    def __init__(self, isim):  
        self.isim = isim  
    def tepki(self):  
        raise NotImplementedError('HATA')  
class Kedi(Hayvanlar):  
    def tepki(self):  
        return 'Miyav!'  
class Kopek(Hayvanlar):  
    def tepki(self):  
        return 'Haav! Hav!'  
hayvan = [Kedi('Boncuk'),
```

```

Kedi('Tekir'),
Kopek('Elmas')]
for hyvn in hayvan:
    print(hyvn.isim + ': ' + hyvn.tepki())

```



```

class Animal():
    def __init__(self, name): # Constructor of the class
        self.name = name
    def talk(self):           # Abstract method, defined by convention only
        raise NotImplementedError("Subclass must implement abstract method")
class Cat(Animal):
    def talk(self):
        return 'Meow!'
class Dog(Animal):
    def talk(self):
        return 'Woof! Woof!'
animals = [Cat('Missy'), Cat('Mr. Mistoffelees'), Dog('Lassie')]

for animal in animals:
    print (animal.name + ': ' + animal.talk())

```

Program çalıştırılırsa,
Missy: Meow!
Mr. Mistoffelees: Meow!
Lassie: Woof! Woof!

Sorular:	
1. Hangisi encapsulation tanımıdır A. Bir sınıftan yeni eleman türetme işlemine denir B. Ata sınıfın özelliklerini almaya denir C. Bir nesnenin metot ve verilerini diğer nesnelerden saklayarak erişime engelleyerek yanlış kullanımı engel olmaktır	
2. yandaki koda ile ilgili bilgilerden hangisi doğrudur. A. A sınıfı B den miras alır. B. B sınıfı A dan miras alır. C. B sınıfı A dan miras alır. Ama tüm özelliklerini almaz	<pre> class A(): def __init__(self,a): print("a sınıfı oluşturuldu") def ilk(self): print(" ilk isimli metot") class B(A): def __init__(self,j=10): self.j=j </pre>
3. Hangisi soyut sınıf (abstract class) tanımıdır A. Bir sınıftan başka sınıf türetebilmek için oluşturulur. B. OOP de nesnesi olmayan sınıflara verilen isimdir. C. Bir sınıfın üst sınıftan miras almasına denir.	
<pre> class Animal(): def sesVer(self): print("ses çıkarırlar") class kedi(Animal): def sesVer(self): #ata sınıfı ezdi print("miyav") a=Animal() a.sesVer() -----> çıktı ?? k=kedi() k.sesVer() -----> çıktı ?? </pre> <p style="text-align: right;">a.sesVer() ve k.sesVer() çıktısı nedir?</p>	

Çalışmak için Kaynaklar

Sonda değil başta veriyorum

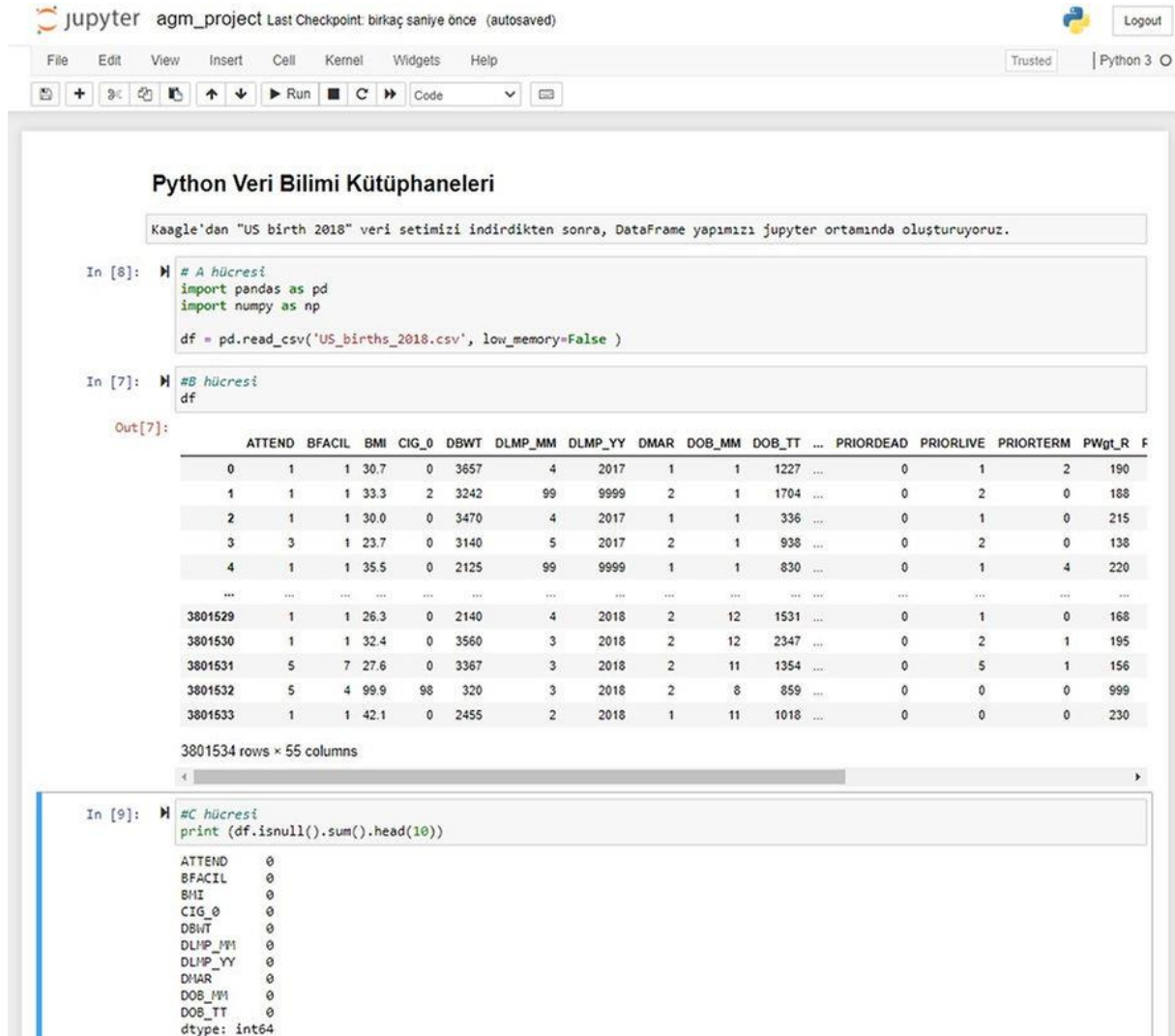
Python:	https://www.python.org/ https://realpython.com/python-basics/ https://www.tutorialspoint.com/python/index.htm
Python Temelleri:	https://www.learnpython.org/ https://www.programiz.com/python-programming
Python OOP:	https://realpython.com/python3-object-oriented-programming/ https://www.programiz.com/python-programming/object-oriented-programming https://www.tutorialspoint.com/python/python_classes_objects.htm
Sınav	https://www.sanfoundry.com/

9. Jupyter

2001 yılında Fernando Perez'in daha iyi etkileşimli Python yorumlayıcısı yapma yan projesi olarak başlayan IPython projesi, sonraki 16 yılda modern Python veri yığınındaki en önemli araçlardan biri haline geldi. Diğer birçok programlama dilinin tipik düzenleme derleyici çalışma iş akışı yerine yürütmeyi keşfetme iş akışını teşvik eder. Ayrıca işletim sisteminizin kabuğuna ve dosya sistemine kolay erişim sağlar. 2014'te Fernando ve IPython ekibi, dilden bağımsız etkileşimli bilgi işlem araçlarını tasarlamak için daha geniş bir girişim olan Jupyter projesini duyurdu. IPython web not defteri, şimdi 40'tan fazla programlama dilini destekleyen Jupyter notebook haline geldi. IPython kabuğu ve Jupyter not defteri veri keşfi ve görselleştirme için kullanışlıdır. Jupyter notebook, Python kodunun yazılmasını, test edilmesini ve hata ayıklanmasını hızlandırır. Ayrıca Markdown ve HTML içerik yazmanıza izin vererek, size kod ve metin içeren zengin belgeler oluşturmak için olanak sunar [3,16].

9.1 Online IDE

<https://pythontutor.com/>



The screenshot shows a Jupyter notebook titled "agm_project" with a "Last Checkpoint: birkaç saniye önce (autosaved)" status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving. The notebook content is as follows:

Python Veri Bilimi Kütüphaneleri

Kaagle'dan "US birth 2018" veri setimizi indirdikten sonra, DataFrame yapımızı jupyter ortamında oluşturuyoruz.

```
In [8]: # A hücresi
import pandas as pd
import numpy as np

df = pd.read_csv('US_births_2018.csv', low_memory=False)
```

```
In [7]: #B hücresi
df
```

Out[7]:

	ATTEND	BFACIL	BMI	CIG_0	DBWT	DLMP_MM	DLMP_YY	DMAR	DOB_MM	DOB_TT	...	PRIORDEAD	PRIORLIVE	PRIORTERM	PWgt_R	F
0	1	1	30.7	0	3657	4	2017	1	1	1227	...	0	1	2	190	
1	1	1	33.3	2	3242	99	9999	2	1	1704	...	0	2	0	188	
2	1	1	30.0	0	3470	4	2017	1	1	336	...	0	1	0	215	
3	3	1	23.7	0	3140	5	2017	2	1	938	...	0	2	0	138	
4	1	1	35.5	0	2125	99	9999	1	1	830	...	0	1	4	220	
...
3801529	1	1	26.3	0	2140	4	2018	2	12	1531	...	0	1	0	168	
3801530	1	1	32.4	0	3560	3	2018	2	12	2347	...	0	2	1	195	
3801531	5	7	27.6	0	3367	3	2018	2	11	1354	...	0	5	1	156	
3801532	5	4	99.9	98	320	3	2018	2	8	859	...	0	0	0	999	
3801533	1	1	42.1	0	2455	2	2018	1	11	1018	...	0	0	0	230	

3801534 rows x 55 columns

```
In [9]: #C hücresi
print (df.isnull().sum().head(10))
```

ATTEND 0
BFACIL 0
BMI 0
CIG_0 0
DBWT 0
DLMP_MM 0
DLMP_YY 0
DMAR 0
DOB_MM 0
DOB_TT 0
dtype: int64

Şekil 4: Jupyter notebook

Şekil 4'te Jupyter ortamında Kaagle veri seti kullanımı. Jupyter'in HTML ve Markdown özelliği ilk iki satırda gösterilmiştir. A hücresinde, Python'un pandas ve numpy kütüphanesini

indirdikten sonra, .csv uzantılı dosyamızı okutuyoruz. B hücresinde, DataFrame yapısındaki verimizi yazdırdığımızda satır ve sütunlardan oluştuğunu görüyoruz. C hücresinde ise, oluşturulan df içerisindeki sütunlar içerisinde boş değer olup olmadığını kontrol ederek toplamalarını (head – ilk 10 sonucu gösterir) yazdırıyoruz [16].

9.2 SciPy, Scikit-learn

SciPy, bilimsel hesaplamada bir dizi farklı standart problem alanını ele alan bir açık kaynaklı paketler koleksiyonudur. Örneğin; scipy.linalg: doğrusal cebir rutinleri ve matris ayrıştırmaları, scipy.signal: signal işleme araçları, scipy.stats ise standart sürekli ve ayrık olasılık dağılımları, çeşitli istatistiksel testler ve daha fazlası tanımlayıcı istatistikleri içermektedir [3,17].

Scikit-learn, 2010 yılından beri scikit-learn Python programcıları için önce gelen makine öğrenimi araç seti haline geldi. Makine öğrenimi için alt modüller içerir. Bunlardan bazıları; sınıflandırma (classification) için: SVM, en yakın komşu, rastgele orman, lojistik regresyon vb., regresyon için: Lasso, ridge regresyon vb., kümeleme (clustering) için: k-means, spektral kümeleme vb., boyut azaltma (dimensionality) için: PCA özellik seçimi/feature selection, matris çarpanlara ayırma/matrix factorization, model seçimi (model selection): grid/ızgara arama, çığraz doğrulama/cross validation, metrikler, ön işleme (preprocessing): özellik çıkarma/feature extraction, normalleştirme için kullanılan modellerdir [3,11].

10. Referanslar

1. David, A. A. (2021). Introducing Python Programming into Undergraduate Biology. The American Biology Teacher, 83(1), 33-41.
2. Dash, S., Shakyawar, S. K., Sharma, M., & Kaushik, S. (2019). Big data in healthcare: management, analysis and future prospects. Journal of Big Data, 6(1), 1-25.
3. Wes McKinney. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.), O'Reilly 2018
4. Adams, C. (2014). Learning Python data visualization. Packt Publishing Ltd.
5. Waskom, M. L. (2021). Seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021.
6. Rabbi Radliya, N. (2019). [2] Numpy User Guide.
7. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
8. Rajagopalan, G. (2021). Prepping Your Data with Pandas. In A Python Data Analyst's Toolkit (pp. 147-241). Apress, Berkeley, CA.
9. Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., ... & Scopatz, A. (2017). SymPy: symbolic computing in Python. PeerJ Computer Science, 3, e103.
10. Rocklin, M., & Terrel, A. R. (2012). Symbolic statistics with SymPy. Computing in Science & Engineering, 14(3), 88-93.
11. Sial, A. H., Rashdi, S. Y. S., & Khan, A. H. (2021). Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python. International Journal, 10(1).

12. Charis J. (2021), blog.jcharistech.com/2021/01/11/pypolars-data-analysis-with-pypolars-a-pandas-alternative/
13. Blank, S. (2009). Python Programming in OpenGL A Graphical Approach to Programming.
14. Dhruv, A. J., Patel, R., & Doshi, N. (2021). Python: The Most Advanced Programming Language for Computer Science Applications.
15. Plotly Technologies Inc. (2021). Collaborative data science. Plotly Technologies Inc. <https://plot.ly>
16. Fangohr, H., Kluyver, T., & DiPierro, M. (2021). Jupyter in Computational Science. *Computing in Science & Engineering*, 23(2), 5-6.
17. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & van Mulbregt, P. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, 17(3), 261-272.