

KONU 14

Dictionary (Sözlük)

Kazanımlar :

- Dictionary Nedir?
 - Nasıl Kullanılır?
-

– Dictionary (Sözlük)

Python'da Listeleri görmüştük. Listeler gibi Dictionary (Sözlük) lerde Array tipi bir değişkendir. Yani çok sayıda bilgiyi barındıran değişken tipleridir.

Python'da Array Tipleri

- 1) Listeler **: Sıralı ve değiştirilebilir
- 2) Tuple *: Sıralı ve değiştirilemez.
- 3) Set: Çok kullanılan bir tip değil.
- 4) Dictionary *: Sırasız ve değiştirilebilir.

Dictionary (Sözlük) Tanımlama

```
dict = {"brand": "Ford", "model": "Mustang"}  
print(dict)
```

“key” : “value”

Key’ler string olmak zorundadır!
Value’lar her tip olabilirler.

_ Dictionary(Sözlük)'de Eleman Erişme

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
  
x = thisdict["model"]  
print(x)
```

Listelerden farklı olarak
Burada “key” leri kullanarak
Değerlerimize erişiyoruz.

Liste vs Dictionary

```
liste = ["Ford", "Mustang", 1964]  
print(liste[1])
```

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
x = thisdict["model"]  
print(x)
```

_Dictionary (Sözlük) 'de Değer Değiştirme

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
  
thisdict["model"] = "Fiesta"  
  
print(thisdict["model"])
```

Dictionary(Sözlük)'de Yeni Değer Ekleme

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
  
thisdict["color"] = "red"  
print(thisdict["color"])
```

- Dictionary(Sözlük)'de Değer Silme

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
  
thisdict.pop("model")  
  
print(thisdict)
```

Dictionary(Sözlük) 'de Aradığım Key Var mı?

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
  
print("model" in thisdict)
```

— Dictionary (Sözlük) 'de for Döngüsü

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
  
for x in thisdict:  
    print(thisdict[x])
```

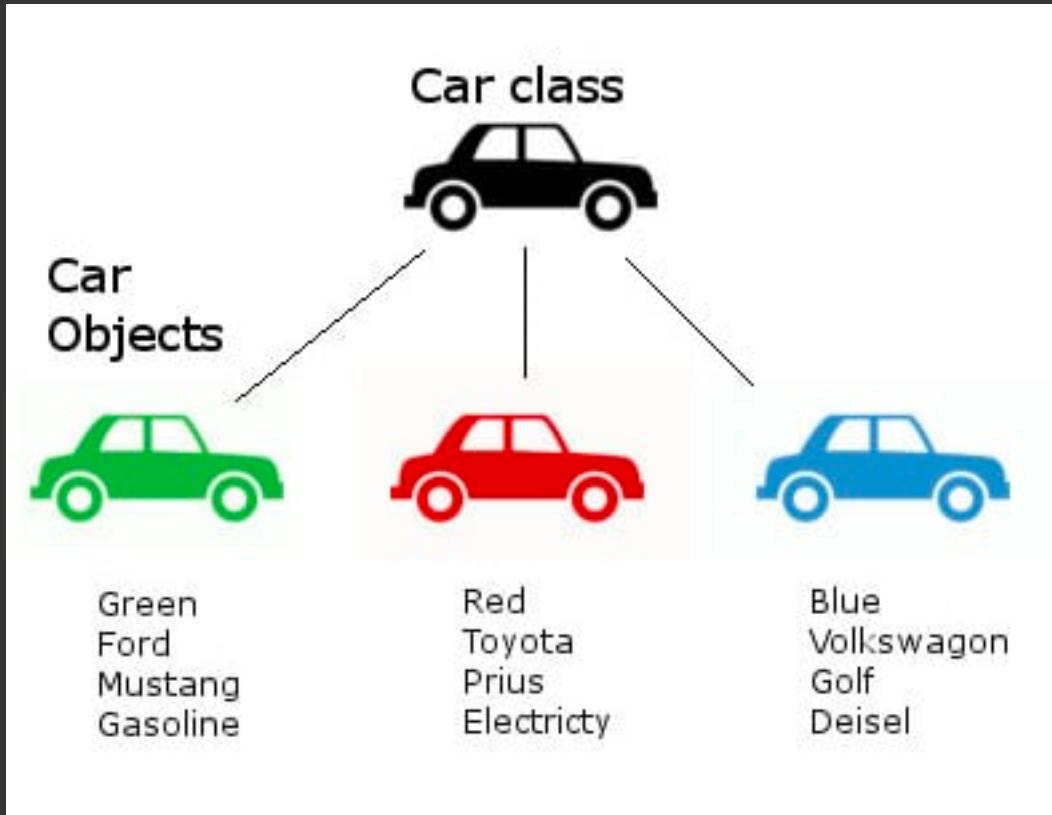
KONU 15

Class/Object

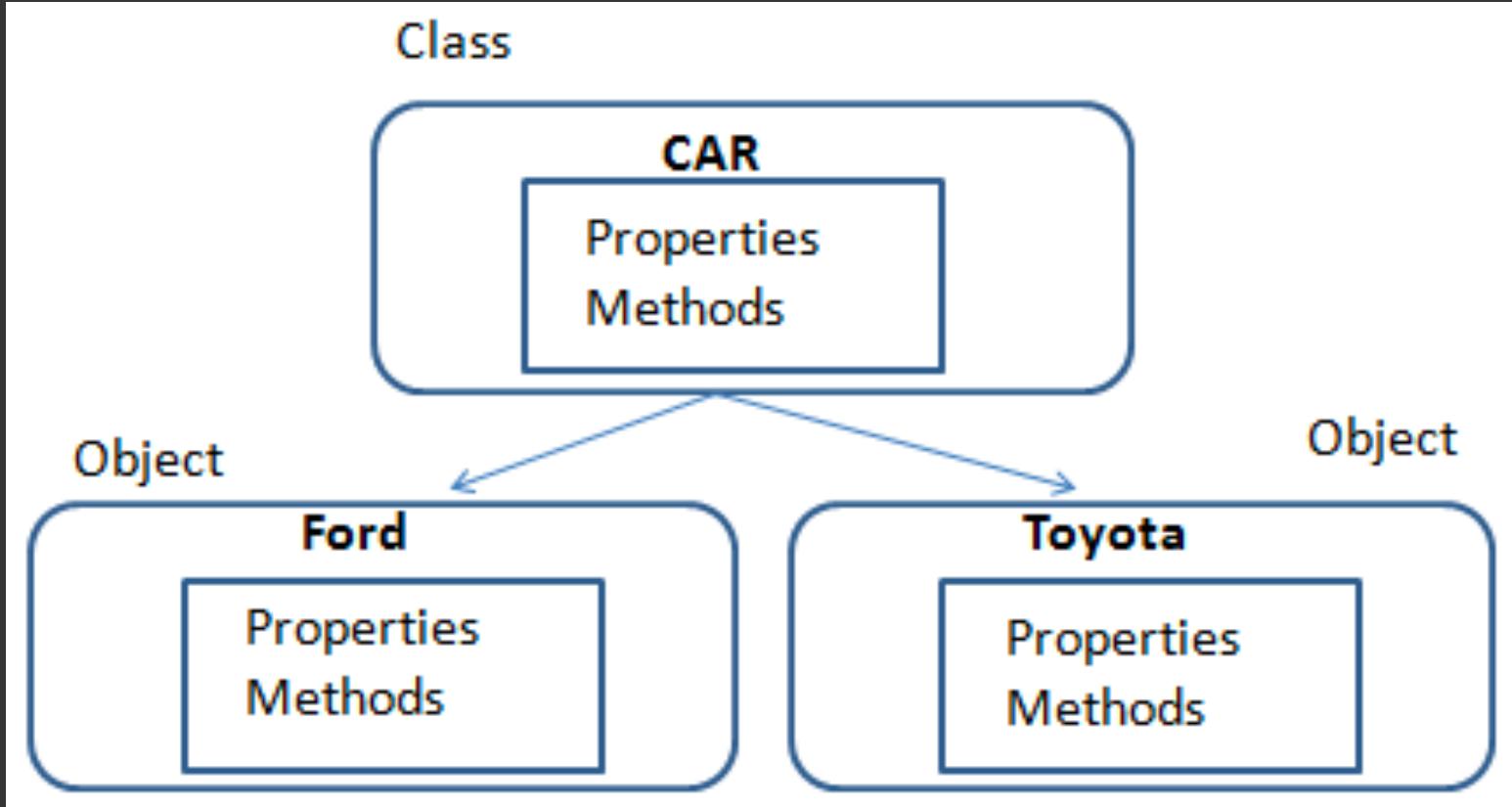
Kazanımlar :

- Class Nedir?
 - Obje Nedir?
 - Nasıl Kullanılır?
-

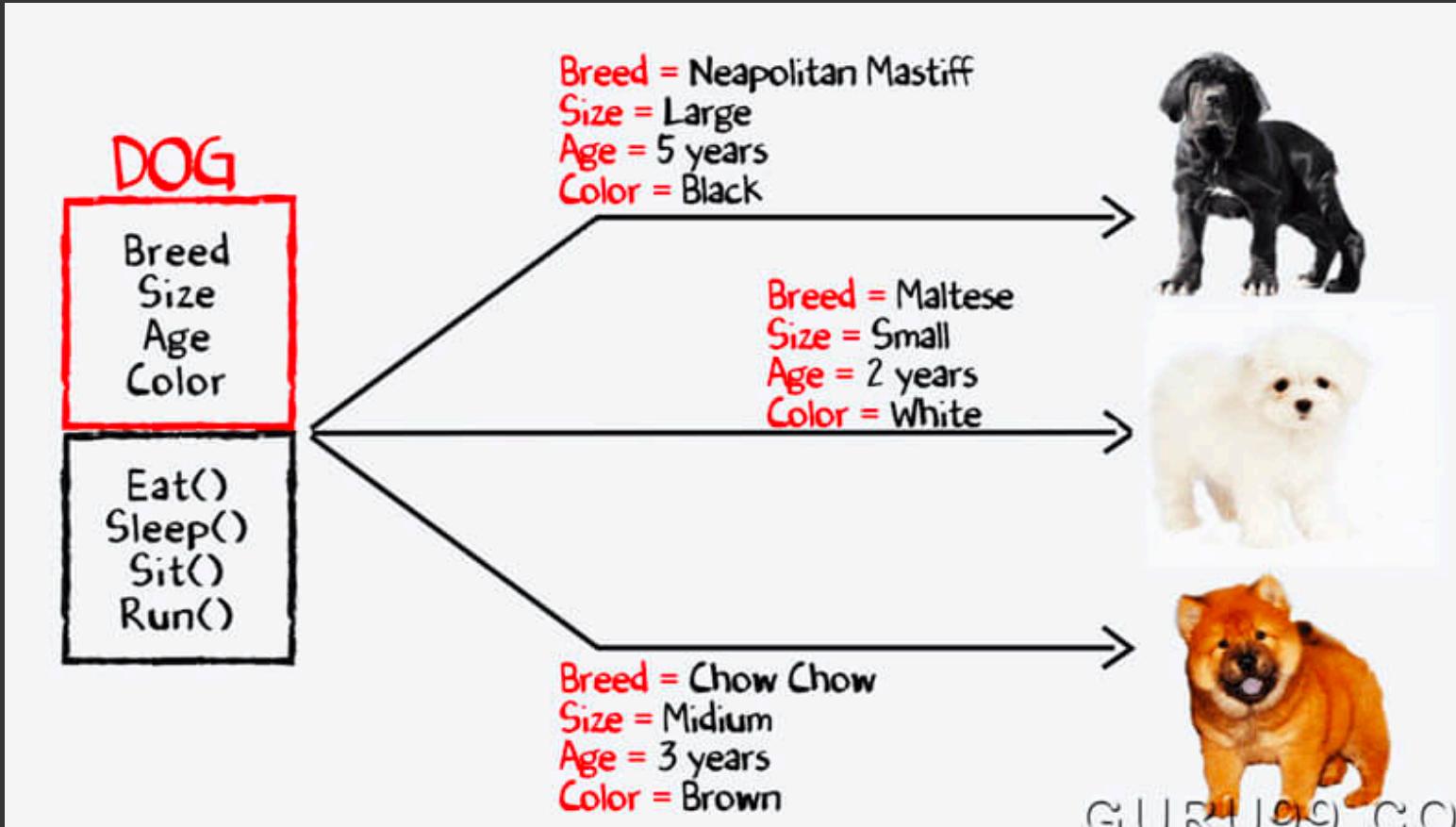
Class ve Objek Nedir?



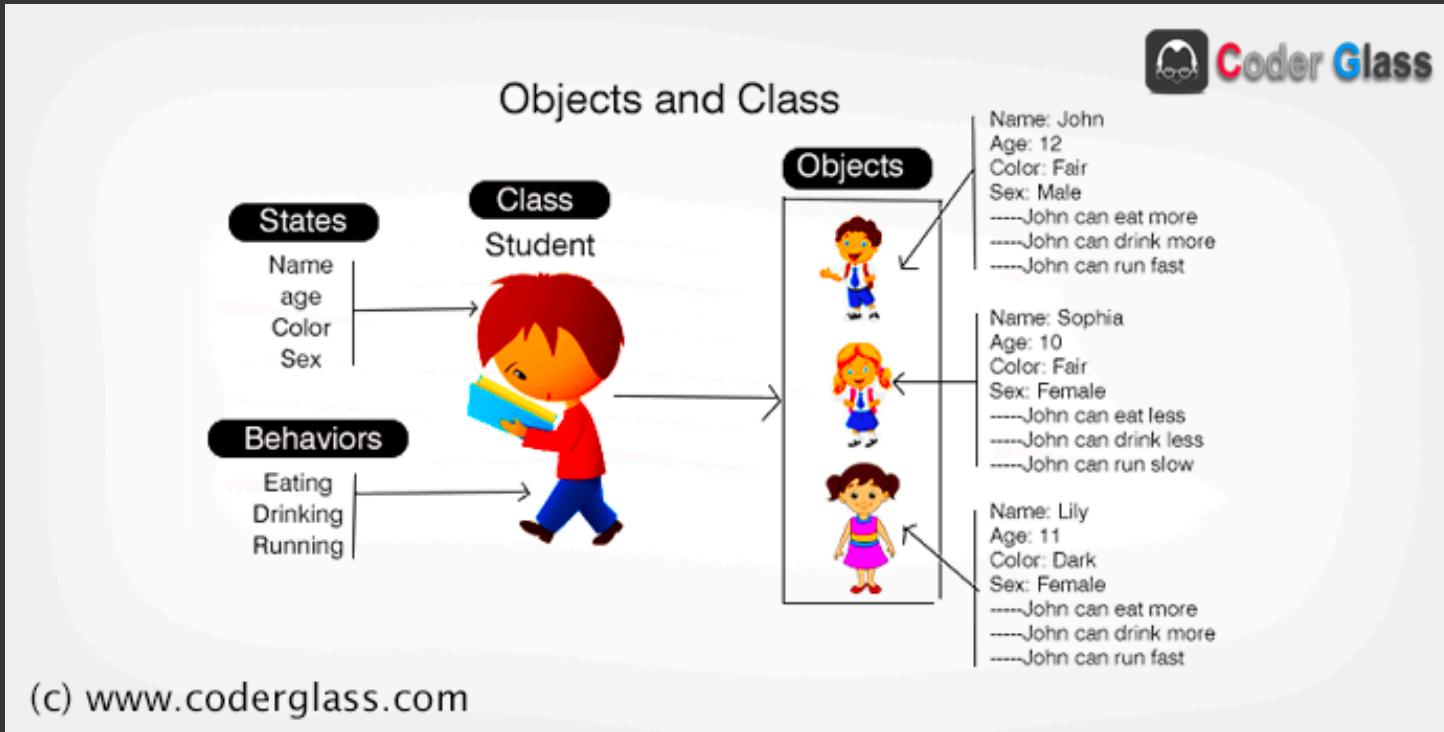
Class ve Objek Nedir?



Class ve Objek Nedir?



Class ve Objek Nedir?



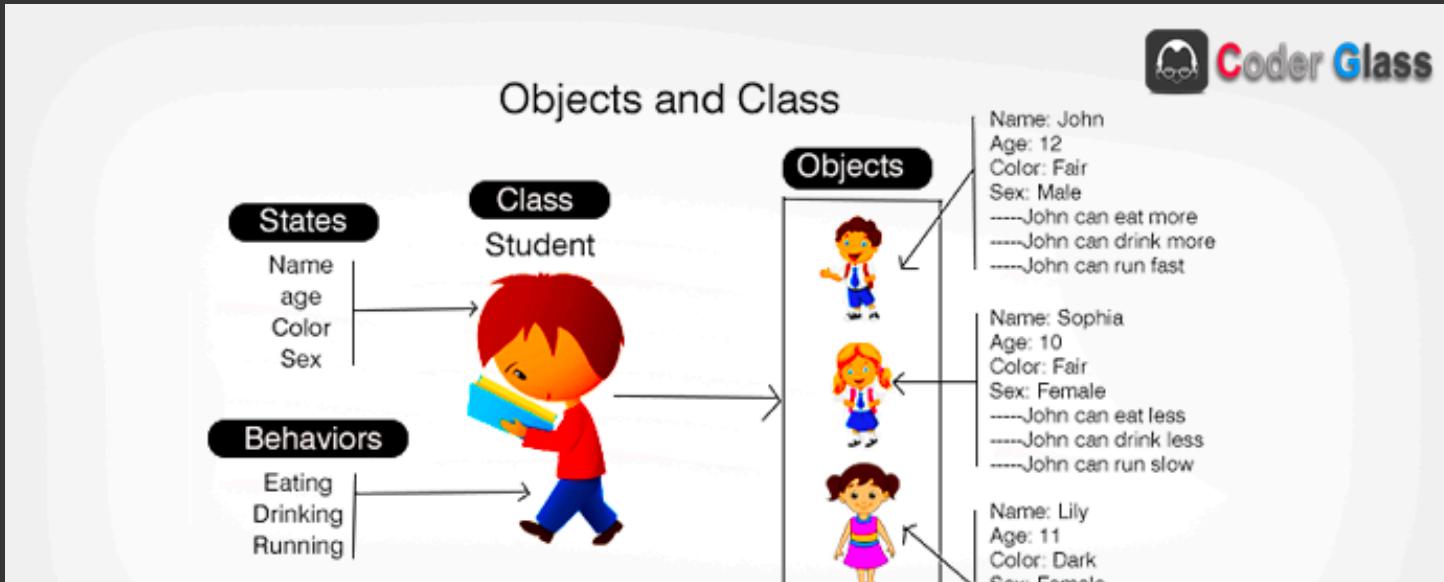
Coder Glass

— Class ve Objeler Nasıl Tanımlanır.

```
class MyClass: # CLASS  
    x = 5
```

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

__init__ fonksiyonu?



ÖNEMLİ
Objeleri Tanımlarken
Kullanılır!

__init__ fonksiyonu

```
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.nation = "White"

p1 = Person("Kaan", 36)
print(p1.name)
print(p1.age)
print(p1.nation)
```

__init__ fonksiyonu

```
class Person:
```

```
    def __init__(self, name, age):
```

```
        self.name = name
```

```
        self.age = age
```

```
        self.nation = "White"
```

__init__ metodu

Her yeni obje tanımlarken
otomatik çalışır

self nedir ?

Başka bir isim kullanabilir miyiz?

Neden self parametresini
doldurmadık?

```
p1 = Person("Kaan", 36)
```

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

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

```
print(p1.nation)
```

OUT : Kaan

OUT : 36

OUT : White

__init__ fonksiyonu

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
        self.nation = "White"  
  
p1 = Person(p1,"Kaan", 36)  
print(p1.name)  
print(p1.age)  
print(p1.nation)
```

Python'ın kendisi Objeyi self'e otomatik olarak atıyor.

Class vs Obje Özellikleri

```
class Person:  
    name = "İNSAN"  
  
    def __init__(self, name, age):  
        self.name = name  
  
        self.age = age  
  
        self.nation = "White"  
  
    print(Person.name)  
  
p1 = Person("Kaan", 36)  
print(p1.name)  
print(Person.name)
```

Class vs Obje Özellikleri

```
class Person:  
    name = "İNSAN"  
  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
        self.nation = "White"  
  
    print(Person.age)
```

Metodlar

Metodların Class yapılarına ait fonksiyonlar olduğunu ve bunların bildiğimiz `.append()`, `.upper()` metodları gibi kullanıldığını biliyoruz.

Şimdi bunların nasıl tanımlandığını ve kullanıldığını daha ayrıntılı olarak göreceğiz.

Metodlar

```
class Person:  
    name = "INSAN"  
    age = 20000  
    def __init__(self, name, mon):  
        self.name = name  
        self.nation = "White"  
        self.money = mon  
        self.age = 0  
  
    def buy(self, price):  
        if price > self.money:  
            return False  
        self.money -= price  
        return True  
  
    def growUp(self):  
        self.age += 1  
  
@staticmethod  
def speak():  
    print("KAAN")  
  
@classmethod  
def growUpHuman(cls, year):  
    cls.age = cls.age + year
```

```
print(Person.name, Person.age)  
p1 = Person("Kaan", 12000)  
print(p1.name, p1.nation, p1.money, p1.age, Person.name, Person.age)  
p1.growUp()  
print(p1.name, p1.nation, p1.money, p1.age, Person.name, Person.age)  
print(p1.buy(5000))  
print(p1.name, p1.nation, p1.money, p1.age, Person.name, Person.age)  
Person.growUpHuman(10)  
p1.speak()  
print(p1.name, p1.nation, p1.money, p1.age, Person.name, Person.age)
```

Objе, Class ve Ortak Metodlar

```
class Person:  
    name = "INSAN"  
    age = 200000  
    def __init__(self, name, mon):  
        self.name = name  
        self.nation = "White"  
        self.money = mon  
        self.age = 0  
  
    def buy(self, price):  
        if price > self.money:  
            return False  
        self.money -= price  
        return True  
  
    def growUp(self):  
        self.age += 1  
  
    @staticmethod  
    def speak():  
        print("KAAN")  
  
    @classmethod  
    def growUpHuman(cls, year):  
        cls.age = cls.age + year
```

```
print(Person.name, Person.age)  
p1 = Person("Kaan", 12000)  
  
p1.buy(1000)  
p1.growUp()
```

Objе Metodları

```
print(Person.name, Person.age)  
p1 = Person("Kaan", 12000)
```

```
Person.buy(1000)  
Person.growUp()
```

ERROR :
Objе Metodları Classlarla Çalışmaz
Neden ?

Objeler, Class ve Ortak Metodlar

```
class Person:  
    name = "INSAN"  
    age = 200000  
    def __init__(self, name, mon):  
        self.name = name  
        self.nation = "White"  
        self.money = mon  
        self.age = 0  
  
    def buy(self, price):  
        if price > self.money:  
            return False  
        self.money -= price  
        return True  
  
    def growUp(self):  
        self.age += 1  
  
    @staticmethod  
    def speak():  
        print("KAAN")  
  
    @classmethod  
    def growUpHuman(cls, year):  
        cls.age = cls.age + year
```

```
pl.buy(1000)  
print(Person.name, Person.age)  
pl = Person("Kaan", 12000)  
  
pl.buy(1000)  
pl.growUp()
```

Bu Örnek Çalışır mı ?
Açıklayınız?

Objе, Class ve Ortak Metodlar

```
class Person:  
    name = "INSAN"  
    age = 200000  
    def __init__(self, name, mon):  
        self.name = name  
        self.nation = "White"  
        self.money = mon  
        self.age = 0  
  
    def buy(self, price):  
        if price > self.money:  
            return False  
        self.money -= price  
        return True  
  
    def growUp(self):  
        self.age += 1  
  
    @staticmethod  
    def speak():  
        print("KAAN")  
  
    @classmethod  
    def growUpHuman(cls, year):  
        cls.age = cls.age + year
```

Person.growHuman(50)

```
print(Person.name, Person.age)  
p1 = Person("Kaan", 12000)
```

Person.growHuman(100)

Class Metodları

```
print(Person.name, Person.age)  
p1 = Person("Kaan", 12000)
```

p1.growHuman(1000)

ERROR:
Class Metodları objelerle
ÇALIŞMAZ!!

Objen, Class ve Ortak Metodlar

```
class Person:  
    name = "INSAN"  
    age = 200000  
    def __init__(self, name, mon):  
        self.name = name  
        self.nation = "White"  
        self.money = mon  
        self.age = 0  
  
    def buy(self, price):  
        if price > self.money:  
            return False  
        self.money -= price  
        return True  
  
    def growUp(self):  
        self.age += 1  
  
    @staticmethod  
    def speak():  
        print("KAAN")  
  
    @classmethod  
    def growUpHuman(cls, year):  
        cls.age = cls.age + year
```

```
print(Person.name, Person.age)  
p1 = Person("Kaan", 12000)  
Person.speak()  
p1.speak()
```

Ortak(Static) Metodları

DİKKAT:

Ortak Metodlar herhangi bir
cls veya self parametresi almaz.

Neden?

Bu özellik bize ne sağlıyor?

—

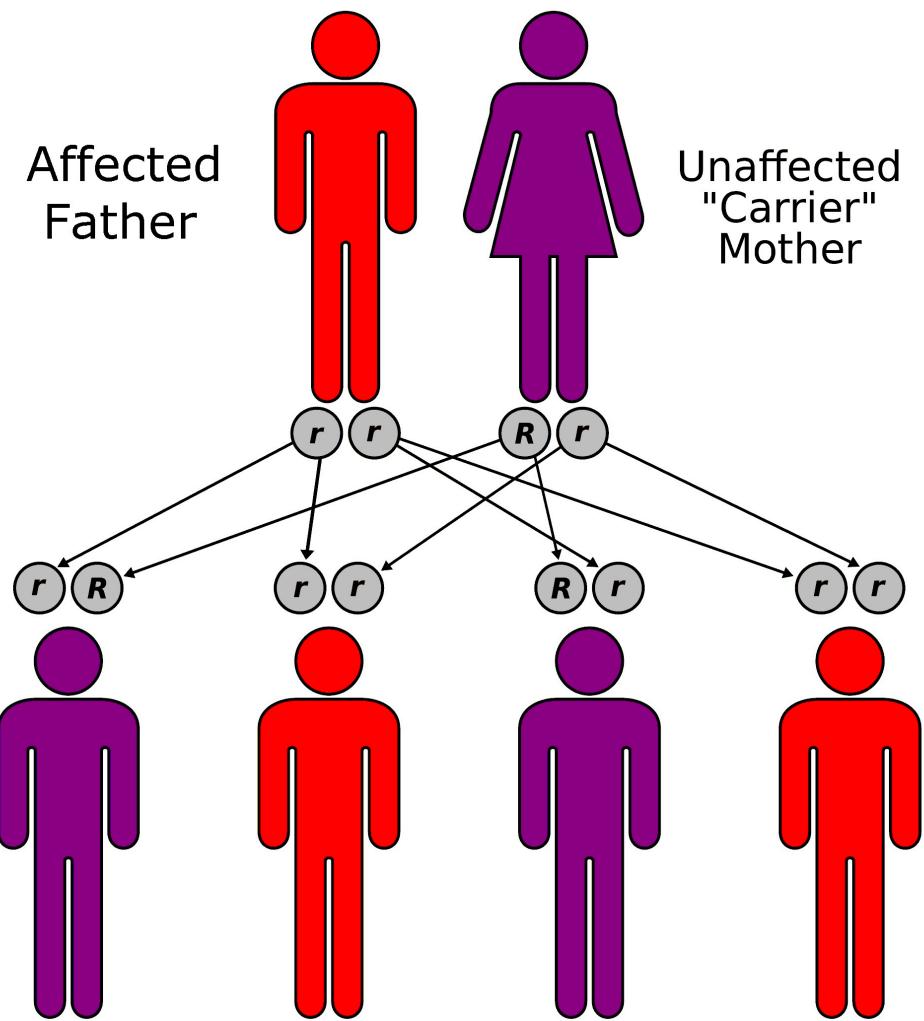
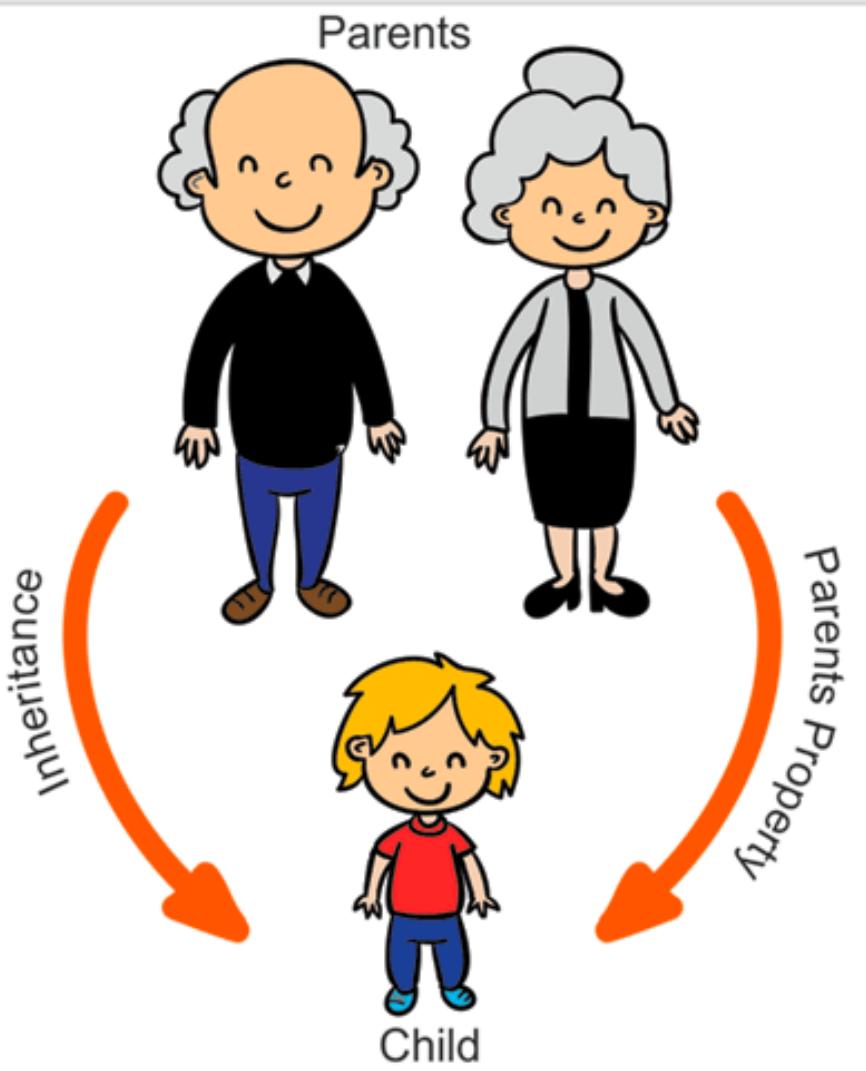
Örnek 1 'e başlıyoruz.

KONU 16

Inheritance
(Kalıtım)

Kazanımlar :

- Kalıtım Nedir?
 - Ne İşe Yararlar?
 - Nasıl Kullanılırlar?
-



Inheritance (Kalıtım) Nedir?



İnsanların Alt Class'ı
İnsanların özelliklerini taşıyor.



Polisler Class'ı

İnsanların Alt Class'ı
İnsanların özelliklerini taşıyor.



Öğrenciler Class'ı

İnsanların Alt Class'ı
İnsanların özelliklerini taşıyor.



Doktorlar Class'ı

Doktorların Alt Class'ı
Hem insanların hemde
doktorların özelliklerini taşıyor.

Kardiyoloji
Uzmanları Class'ı



Nöroloji
Uzmanları Class'ı

Inheritance (Kalıtım) Ne İşe Yarar?

Inheritance (Kalıtım) - 1

```
class Person:  
    name = "İNSAN"  
    age = 200000  
  
    def __init__(self, name, para):  
        self.name = name  
        self.nation = "White"  
        self.money = para  
        self.age = 0
```

```
    def growUp(self):  
        self.age += 1
```

```
class Student(Person):  
    student_id = 5  
  
    def study(self):  
        print("Studying")
```

```
p1 = Person("Ayşe", 5000)  
print(p1.name, p1.nation, p1.money)  
p1.growUp()
```

```
s1 = Student("Kaan", 1200)  
print(s1.name, s1.nation, s1.money, s1.student_id)  
s1.growUp()  
s1.study()
```

!!! DİKKAT !!!

`__init__` metodu tanımlamamamıza rağmen
Student Obje'sine İsim ve Para bilgilerini aktardık.

Person Class'ının özelliklerini alarak
Student Class'ı alt Class oluyor

Person Class'ına ait bir obje.

Student Class'ına ait bir obje.

Child Class'ta `__init__` tanımlamadığımızda, Parent Class'ın `__init__` metodunu kullandığını gördük.

Eğer Child Class'a `__init__` metodunu eklemiş olsaydık, Verdiğimiz isim ve para parametresi Otomatik olarak Parent Class'a gönderilir miydi?

Inheritance (Kalıtım) – 2

```
class Person:  
    name = "İNSAN"  
    age = 200000  
  
    def __init__(self, name, para):  
        self.name = name  
        self.nation = "White"  
        self.money = para  
        self.age = 0  
  
    def growUp(self):  
        self.age += 1  
  
class Student(Person):  
    id = 21928000  
    def __init__(self, name, money, yil):  
        Student.id = Student.id + 1  
        self.student_id = student.id  
        self.girisYil = yil  
        Person.__init__(self, name, money)  
        #super().__init__(self, name, money)  
  
    def study(self):  
        print("Studying")
```

```
pl = Person("Ayşe", 5000)  
print(pl.name, pl.nation, pl.money)  
pl.growUp()  
  
s1 = Student("Kaan", 1200, 2019)  
print(s1.name, s1.nation, s1.money, s1.student_id)  
s1.growUp()  
s1.study()  
  
s2 = Student("Baki", 3000, 2019)  
print(s2.name, s2.nation, s2.money, s2.student_id)  
s2.growUp()
```

KONU 17

Dates
(Tarih-Zaman Kütüphanesi)

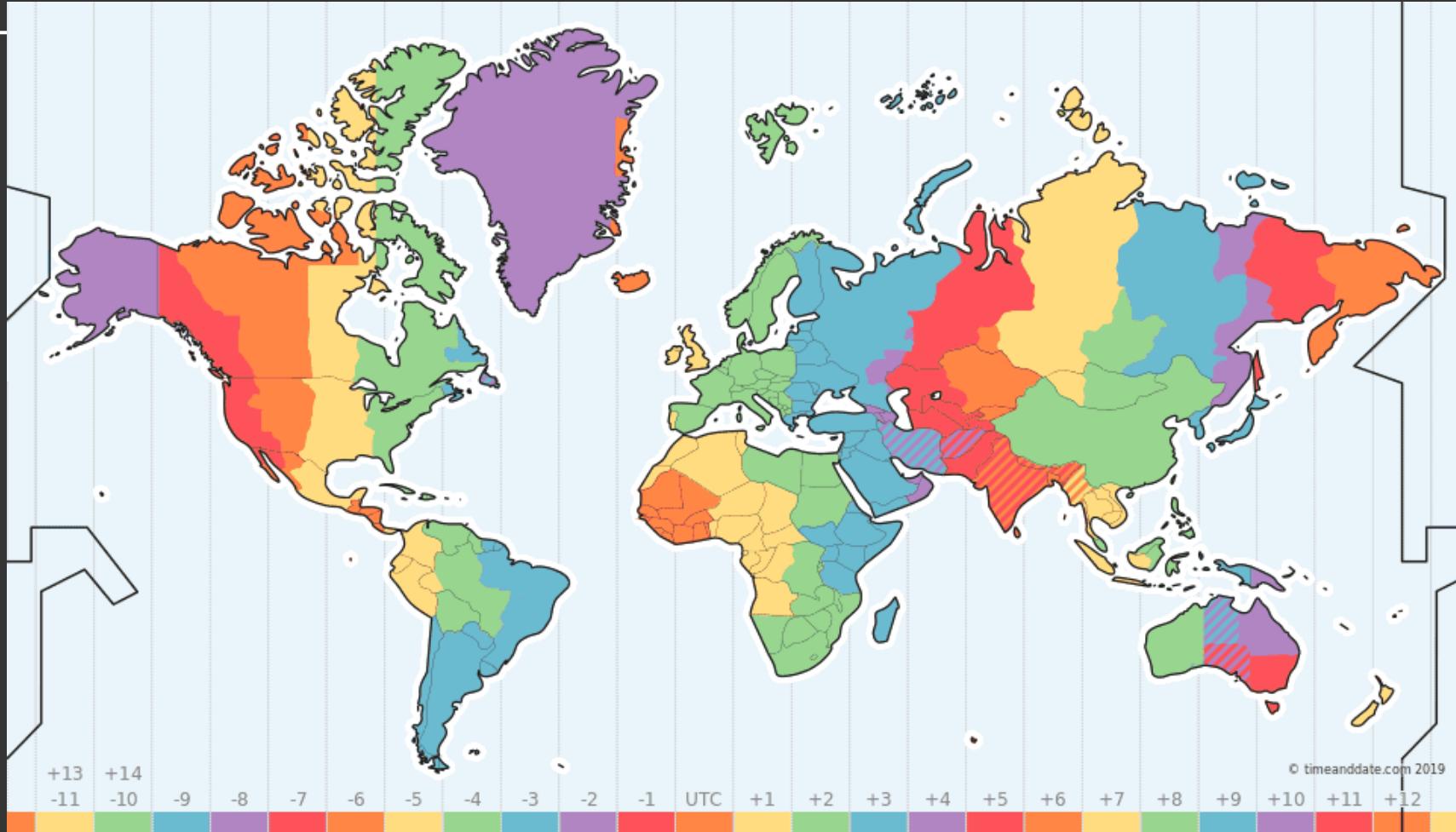
Kazanımlar :

- Yazılımlarda Zaman Kavramı
 - Python'da Tarih-Zaman
 - datetime Kütüphanesi ve Kullanımı
-

Yazılımlarda Zaman Kavramı?

Bilgisayarlar UTC saat dilimlerine göre saat kavramına hakimdirler.

Bilgisayarınızın yada telefonunuzun başka ülkelere gittiğinizde kendini otomatik ayarlaması bu sistemden dolayı kolayca gerçekleştirilebilir.



© timeanddate.com 2019

Python'da Tarih-Zaman?

Python'da zamanı tutan yerleşik(built-in) olarak gelen bir değişken tipi bulunmamaktadır.

Ancak bunun için built-in olarak gelen datetime modülü (kütüphane) mevcuttur

— datetime Modülü (Kütüphanesi) Kullanımı

```
import datetime

x = datetime.datetime.now()
print(x)
print(type(x))

print(x.year)
print(x.strftime("%A"))
print(x.strftime("%B"))
```

(Örnek : time.strftime("%d") = 30)

Sembol	Açıklama
%a	Kısaltılmış Gün İsmi
%A	Tam Gün İsmi
%w	0-6 arasında numaralandırılmış günler, 0 Pazar günü
%d	Ayın Günü
%b	Ayın kısa ismi
%B	Ayın tam ismi
%m	Ayın numarası 01-12
%y	Yıl, kısa versiyon
%Y	Yıl, uzun versiyon
%H	Saat 00-23
%I	Saat 00-12 (Analog)
%p	AM/PM
%M	Dakika 00-59

—

Örnek 2 'e başlıyoruz.

KONU 18

JSON

Kazanımlar :

- JSON Nedir?
 - Nerelerde Kullanılır?
 - Python'da Kullanımı
-

JSON Nedir?

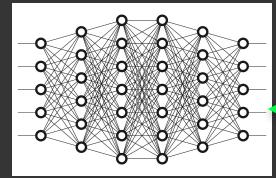
JSON (JavaScript Object Notation) Java
Script alt yapısı üzerine kurulmuş bir
veri aktarım formatıdır.

JSON programlama dillerinden bağımsız
evrensel bir veri tabanı olma yönünde
ilerleyen bir veri aktarım formatıdır.

JSON Veri Tipi

```
{  
    "Rehberim": [  
        {  
            "ID": "1",  
            "Adi": "Fatih",  
            "Soyadi": "Alkan",  
            "Telefon": "+905547740000",  
            "Email": "burakdalgic@gmail.com"  
        },  
        {  
            "ID": "2",  
            "Adi": "Burak",  
            "Soyadi": "Alkan",  
            "Telefon": "+905326550000",  
            "Email": "burakdalgic@gmail.com"  
        },  
        {  
            "ID": "3",  
            "Adi": "Canan",  
            "Soyadi": "Alkan",  
            "Telefon": "+905337770000",  
            "Email": "canandalgic@gmail.com"  
        }  
    ]  
}
```

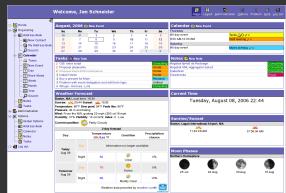
JSON Kullanım Örneği



{JSON}

JavaScript Object Notation

php



Json Veri Tipi Kullanımı

```
import json

x = '{ "name":"John", "age":30,
"city":"New York"}'

y = json.loads(x)

print(y["age"]) ← y artık Dictionary'dir
```

json kütüphanesinin loads fonksiyonu elimizdeki json stringlerini Python' Dictionary'lerine dönüştürür. Kullanımı Dictionary'ler ile aynıdır.

```
import json

x = {
  "name": "John",
  "age": 30,
  "city": "New York"
}

y = json.dumps(x) ← y artık JSON string'dir

print(y)
```

json kütüphanesinin dumps fonksiyonu elimizdeki Dictionary'i json string'ine dönüştürür.

—

Örnek 3 'e başlıyoruz.

KONU 19

Dosya İşlemleri

Kazanımlar :

- Dosya İşlemleri Nedir?
 - Nerelerde Kullanılır?
 - Python'da Kullanımı
-

Dosya İşlemleri

Dosya İşlemleri tüm yazılımlarda büyük rol oynamaktadır.

Python'da bunun ne kadar kolay olduğunu hep beraber inceleyeceğiz.

İlk Görev: Dosya Aşmak

Read - Dosya Okuma

```
f = open("demofile.txt", "r")
```

demofile.txt dosyasını Okuma Modunda açıp, f değişkenine file objesini atar.

Bu modda dosya içeriği değiştirilemez

Write - Dosya Üzerine Yazma

```
f = open("demofile.txt", "w")
```

demofile.txt dosyasını Yazma Modunda açıp, f değişkenine file objesini atar.

Bu modda dosyanın üzerine yazılır. Okuma yapılamaz. Dosya bulunmuyorsa yeni dosya oluşturulur.

Append - Dosyaya Ekleme

```
f = open("demofile.txt", "a")
```

demofile.txt dosyasını Ekleme Modunda açıp, f değişkenine file objesini atar.

Bu modda dosyaya yeni satır olarak içerik eklenir. Okuma ve Üzerine yazma yapılamaz.

Create - Dosya Oluşturma

```
f = open("demofile.txt", "x")
```

demofile.txt dosyasını oluşturur eğer dosya mevcutsa hata döndürür.

Bu modda yazma veya okuma yapılamaz.

Dosya Okuma Örnekleri

```
f = open("demofile.txt", "r")  
print(f.read())
```

Tüm dosyayı okur.

```
f = open("demofile.txt", "r")  
print(f.read(10))
```

İlk 10 karakterini okur.

```
f = open("demofile.txt", "r")  
print(f.readline())  
print(f.readline())
```

Satır satır okur.

Her çağrırlığında yeni satırı
okur!!!

```
f = open("demofile.txt", "x")  
for x in f:  
    print(x)
```

Satır satır tüm satırları yazdırıldı.

Dosyaya Yazma Örnekleri

```
f = open("demofile.txt", "w")  
f.write("Merhaba Dünya!!!")  
f.close()
```

DİKKAT:

Dosyanın içindekileri silip üzerine
yazdırın!

```
f = open("demofile.txt", "a")  
f.write("Merhaba Yeni Satır!!!")  
f.close()
```

Dosyaya yeni satır eklediniz.

—

Örnek 4 'e başlıyoruz.

KONU 20

Try - Except Yapısı

Kazanımlar :

- Try-Except Nedir?
 - Nerelerde Kullanılır?
 - Python'da Kullanımı
-

Try-Except Yapısı

Try-Except yapısı Python'da önemli bir yere sahiptir.

Yazılımlarda her zaman hatalarla karşılaşabileceğini hepimiz biliyoruz. Önemli olan konu bu hatalarla karşılaşınca ne tepkiler vereceğimizdir.

İşte bu noktada Try-Except devreye giriyor.

Try-Except Yapısı

```
try:  
    print(x)  
except NameError:  
    print("x Tanımlı değil!")  
except:  
    print("Bilinmeyen Hata")
```

Hatanın İsmi



```
try:  
    print(x)  
except:  
    print("Bilinmeyen Hata")
```

Try-Except Kullanım Örneği

```
def yil_input():
    yil = int(input("Doğum Yılı: "))
    return yil

print(yil_input())
```

İki programın arasındaki
fark nedir?

```
def yil_input():
    while True:
        try:
            yil = int(input("Doğum Yılı: "))
            break
        except:
            print("Yanlış Giriş.")

    return yil

print(yil_input())
```

—

Örnek 5 'e başlıyoruz.