2019-2020

Veri Yapıları ve Programlama

Ders Notları

Öğr. Gör. Hakan Can ALTUNAY
ÇARŞAMBA TİCARET BORSASI MESLEK YÜKSEKOKULU

Python'da Sayılar ve Aritmetik İşlemler

Python'un çalışma ekranında *(IDLE)* aritmetik işlem yaparken bir değişkene ihtiyaç duyulmadan rakamlar doğrudan kullanılabilir. Burada dikkat edilmesi gereken, sonucun yanlış çıktığı zamanlarda kullanılan sayı tiplerinin sonuca uygun olmamasıdır.

`2 ** 3 # Çıktı: 8`	`10 % 2 # Çıktı: 0`
`10 / 3 # Çıktı: 3.3… (Python 3)`	`10 / 3 # Çıktı: 3 (Python 2)`

Not: Aritmetik işlemlerde matematiksel dildeki kurallar birebir geçerlidir. Buna göre bölme işleminde ondalıklı bir sonuç çıkması gerekiyorsa:

- İşleme giren sayılardan biri veya her ikisinin de ondalık olarak yazılması gerekmektedir.[1]

Python'da Değişkenler ve Veri Çıkışı

Bir veriyi kendi içerisinde depolayan birimlere değişken adı verilir. Değişken tanımlamada Python, veri tipi olarak karakter için string, tamsayı için int, ondalık için float komutları kullanılır. Bu veri tipleri gerektiğinde birbirlerine dönüştürülebilirler.

print() Komutu

Python'da ekrana yazdırma komutu olarak "print()" kullanılır. Ekrana yazdırma komutu diğer dillerde olduğu gibi üç farklı şekilde kullanılabilir. Bunlar:

1) Tırnak işareti içerisindeki ifadenin ekrana doğrudan yazdırılması,
2) Değişkene ait olan değerin ekrana doğrudan yazdırılması,
3) Formatlı yazım şeklinde yazdırılması,

şeklindedir.

```
a = 5
b = 3

print("%s ile %s çarparsak %s eder." %(a, b, a * b)) # Formatlı biçim 1.
print("{} ile {} çarparsak {} eder.".format(a, b, a * b)) # Formatlı biçim 2.
# 5 ile 3 çarparsak 15 eder.
```

Python'da Veri Girişi

input() Komutu

Python'da klavyeden veri girmek için kullanılan fonksiyondur. Tırnak içerisinde kullanılan değer bilgilendirmek için kullanılır. Alınan değer karakter dizisidir. Kullanımı aşağıda gösterildiği gibidir:

```
parola = input("Lütfen parolayı giriniz: ")
print("Parola: {}".format(parola))
```

`a = input("1. sayıyı girin: ") # 25`
`b = input("2. sayıyı girin: ") # 15`
`print(a + b) # 2515`
	`a = int(input("1. sayıyı girin: ")) # 25`
	`b = int(input("2. sayıyı girin: ")) # 15`
	`print(a + b) # 40`

[1] Bu durum Python 2'de geçerlidir. Python 3'ten itibaren sayılar tamsayı olsa da bölümün sonucu ondalıklıdır.

Yukarıdaki örneklere bakıldığında input fonksiyonu ile girilen karakter dizisi olarak okunmaktadır. Bu yüzden sayısal ifadelerin çevrilmesi için int() gibi bir çevirici komutun kullanılması gerekir.

Python'da Kaçış Dizileri

Kendilerinden sonra gelen karakterlerle birleşerek karakterin farklı veya özel bir anlam kazanmasını sağlayan işaretlere kaçış dizisi adı verilir.

```
print('Linux\'un faydaları')
print("Gökhan, \"Yarın gidiyorum!\" dedi.")
```

Python dilinde tür dönüşümü yapmak mümkündür. Aşağıda üç farklı veri tipi birbirleri arasında tür dönüşümüne uğramıştır.

```
a = 23
print(str(a))
# Çıktı: 23
```

```
a = 23
print(float(a))
# Çıktı: 23.0
```

```
print(float("34"))
# Çıktı: 34.0
```

Python'da Koşula Bağlı Durumlar

If yapısı kullanılırken mutlaka şartın sonuna iki nokta koyulmalıdır. *(:)*

```
if şart:
        # şart doğru
        # ise yapılacaklar
```

```
if şart:
        # şart doğru
        # ise yapılacaklar
else:
        # şart yanlış
        # ise yapılacaklar
```

```
if şart1:
        # şart1 sağlanırsa
        # yapılacak işlemler
elif şart2:
        # şart1 sağlanmazsa
        # şart2 kontrol
        # edilir, doğru ise
        # yapılacak işlemler
else:
        # şart1 ve şart2
        # sağlanmıyor ise
        # yapılacak işlemler
```

Örnek: Klavyeden girilen iki sayı ve seçenek için dört işlem yapan programı if yapısıyla hazırlayınız.

```
a = int(input("1. sayı: "))
b = int(input("2. sayı: "))

print("1) Toplama")
print("2) Çıkarma")
print("3) Çarpma")
print("4) Bölme")

islem = int(input("Seçenek seçiniz: "))

if islem == 1:
        print("Sonuç:", a + b)
elif islem == 2:
        print("Sonuç:", a - b)
elif islem == 3:
        print("Sonuç:", a * b)
else:
        print("Sonuç:", a / b)
```

Python'da Döngü Yapıları

1. while Döngüsü

Belirtilen şart sağlandığı sürece döngü içerisindeki işlem(ler)in tekrar edildiği yapıdır.

```
while şart:
      # şart doğru
      # ise yapılacaklar
```

```
a = 0
while a < 100:
    a += 1
    print(a)
```

Örnek: Klavyeden girilen cevaba göre ekrana "Yanlış cevap!" yazdıran programı hazırlayınız.

```
soru = input("Python mu?, C mi?")
while soru != "Python":
    print("Yanlış cevap!")
```

Yukarıdaki programı çalıştırdığımızda sorulan soruya "Python" cevabını vermezsek program, biz müdahale edene kadar ekrana "Yanlış cevap!" çıktısını vermeye devam edecektir. Çünkü biz Python'a *"Soru değişkeninin cevabı "Python" olmadığı müddetçe ekrana "Yanlış cevap!" çıktısını ver!"* komutunu verdik. Aynı soruyu if yapısı kullanarak çözersek if ile while arasındaki fark ortaya çıkacaktır.

2. for Döngüsü

Python dilinde for döngüsünde başlangıç, bir bitiş, bir artış miktarı ve bir döngü değişkeni mevcuttur. For döngüsü Python dilinde range ve in döngü deyimleri ile birlikte genel kullanımı aşağıdaki gibidir. Artış miktarı yazılmazsa birer birer artış yapar.

```
for i in range(1, 100):
    print(i)
```

```
for a in "çarşamba":
    print(a)
```

3. range() Komutu

Bu fonksiyon Python dilinde sayı aralıklarının belirlenmesi için kullanılır. Parantez içerisinde kullanılan ilk ifade başlangıç noktasını, ikinci ifade bitiş noktasını, üçüncü ifade ise artış miktarını gösterir. Eğer range() komutunun içerisinde tek bir değer varsa bu daima bitiş noktasıdır. Bu durumda başlangıç noktası her zaman sıfır olarak kabul edilir.

```
print(range(100))
# Çıktı: range(0, 100)
```

```
print(range(100, 200))
# Çıktı: range(100, 200)
```

```
print(range(1, 100, 2))
# Çıktı: range(1, 100, 2)
```

4. len() Komutu

Bu fonksiyon karakter dizilerinin *(string)* uzunluğunu gösteren fonksiyondur.

```
a = "Samsun"
print(len(a)) # Çıktı: 6
```

Örnek: Klavyeden girilen parola değeri sekiz karakterden az ise ekrana "Parola en az sekiz karakter olmalıdır!" şeklinde mesaj veren, parola sekiz karakterden fazla olduğunda "Parolanız aktif edilmiştir!" mesajını ekrana yazdıran programı hazırlayınız.

```
parola = input("Lütfen parola giriniz: ")
if len(parola) < 8:
    print("Parola en az 8 karakter olmalıdır!")
else:
    print("Parolanız aktif edilmiştir!")
```

5. break Komutu

Bir döngüyü sona erdirmek için kullanılan komuttur. Genellikle döngü yapılarının içerisinde "if" şart ifadesi ile birlikte kullanılır.

Örnek: Klavyeden girilen kullanıcı adı bilgisi "carsamba", parola bilgisi "myo" olduğunda "Giriş başarılı!" mesajını ekrana yazdıran, aksi durumda "Kullanıcı adı veya parola yanlış! Lütfen tekrar deneyiniz." mesajını ekrana çıkartan programı hazırlayınız.

```
kullaniciAdi = "carsamba"
parola = "myo"

while True:
    a = input("Kullanıcı adı: ")
    b = input("Parola: ")
    if a == kullaniciAdi and b == parola:
        print("Giriş başarılı!")
        break
    else:
        print("Kullanıcı adı veya parola yanlış!")
```

6. continue Komutu

Bu deyim, döngü içerisinde kendisinden sonra gelen bütün komutların es geçilerek döngünün başına dönülmesini sağlayan komuttur.

Örnek: Klavyeden bir sayı girilmektedir. Bu sayı en fazla üç basamaklı olmalıdır. Eğer üç basamaktan daha büyük bir sayı girilirse uyarı mesajı veren, diğer durumlarda programı devam ettiren, ayrıca klavyeden "iptal" yazıldığında programı sonlandıran yapıyı hazırlayınız.

```
while True:
    s = input("Bir sayı giriniz: ")
    if s == "iptal":
        break
    if len(s) <= 3:
        continue
    print("En fazla 3 basamaklı bir sayı giriniz!")
```

Not: "in" ifadesi Python dilinde for döngüsü ile birlikte kullanılır. İfadeye "*içerisinde*" anlamı katar.

Python'da Listeler, Demetler ve Sözlükler

a. Listeler (List)

Python dilinde birden fazla farklı veri tipini bir arada tutabilen yapılar mevcuttur. Liste ifadesi farklı veri tiplerini içerisinde barındıran, üzerinde ekleme, çıkarma, güncelleme vb. gibi işlemlerin gerçekleştirilebildiği yapılardır. Listeler tanımlanırken Türkçe karakterler kullanılamaz ve listeyi oluşturan elemanlar köşeli parantezler (*[ ]*) içerisine alınır. Aşağıda örnek bir liste tanımlanmıştır:

```
liste = [24, 32, 46, "Ali", "Veli", 10]
```

Python dili sıralama yöntemi ile ilgili olarak şu kuralı uygular: Liste ve diğer bütün veri tipleri için ilk ögeyi sıfırdan başlatır.

Listenin herhangi bir elemanının değerini ekranda görmek için o elemana ait index numarası köşeli parantez içerisine liste adı ile birlikte yazılır.

```
liste[2]
```

```
# Listeden 46 değeri seçilir.
```

Python dili elemanları numaralandırırken sıfırdan başlasa da elemanları sayarken 1'den başlar. Yani bir listenin eleman sayısını öğrenmek için len() komutunu aşağıdaki gibi kullandığımızda listenin eleman sayısı 1'den başlayarak sayılır.

```
len(liste) # Çıktı: 6
```

Python dilinde listenin herhangi bir noktasına eleman eklemek için insert() metodu kullanılır.

```
liste = [24, 32, 46, "Ali", "Veli", 10] # Listeyi oluşturduk.
liste.insert(1, "Ahmet") # 1. index numarasına "Ahmet" verisini ekledik.
# Çıktı: [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
```

Not: Python dilinde listenin sonuna eleman eklemek için append() metodu kullanılır.

Listeyi uzatmak yani genişletmek için extend() metodu kullanılır.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
yeniListe = ["Ayşe", 5, 8]
liste.extend(yeniListe)
# Çıktı: [24, "Ahmet", 32, 46, "Ali", "Veli", 10, "Ayşe", 5, 8]
```

Aynı işlemi artı operatörünü kullanarak da gerçekleştirebiliriz.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
yeniListe = ["Ayşe", 5, 8]
liste += yeniListe
# Çıktı: [24, "Ahmet", 32, 46, "Ali", "Veli", 10, "Ayşe", 5, 8]
```

Olan bir listeden herhangi bir elemanı çıkarmak için remove() metodu kullanılır. Remove metodunun içerisine elemanın değeri aynen yazılır. Fakat listenin içerisinde aynı değere sahip birden fazla eleman varsa remove() metodu listedeki ilk bulunan değeri siler.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
liste.remove("Veli")
# Çıktı: liste = [24, "Ahmet", 32, 46, "Ali", 10]
```

Listeden silinen herhangi bir elemanın değerini ekrana yazdırmak için pop() metodu kullanılır.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
liste.pop(2)
# Çıktı: liste = [24, "Ahmet", 46, "Ali", "Veli", 10]
```

Liste içerisindeki herhangi bir elemanın listenin kaçıncı sırasında olduğunu görmek için elemanın değeri ile birlikte index() metodu kullanılır.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
liste.index("Ahmet") # Çıktı: 1
```

Listedeki elemanları alfabetik olarak sıralamak için sort() metodu kullanılır.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
liste.sort() # Listeyi alfabetik sıraya göre sıralar.
```

Listedeki ögelerin sırasını tersine çevirmek için reverse() metodu kullanılır.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
liste.reverse() # Listeyi sondan başa doğru sıralar.
```

Bir listenin içerisindeki herhangi bir elemanın o liste içerisinde kaç kere kullanıldığını bulmak için count() metodu kullanılır.

```
liste = [24, "Ahmet", 32, 46, "Ali", "Veli", 10]
liste.count(5) # Listenin içerisinde kaç tane 5 sayısı olduğunu bulur.
```

Not: append ve insert komutları ile listeye yalnızca bir adet eleman ekleyebiliyorduk. Listede istediğimiz yere istediğimiz kadar eleman yerleştirmek için iki nokta üst üste (**:**) insert komutunun işlevini yerine getirebiliriz. İki nokta üst üste işareti köşeli parantezler ile birlikte kullanılır ve birinci parametre eklenecek index numarasını, ikinci parametre ise eklenecek index numarasından itibaren eklenecek olan listeden silinecek eleman sayısını belirtir. Aşağıda listenin sıfırıncı index numarasına dört eleman eklenmiştir ve eklenecek index numarasından itibaren listede olan 1 değer silinmiştir:

```
liste = [24, "Ahmet", 46, "Ali", "Veli", 10]
liste[0:1] = ["Can", 20, "Gökhan", "Furkan"]
# 1. parametre: sıra // 2. parametre: silinecek eleman sayısı
# Çıktı: ["Can", 20, "Gökhan", "Furkan", "Ahmet", 46, "Ali", "Veli", 10]
```

b. Demetler (Tuple)

Listelerle benzer yapıdadırlar. Fakat listeler üzerinde oynamalar yapılabilirken demetler üzerinde tanımlama yapıldıktan sonra bir daha değişiklik yapılamaz.

```
demet1 = ("Ali", "Veli", 3, 10)
demet2 = ("Ali", "Cem", "Can")
```

Herhangi bir ifadenin veri tipini öğrenmek için type() metodu kullanılır.

```
demet = ("Ali", "Veli", 3, 10)
type(demet) # Çıktı: type 'tuple'
```

Python dilinde demet ifadesinde iki istisna vardır. Bunlar boş bir demet oluşturmak ve bir elemanlı demet oluşturma ifadeleridir.

```
# Boş bir demet oluşturmak için >> demet = ()
# Tek elemanlı demet oluşturmak >> demet = ("Su",)
```

Yukarıda anlatılan işleme packing *(demetleme)* adı verilir.

c. Sözlükler

Sözlük ifadesi Python dilinde anahtar ve o anahtara ait değer arasında bağ kuran bir veri tipidir. Anahtar ve değer birbirlerinden iki nokta üst üste operatörü ile ayrılır. Sözlükler tanımlanırken süslü parantez kullanılır. Geniş veritabanlarında herhangi bir elemana ait anahtar ve değerin alt alta sıralanmış bir biçimde çıkması için ilk elemanın yanında bulunan virgülden sonra "ENTER" tuşuna basılır. Her elemandan sonra bu işlem tekrar edilir.

```
telefon = {"Ahmet": "0533 123 45 67", "Salih": "0533 123 45 67"}
```

Sözlüklerde herhangi bir elemanın değerini görmek için o elemana ait anahtar adını yazmalıyız.

```
telefon["Salih"] # Çıktı: 0533 123 45 67
```

Sözlüğe yeni bir eleman eklemek aşağıdaki gibidir;

```
telefon["Zeki"] = "0555 444 33 22"
```

Sözlükteki herhangi bir elemanın değerini değiştirmek aşağıdaki gibidir;

```
telefon["Zeki"] = "0533 123 45 67"
```

Liste üzerindeki herhangi bir elemanı silmek için o elemana ait ayırt edici özellik olan anahtar ifadesini "del" komutu ile birlikte kullanırız.

```
del telefon["Ahmet"] # Ahmet anahtarı ve değeri varsa silinir.
```

Bir sözlükteki bütün elemanları silmek için clear() metodu kullanılır.

```
telefon.clear()
```

d. Sıralı Sözlükler *(Ordered List)*

Listeler ve demetler sıralı veri tipi iken sözlükler sırasız veri tipidir. Herhangi bir sıra kavramı olmadığı için veritabanlarında bir elemanın aranması, çağrılması ve çalıştırılması sırasında çeşitli zorluklar çıkmaktadır. Bu nedenle sözlüklerin "*OrderedDict*" komutunu kullanarak sıralı hale getiririz. Bu modül sayesinde sıralı sözlükler bir liste içerisindeki iki ögeli demetler olarak da adlandırılırlar. Bu demetler içerisindeki ilk öge sözlük anahtarını, ikinci öge ise sözlük değerlerini temsil etmektedir.

Not: Bu modülü kullanabilmek için koda "collections" kütüphanesini dahil etmemiz gerekmektedir.[2]

```
from collections import *

personel = {"Ahmet": "19.01.2013", "Mehmet": "21.03.2015"}
personel = OrderedDict([("Ahmet", "19.01.2013"), ("Mehmet", "21.03.2015")])
```

Boş sıralı bir sözlük oluşturmak için aşağıdaki komutu gireriz. Daha sonra oluşturduğumuz bu sözlüğe eleman eklenmesini aşağıdaki gibi yaparız;

```
from collections import *

personel = OrderedDict()
personel["Ahmet"] = "0533 123 45 67"
personel["Mehmet"] = "0541 111 33 22"
personel["Selin"] = "0530 000 00 00"
```

Daha sonra çalışma alanına "`print(personel)`" komutunu yazıp "ENTER" tuşuna basarsak yazdığımız sırada sözlüğün karşımıza çıktığını görürüz.

Bir sözlükteki anahtarları görmek için keys(), değerleri görmek için ise values() metodunu kullanırız.

```
personel.keys()
# Çıktı: ["Ahmet", "Mehmet", "Selin"]
```

```
personel.values()
#     Çıktı:     ["19.01.2013",     "21.03.2015",
"20.02.2016"]
```

Python'da Fonksiyonlar

Python dilinde fonksiyon tanımlamak için "`def`" anahtar sözcüğü kullanılır. İlk önce "`def`" anahtar sözcüğünü, daha sonra fonksiyonun adını yazarız. Fonksiyon adlandırılırken Türkçe harf kullanılmaz. Hazırlanan fonksiyonlar onlara verilen isimler ile program içerisinde mutlaka çağrılmalıdırlar.

```
def fonksiyonAdi():
    # Fonksiyon içi işlemler
```

```
def fonksiyon_adi():
    print("Merhaba!")
```

Yukarıdaki kod bloklarında programın çalışmasını sağlayan, "`fonksiyon_adi()`" komut satırıdır.

[2] Bu durum Python 3 sürümlerinde geçerlidir.

Örnek: Klavyeden girilen bir tam sayının tek veya çift olduğunu ekrana yazdıran programı fonksiyon kullanarak hazırlayınız.

```
def tek():
    print("Girilen sayı tek sayıdır." )

def cift():
    print("Girilen sayı çift sayıdır.")

sayi = int(input("Sayıyı giriniz: "))
if sayi % 2 == 0:
    cift()
else:
    tek()
```

Fonksiyonlarda Parametre Kullanımı

Bir fonksiyon tanımlarken fonksiyon adını belirledikten sonra bir parantez işareti kullanıyoruz. Bu parantez işaretleri yapılacak işin niteliğine göre içlerinde bilgi barındırabilirler. Parantez içine yazılan bu bilgilere parametre adı verilir. Fonksiyon tanımlanırken kullanılan parametre fonksiyon çağrılırken de gövdeye mutlaka yazılır.

```
def selamla(isim):
    print("Merhaba, benim adım %s." %isim)

selamla("Gökhan") # Çıktı: Merhaba, benim adım Gökhan.
```

Örnek: 3, 5 ve 7 sayılarından oluşan bir listenin elemanlarının çarpılması sonucu ortaya çıkan değeri ekrana yazdıran programı bir parametreli fonksiyon tanımlayarak hazırlayınız.

```
def carpimSonucu(liste):
    carpim = 1
    for i in liste:
        carpim *= i
    print("Sonuc: %s." %carpim)

x = [3, 5, 7]
carpimSonucu(x) # Çıktı: 105
```

Örnek 2: Klavyeden "Kaç sayı gireceksiniz?" şeklinde çıkan mesaja bir cevap vererek girilen sayıları bir listeye gönderiniz. Girilen sayıları tek ve çift olarak ayırarak;

a. girilen sayı adedini,
b. tek sayıların toplamını,
c. çift sayıların toplamını

ayrı ayrı ekranda gösteren programı hazırlayınız. *(Çıkmış Vize Sorusu 2015 veya 2016)*

```
liste = []
tekSayiToplami = 0
ciftSayiToplami = 0
girilecekSayiAdeti = int(input("Kaç sayı gireceksiniz: "))

for sayi in range(girilecekSayiAdeti):
  liste.append(int(input("{}. sayıyı girin: ".format(sayi + 1))))
  if liste[sayi] % 2 == 0:
    ciftSayiToplami += liste[sayi]
  else:
    tekSayiToplami += liste[sayi]


print("Girilen sayı adeti: {}".format(girilecekSayiAdeti))
print("Tek sayıların toplamı: {}".format(tekSayiToplami))
print("Çift sayıların toplamı: {}".format(ciftSayiToplami))
```

Örnek 3: Kullanıcı adı girilerek bir sisteme dahil olunmak istenmektedir. Kullanıcı adını doğru olarak girmek için kullanıcıya üç hak verilmiştir. Doğru cevap girildiğinde sisteme girişi sağlayan, her yanlış cevap girildiğinde ise kullanıcının kaç hakkı kaldığını belirterek tekrar kullanıcı adını girmesini isteyen ve kullanıcının hakkı kalmadığında sistemi kapatan programı sadece if yapısı kullanarak hazırlayınız.

```
cevap = "samsun"
kullaniciAdi = input("Kullanıcı adınızı giriniz: ")
if kullaniciAdi == cevap:
    print("Giriş başarılı!")
else:
    print("Giriş başarısız! Kalan hak: 2")
    kullaniciAdi = input("Kullanıcı adınızı giriniz: ")
    if kullaniciAdi == cevap:
        print("Giriş başarılı!")
    else:
        print("Giriş başarısız! Kalan hak: 1")
        kullaniciAdi = input("Kullanıcı adınızı giriniz: ")
        if kullaniciAdi == cevap:
            print("Giriş başarılı!")
        else:
            print("Giriş başarısız! Sistem kapatılıyor...")
```

```
hak = 3
cevap = "samsun"

while hak != 0:
    kullaniciAdi = input("Kullanıcı adınızı giriniz: ")
    if kullaniciAdi == cevap:
        print("Giriş başarılı!")
        break
    else:
        hak -= 1
        print("Giriş başarısız! Kalan hak: {}".format(hak))
if hak == 0:
    print("Sistem kapatılıyor...")
```

Örnek 4: Dört işlem yapabilen programı dört farklı fonksiyon tanımlayarak hazırlayınız.

```
def Topla(sayi1, sayi2):
    print("Sonuç:", (sayi1 + sayi2))

def Cikar(sayi1, sayi2):
    print("Sonuç:", (sayi1 - sayi2))

def Carp(sayi1, sayi2):
    print("Sonuç:", (sayi1 * sayi2))

def Bol(sayi1, sayi2):
    print("Sonuç:", (sayi1 / sayi2))

s1 = int(input("Birinci sayı: "))
s2 = int(input("İkinci sayı: "))

print("1: Toplama")
print("2: Çıkarma")
print("3: Çarpma")
print("4: Bölme")
islem = int(input("İşlem seçiniz: "))

if islem == 1:
    Topla(s1, s2)
elif islem == 2:
    Cikar(s1, s2)
elif islem == 3:
    Carp(s1, s2)
else:
    Bol(s1, s2)
```

Karakter Dizilerinin İçeriğinin Karşılaştırılması

Yapısal olarak elimizde bulunan karakter dizilerinin *(metinlerin)* içeriğini karşılaştırabiliriz. Amacımız bir metinde bulunup diğer metinde bulunmayan karakterleri ortaya çıkarmak ise; karakterlerin tek tek saydırılması için for döngüsünden, karşılaştırma işlemi için ise if deyiminden yararlarınız. Aradığımız değerler bir metinde olup diğerinde olmayan karakterler olduğu için if deyimine "not" sözcüğünü de ekleriz. Aşağıdaki örnekte ilk metinde olup ikinci metinde olmayan karakterlerin ekrana yazdırılması gösterilmiştir.

```
ilkMetin = "Hakan"
ikinciMetin = "Cem"
for i in ilkMetin:
    if not i in ikinciMetin:
        print(i)
```

Yukarıdaki ekran çıktısında tekrarlanan karakterler de gözükmektedir. Bu durumun önüne geçmek için "fark" isminde bir değişken tanımlayarak bu değişkeni başlangıç değerini "boş" olarak belirttik. Ardından for döngüsü içerisindeki i değişkeninin o anki değerinin hem ikinci metin hem de fark değişkeninin içerisinde karşılaştırma işlemine tuttuk. Eğer ki i değişkeni fark değişkeninin içerisinde bulunmuyorsa "fark += i" işlemini yaparak fark değişkenini s kadar arttırdık. Bu sayede tekrarlanan ifadelerin önüne geçtik. Son olarak da fark değişkenini ekrana yazdırdık.

```
ilkMetin = "Hakan"
ikinciMetin = "Cem"
fark = ""
for i in ilkMetin:
    if not i in ikinciMetin:
        if not i in fark:
            fark += i
print(fark)
```

Dosya İçeriğinin Karşılaştırılması

Karakter dizilerinin içeriğini karşılaştırabileceğimiz gibi dosyalar üzerinde de karşılaştırma işlemi uygulayabiliriz. Bunun için Python klasörü ile aynı yere farklı isimlerde iki dosya açalım. Ardından bu dosyaların karşılaştırma işlemine geçildiğinde ilk olarak dosyalar "open" komutu ile açılır. Daha sonra satır satır okuma yapmak için "readlines" metodu kullanılır. Bu komut sayesinde birbirinden farklı büyüklükteki dosyalar satır satır okunarak karşılaştırılabilir. Karşılaştırma işlemi için karakterlerde olduğu gibi for ve if yapıları birlikte kullanılır. Son olarak programın başında "open" komutu ile açılan dosyalar "close" komutu ile mutlaka kapatılır. Aşağıda "isimler1.txt" ve "isimler2.txt" uzantılı iki metin dosyasının içerikleri satır satır karşılaştırılmış ve "isimler2.txt" dosyasında olup "isimler1.txt" dosyasında olmayan ögeler ekrana yazdırılmıştır.

```
dosya1 = open("isimler1.txt")
dosya1_satirlar = dosya1.readlines()
dosya2 = open("isimler2.txt")
dosya2_satirlar = dosya2.readlines()
# Yukarıda isimler dosyalarını açıp satır satır okuttuk.
# Dosya açmak için değişken kullanmalıyız.

for i in dosya2_satirlar:
    if not i in dosya1_satirlar:
        print(i)

dosya1.close()
dosya2.close()
```

Bir Metindeki Karakterlerin Sayısını Bulmak

```
sayac = 0
metin = "çarşamba ticaret borsası meslek yüksekokulu"
harf = input("Sorgulamak istediğiniz karakteri giriniz: ")
for i in metin:
    if harf == i:
        sayac += 1
print("Toplam:", sayac)
```

Yukarıdaki örnekte klavyeden girilecek olan harf değişkeninin başlangıç değeri sıfır olarak alınmıştır. Ardından döngü değişkeni olan i değişkeninin o anki değişkenin klavyeden girilen harf değişkenine eşit olup olmadığına bakılır. Eşit olduğu durumlarda sayı değişkeni bir arttırılır. Ardından sayı değişkeni ekrana yazdırılır.

Bir Dosya İçerisindeki Karakterlerin Sayısını Bulma

Bir dosyada okunan herhangi bir metin üzerinde işlem yaparken o metnin yolu Python klasörü ile aynı olmalıdır. Programa başlarken üzerinde işlem yapılacak olan dosya "open" komutu ile açılır. Ardından hem dosyanın hem de karakter dizisinin üzerinde klavyeden girilen karakter tek tek karşılaştırılır. Daha sonra klavyeden girilen karakter döngü değişkenine eşitse sayı değeri 1 arttırılır. Son olarak sayı değeri ekrana yazdırılır.

```
dosya = open("dosya01.txt")
harf = input("Sorgulanacak olan karakteri giriniz: ")
sayac = 0
for satir in dosya:
    for karakter in satir:
        if harf == karakter:
            sayac += 1
print("Toplam:", sayac)
```

Ödev: Ödev isimli bir metin dosyası (.txt) oluşturularak içerisine "çarşamba meslek yüksekokulu" bilgisini giriniz. Klavyeden rastgele girilen herhangi bir karakterin dosyanın içerisinde bulunup bulunmadığını ekrana yazdıran, eğer karakter bulunduysa o karakterin kaçıncı sırada bulunduğunu ekrana yazdıran programı hazırlayınız.

```
dosya = open("Odev.txt") # Dosyamızı açtık.
harf = input("Sorgulanacak olan karakteri giriniz: ") # Aranan karakteri girdik.

sayac = 0 # Karakterin hangi satır numarasında olduğunu görmek için gerekli.
dizi = [] # Bulunan karakterin numaralarını bu diziye atacağız.
bayrak = False # Karakterin bulunup bulunmadığını sorgulamak için gerekli.
for satir in dosya: # Dosyayı satır satır for döngüsünde döndürüyoruz.
    for karakter in satir: # Satırı karakter karakter for döngüsünde döndürüyoruz.
        sayac += 1 # Karakter sırasını her seferinde 1 artırıyoruz.
        if karakter.lower() == harf.lower(): # Gereken sorgu ifadesi sağlanırsa…
            dizi.append(sayac) # Karakterin bulunduğu numarayı diziye ekledik.
            bayrak = True # Karakterin bulunduğunu belirttik.

print("Aranan karakter: {}".format(harf)) # Aranan karakteri yazdırdık.
if bayrak == True: # Karakter bulunduysa…
    print("Bulunan karakter sayıları: {}".format(dizi))
    # Bulunan karakterin olduğu numaraları diziden gösteriyoruz.
else:
    print("Aranan karakter bulunamadı!")
    # Aranan karakter bulunmadıysa bayrak false olarak kaldığı için bunu yazdırdık.
```

Bugün Python üzerinden masaüstü programlama ile ilgili toplamda altı uygulama yapacağız.

1) Python'da Menü Hazırlama

Python'da menü hazırlamak için "tkinter" modülü kullanılmıştır. Bu menüye bağlı olan alt menüler (subMenu) "cascade" şeklinde bağlanarak, ilgili bölümler de "separator" ayracı kullanılarak gruplandırılmıştır. Son olarak da pencere nesnesi "mainloop" komutu ile sonlandırılmıştır.

```
from tkinter import *

pencere = Tk()

def dosyaKaydet():
    print("Dosya kaydedildi!")
def dosyaAc():
    print("Dosya açıldı!")
def geriAl():
    print("Geri alma işlemi uygulandı!")

menu = Menu(pencere)
pencere.config(menu = menu)

subMenu = Menu(menu)

menu.add_cascade(label = "Dosya", menu = subMenu)
subMenu.add_command(label = "Kaydet", command = dosyaKaydet)
subMenu.add_command(label = "Aç", command = dosyaAc)
subMenu.add_separator()
subMenu.add_command(label = "Çıkış", command = pencere.destroy)

editMenu = Menu(menu)
menu.add_cascade(label = "Düzenle", menu = editMenu)
editMenu.add_command(label = "Geri Al", command = geriAl)

pencere.mainloop()
```

2) Sayfa Üzerine Araç Çubuğu Eklemek

Sayfa üzerine araç çubuğu ekleyerek araç çubuğu üzerine buton ilave edip butona görsel özellikler kazandırma kodları aşağıdaki gibidir.

```
# Fonksiyon kısmına bu kodlar eklendi.

def ekle():
    print("Resim eklendi!")
def yukle():
    print("PDF yüklendi!")
```

```
# "Geri Al" komutundan sonra, mainloop komutundan önce bu kodlar eklendi.

aracCubugu = Frame(pencere, bg = "red")
resimEkle = Button(aracCubugu, text = "Resim Ekle", command = ekle)
resimEkle.pack(side = LEFT, padx = 2, pady = 2)
pdf = Button(aracCubugu, text = "PDF Yükle", command = yukle)
pdf.pack(side = LEFT, padx = 2, pady = 2)
aracCubugu.pack(side = TOP, fill = X)
```

3) Sayfa Üzerine Durum Çubuğu Eklemek

```
# İkinci kodun sonu ile mainloop komutunun öncesine eklenen kodlar aşağıdadır.
durumCubugu = Label(pencere, text = "Sayfa yükleniyor...", bd = 2, relief = SUNKEN,
anchor = W)
durumCubugu.pack(side = BOTTOM, fill = X)
```

4) Butona Basıldığında Sayfa Üzerine Bilgilendirme Amaçlı Mesaj Kutusu Eklemek

```
from tkinter import *
from tkinter import messagebox # messagebox kullanmak için bu gerekli.

pencere = Tk() # Pencere oluşturduk
def bilgiMesaji(): # Bir fonksiyon oluşturduk.
       messagebox.showinfo("Çarşamba MYO", "Python uygulamaları")

buton = Button(pencere, text = "Giriş", command = bilgiMesaji) # Buton oluşturduk.
buton.pack(side = TOP) # Butonun yerleşeceği yeri belirledik.

pencere.mainloop()
```

5) Ekrana Çıkan MessageBox'da Kişiye Soru Soran ve Kullanıcının Vereceği Cevaba Göre İşleme Devam Eden Program

```
from tkinter import *
from tkinter import messagebox # messagebox kullanmak için bu gerekli.

pencere = Tk() # Pencere oluşturduk

cevap = messagebox.askquestion("Soru", "Devam etmek istiyor musunuz?") # Bir soru
soruyoruz.

if cevap == "yes": # Cevap evet ise...
    print("Tebrikler!")
else: # Cevap hayır ise...
    print("Üzgünüz!")

pencere.mainloop()
```

6) Sayfa Üzerinde Fotoğraf Göstermek

```
from tkinter import *
pencere = Tk()

fotograf = PhotoImage(file = "06.gif") # Bir fotoğraf yolu belirledik.
etiket = Label(pencere, image = fotograf)
etiket.pack()

pencere.mainloop()
```

Soru: Bir veritabanı üzerindeki şehir isimlerinin bulunduğu alandaki bilgileri rakam kullanarak ve şehir isimlerini de ekrana alt alta yazarak çalıştıran programı Python dilinde hazırlayınız.

```
sehir = ["İstanbul", "Ankara", "İzmir", "Samsun"]

for x in range(len(sehir)):
    print("{} - {}".format(x + 1, sehir[x]))

# İkinci yapma yolu da bu şekilde.
sayi = 0
while sayi < len(sehir):
    print("{} - {}".format(sayi + 1, sehir[sayi]))
    sayi += 1
```

Ödev: Aşağıdaki matematiksel denklemin sonucunu bulan programı Python dilinde hazırlayınız.

$$((1 * 2) / (3 * 4)) - ((5 * 6) / (7 * 8)) + ((9 * 10) / (11 * 12)) - ((13 * 14) / (15 * 16))$$

```
sayilar = []
tek = 1
cift = 2
for x in range(4):
    islem1 = tek * cift
    tek += 2
    cift += 2
    islem2 = tek * cift
    islem3 = islem1 / islem2
    if x % 2 == 0:
        sayilar.append(islem3)
    else:
        sayilar.append(-islem3)

    tek += 2
    cift += 2

# Bu kısım yukarıdaki işlemin sonucunu göstermek için.
toplam = 0
for y in range(len(sayilar)):
    toplam += sayilar[y]

print(toplam) # Çıktı: -0.4455627705627706
```

Oyun Tasarlamak

Bu derste Python masaüstü programlama üzerinden 3x3 boyutunda SOS oyununu geliştireceğiz. Bunun için öncelikle oyunun basit bir tasarımını yapmak gerek.

- 9 buton gerek.
- Bir oyuncunun oyunu kazanabilmesi için 8 ihtimal var. (3 dikey, 3 yatay, 2 çapraz)
- Oyuna başlarken bir oyuncu ataması yapmalıyız. Birinci oyuncu X, ikinci oyuncu Y olsun.
- Oyunun kodları bir sonraki sayfada bulunmaktadır.

```python
from tkinter import *
from tkinter import messagebox

root = Tk() # Pencere oluşturduk.
root.title = "SOS Oyunu" # Pencerenin başlığını verdik.

b_click = True

def SOS(buttons): # SOS adında bir fonksiyon tanımladık.
    global b_click

    if ((button1["text"] == "X" and button2["text"] == "X" and button3["text"] == "X") or
            (button4["text"] == "X" and button5["text"] == "X" and button6["text"] == "X") or
            (button7["text"] == "X" and button8["text"] == "X" and button9["text"] == "X") or
            (button1["text"] == "X" and button4["text"] == "X" and button7["text"] == "X") or
            (button2["text"] == "X" and button5["text"] == "X" and button8["text"] == "X") or
            (button3["text"] == "X" and button6["text"] == "X" and button9["text"] == "X") or
            (button1["text"] == "X" and button5["text"] == "X" and button9["text"] == "X") or
            (button3["text"] == "X" and button5["text"] == "X" and button7["text"] == "X")):
        messagebox.showinfo("Kazandınız!", "X oyuncusu oyunu kazandı!")
        root.destroy()

    elif ((button1["text"] == "Y" and button2["text"] == "Y" and button3["text"] == "Y") or
            (button4["text"] == "Y" and button5["text"] == "Y" and button6["text"] == "Y") or
            (button7["text"] == "Y" and button8["text"] == "Y" and button9["text"] == "Y") or
            (button1["text"] == "Y" and button4["text"] == "Y" and button7["text"] == "Y") or
            (button2["text"] == "Y" and button5["text"] == "Y" and button8["text"] == "Y") or
            (button3["text"] == "Y" and button6["text"] == "Y" and button9["text"] == "Y") or
            (button1["text"] == "Y" and button5["text"] == "Y" and button9["text"] == "Y") or
            (button3["text"] == "Y" and button5["text"] == "Y" and button7["text"] == "Y")):
        messagebox.showinfo("Kazandınız!", "Y oyuncusu oyunu kazandı!")
        root.destroy()
    if buttons["text"] == " " and b_click == True:
        buttons["text"] = "X" # Oyuna başlayacak olan oyuncuyu belirledik.
        b_click = False

    elif buttons["text"] == " " and b_click == False:
        buttons["text"] = "Y"
        b_click = True

buttons = StringVar()
button1 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button1))
button1.grid(row = 1, column = 1, sticky = S+N+E+W)

button2 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button2))
button2.grid(row = 1, column = 2, sticky = S+N+E+W)

button3 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button3))
button3.grid(row = 1, column = 3, sticky = S+N+E+W)

button4 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button4))
button4.grid(row = 2, column = 1, sticky = S+N+E+W)

button5 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button5))
button5.grid(row = 2, column = 2, sticky = S+N+E+W)

button6 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button6))
button6.grid(row = 2, column = 3, sticky = S+N+E+W)

button7 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button7))
button7.grid(row = 3, column = 1, sticky = S+N+E+W)

button8 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button8))
button8.grid(row = 3, column = 2, sticky = S+N+E+W)

button9 = Button(root, text = " ", font = ('Arial 30 bold'), height = 4, width = 8, command = lambda:SOS(button9))
button9.grid(row = 3, column = 3, sticky = S+N+E+W)

cevap = messagebox.askquestion("Soru", "Oyuna birinci oyuncu mu başlasın?")
if cevap == "yes":
    b_click = True
else:
    b_click = False

root.mainloop()
```

Python ile Analiz Programı Gerçekleştirme

Herhangi bir kişinin girdiği cümleye göre kişinin o anki ruh halini analiz etmek için yapay zekâ tabanlı bir tahmin programı oluşturulmalıdır. Bir fonksiyon tanımlanarak veri setini olumlu ve olumsuz olmak üzere iki farklı liste üzerinden kurarız. Buradaki amaç kişinin klavyeden girdiği cümlenin içerisindeki kelimeler ile veri setindeki kelimeleri karşılaştırmaktır. Klavyeden girilen cümlenin içerisindeki kelimeleri split() metodu ile ayırabiliriz. Hem olumlu hem de olumsuz veri setinin içerisindeki kelimeleri ayrı ayrı çekerek len() komutu yardımıyla sayılarını belirleriz. Daha sonra if komutu ile karşılaştırma işlemini gerçekleştirip programı başında oluşturduğumuz fonksiyonu kapatarak işlemi sonlandırırız. Bu programda "set" komutu verilerin çekileceği liste veya listelere uygulanmaktadır. Bu "set" komutu ile kurulan veriler hangi veri setinden çekileceği belirtildikten sonra mutlaka list komutu ile veri tipinin cinsi *(sözlük, liste, demet)* belirtilmelidir.

```
def analiz():
    olumluKelimeler = ["süper", "harika", "güzel", "iyi", "olumlu"]
    olumsuzKelimeler = ["berbat", "iğrenç", "kötü", "olumsuz", "çöp", "sıkıcı"]
    # Sözlükteki kelime sayısı ne kadar artarsa o kadar doğru ihtimaller artar.

    cumle = input("Bir cümle giriniz: ").lower()
    # Cümleyi kullanıcı girsin.
    # Giriş yaptığımız cümle: "rezil berbat güzel iyi olumlu bir gün geçirdim."
    kelimeListesi = cumle.split() # Cümleyi split ile kelimeListesine atıyoruz.

    olumluKesim = list(set(olumluKelimeler) & set(kelimeListesi))
    olumsuzKesim = list(set(olumsuzKelimeler) & set(kelimeListesi))

    # Sayaç kısmımız.
    olumluKesim = len(olumluKesim)
    olumsuzKesim = len(olumsuzKesim)

    if olumluKesim > olumsuzKesim:
        print("Bugün seni iyi gördüm!")
    elif olumsuzKesim > olumluKesim:
        print("Bugün iyi görünmüyorsun.")
    else:
        print("Bir fikre varamadım.")

analiz() # Fonksiyonu çağırdık.
```

Python ile Veri Şifreleme

Bir dosya içerisindeki metni şifrelemek veya şifreyi çözmek için "sys" modülünden faydalanılır.

Şifreleme işlemi için şifreyi klavyeden isteyerek şifrelenecek olan metin dosyasının tam yolu yazılır. Ardından bu dosya açılıp okutularak içerisindeki veriler ve girilen şifre değeri bir liste halinde sıralanarak for döngüsü yardımıyla tek tek saydırılır. Son olarak metin içerisindeki her bir karaktere 127 değeri eklenerek karakterin ASCII kod karşılığındaki 127'ye bölümünden kalan değer tekrar karaktere çevrilerek hafızada tutulur.

Şifreyi çözmek için ise hafızada tutulan "sifreliDosya.txt" kodları açılıp okutularak tek tek 127 değeri ile toplanıp tekrardan karaktere çevrilir ve hafızada tutulur ve "cozulmusDosya.txt" adıyla saklanır.

```python
import sys # Sistem sınıfını import ettik.

def veriyiSifrele():
    giris = input("Şifreleme işlemi için en az 5 karakter, en fazla 15 karakter giriniz: ")
    if len(list(giris)) >= 5 & len(list(giris)) <= 15:
        yol = input("Şifrelenecek yolu giriniz: ")
        d = open(yol)
        veri = d.read()
        veri = list(veri)
        giris = list(giris)

        for i in giris:
            araSonuc = "" # Şifreleme işlemi için bir ara sonuç oluşturduk.
            for l in veri:
                sonuc = ord(l) + ord(i)
                if sonuc >= 127: #ASCII karakteri olayı için 127 yazdık.
                    sonuc %= 127
                araSonuc += chr(sonuc)

        d = open("sifreliDosya.txt", "w")
        d.write(araSonuc)
        d.close()
    else:
        print("Lütfen kurallara uygun parola giriniz.")

def veriyiCoz():
    giris = input("Şifreyi çözmek için anahtar sözcüğü giriniz: ")
    if len(list(giris)) >= 5 & len(list(giris)) <= 15:
        d = open("sifreliDosya.txt")
        veri = d.read()
        veri = list(veri)
        giris = list(giris)
        gelen = " "
        d.close()

        for i in giris:
            araSonuc = "" # Şifreleme işlemi için bir ara sonuç oluşturduk.
            for l in veri:
                sonuc = ord(l) - ord(i)
                if sonuc <= 0:
                    sonuc += 127 #ASCII karakter olayı için 127 yazdık.
                araSonuc += chr(sonuc)

        d = open("cozulmusDosya.txt", "w")
        d.write(araSonuc)
        d.close()
    else:
        print("Lütfen kurallara uygun parola giriniz!")


while True:
    giris = input("Hangi işlemi yapmak istiyorsun?\n1) Şifrele\n2) Şifreyi çöz\n3)
Çıkış\nİşlem: ")
    if giris == "1":
        veriyiSifrele()
    elif giris == "2":
        veriyiCoz()
    else:
        sys.exit(0)
```

Sınıflar ve Miras Alma *(Inheritance)*

Nesne tabanlı programlamada en önemli kavram sınıflardır. Sınıflar yapısal olarak fonksiyonlara benzetilebilir. Sınıfların içerisinde fonksiyonları, veri tiplerini, değişkenleri, metotları gruplandırabiliriz. Sınıfları tanımlamak için öncelikle "class" sözcüğünü kullanıp sonrasında da sınıf adını belirtiriz.

```
class OrnekSinif:
```

Sınıfı kullanırken veya sınıf üzerinde işlem yaparken örnekleme adı verilen işleri mutlaka kullanırız. Oluşturduğumuz sınıfa isim verme işlemine "örnekleme" denir.

```
ornek01 = OrnekSinif()
```

Yani OrnekSinif adlı sınıfa isim verme işlemine örnekleme denirken bu işlem sonucunda ortaya çıkan değişkene de örnek değişken adını veriyoruz.

Bir sınıfı örneklemezsek o örneklenmeyen sınıf program tarafından otomatik olarak çöp toplama (Garbage Collection) adı verilen bir sürece tabii tutulacaktır.

Python dilinde her bir nesnenin niteliğine erişmek için sınıf adından sonra *(örnekleme yapıldıysa örnekten sonra)* mutlaka araya nokta konulur.

```
class Toplama:
    a = 15
    b = 20
    c = a + b

sonuc = Toplama()

print(sonuc.a) # 15
print(sonuc.b) # 20
print(sonuc.c) # 35
```

init fonksiyonu içerisinde tanımlanan bütün değişkenlerin ve niteliklerin ilk çalışma esnasında ekrana yazdırıldığı fonksiyondur. init fonksiyonunun varsayılan değerleri belirleme yani inşa etme özelliği yoktur.

self fonksiyonu ise programın farklı noktalarında kullanılacak olan değişkenlere veya fonksiyonlara erişmek için kullanılan bir ön ektir. Bütün bu işlemler aslında hep miras alma *(inheritance)* için kullanılır. Miras alma kavramı aynı sınıf içerisinde veya farklı sınıflar içerisinde bulunan fonksiyon ya da değişkenlerin değerlerini belirlemek için kullanılan bir özelliktir. Miras alma işleminde bir ana sınıftan, o ana sınıfa bağlı bütün yavru sınıftaki özelliklere veri akışı yapılabileceği gibi aynı sınıf içerisindeki değişken ve nesnelere de veri akışı uygulanabilir.

Bir sonraki sayfada dört örnek yapacağız.

Örnek 1: Örnekleme yaparak form ekranına Frame ve Entry eklemek.

```
from tkinter import *

pencere = Tk() # Yeni bir pencere oluşturduk.
pencere.title("Çarşamba MYO") # Pencereye başlık verdik.
pencere.geometry("400x300") # Pencereye genişlik ve yükseklik verdik.

uygulama = Frame(pencere)
uygulama.grid()

L1 = Label(uygulama, text = "Adınızı girin: ")
L1.grid(padx = 110, pady = 10)
E1 = Entry(uygulama, bd = 2)
E1.grid(padx = 110, pady = 3)

pencere.mainloop()
```

Örnek 2: Örnekleme yaparak form ekranına Listbox da eklemek.

```
from tkinter import *

pencere = Tk() # Yeni bir pencere oluşturduk.
pencere.title("Çarşamba MYO") # Pencereye başlık verdik.
pencere.geometry("400x300") # Pencereye genişlik ve yükseklik verdik.

uygulama = Frame(pencere)

uygulama.grid()

Lbl = Listbox(uygulama)
Lbl.insert(1, "Python")
Lbl.insert(2, "C#")
Lbl.insert(3, "Java")

Lbl.grid(padx = 110, pady = 10)

pencere.mainloop()
```

Örnek 3: Bir oyun programındaki herhangi bir oyuncu için kalan can sayısını ve oyuncunun hayatta kalıp kalmadığını ekrana yazdıran programı Python dilinde sınıf ve miras alma kullanarak hazırlayınız.

```
class Oyuncu:
    KalanCan = 3

    def Saldir(self):
        print("Hücum!")
        self.KalanCan -= 1

    def HayattaMi(self):
        if (self.KalanCan <= 0):
            print("Öldü!")
        else:
            print("Kalan can: {}".format(self.KalanCan))

oyuncu01 = Oyuncu()
oyuncu02 = Oyuncu()

oyuncu01.Saldir()
oyuncu01.HayattaMi()

oyuncu02.Saldir()
oyuncu02.Saldir()
oyuncu02.Saldir()
oyuncu02.HayattaMi()
```

Örnek 4: Sayfaya eklenen herhangi bir nesne üzerinde başlangıç değeri ve daha sonra eklenecek olan nesneleri için aynı özelliğin miras yoluyla eklenmesi.

```
from tkinter import *

class Arayuz:
    sayac = 0

    def Yaz(self):
        self.sayac += 1
        print("{}. iyi günler!".format(self.sayac))

    def __init__(self):
        pencere = Tk()
        pencere.geometry("400x300")
        Dugme01 = Button(pencere, text = "Gönder", command = self.Yaz)
        Dugme01.pack()

        pencere.mainloop()

uygulama = Arayuz()
```

Örnek 1: Bir banka hesabında; para çekmek, para yatırmak, havale ve kalan parayı göstermek gibi işlemleri bütün kullanıcılar için gerçekleştirebilecek programı sınıf ve miras alma yoluyla hazırlayınız.

```
class BankaHesabi:
    def __init__(self, ilkMiktar = 0):
        self.kalanPara = ilkMiktar
        print("Yeni hesap {} ₺ ile açılmıştır.".format(ilkMiktar))

    def KalanPara(self):
        print("Kalan para: {}".format(self.kalanPara))

    def ParaYatir(self, yatirilacakMiktar):
        self.kalanPara = self.kalanPara + yatirilacakMiktar

    def ParaCek(self, cekilecekMiktar):
        self.kalanPara = self.kalanPara - cekilecekMiktar

    def Havale(self, miktar, havaleEdilecekKisi):
        self.ParaCek(miktar)
        havaleEdilecekKisi.ParaYatir(miktar)

Ali = BankaHesabi(50000) # Ali için bir banka hesabı oluşturduk.
Ayse = BankaHesabi(20000) # Ayşe için de bir banka hesabı oluşturduk.

Ali.ParaYatir(15000) # Ali'nin hesabına 15.000 ₺ yatırdık.
Ali.KalanPara() # Ali'nin hesabına tekrar baktık.

Ali.Havale(20000, Ayse) # Burada Ali, Ayşe'ye 20000 ₺ gönderdi.
Ayse.KalanPara() # Ayşe'nin yeni parasını görüntülemek için kullandık.
Ali.KalanPara() # Ali'nin yeni parasını görüntülemek için kullandık.
```

Örnek 2: Aşağıda özellikleri verilen restoran programını klavyeden girilecek kişinin adına göre tüm kişiler ve seçenekler için çalıştırabilen programı hazırlayıp kişinin ödeyeceği hesabı ekrana çıkartan programı sınıf ve miras alma özelliklerini kullanarak Python dilinde hazırlayınız.

Yiyecekler		İçecekler		Tatlılar	
Et	25 ₺	**Su**	1 ₺	**Baklava**	8 ₺
Tavuk	20 ₺	**Ayran**	1 ₺	**Şöbiyet**	8 ₺
Balık	25 ₺	**Kola**	3 ₺	**Künefe**	10 ₺
Köfte	15 ₺	**Fanta**	3 ₺		

```
class Restorant:
	yiyecek = {"Et": 25, "Tavuk": 20, "Balık": 25, "Köfte": 15}
	icecek = {"Su": 1, "Ayran": 1, "Kola": 3, "Fanta": 3}
	tatlilar = {"Baklava": 8, "Şöbiyet": 8, "Künefe": 10}

	# "print()" komutları boşluk basmak için kullanılmıştır.
	print()
	print("Yiyecekler: {}".format(yiyecek))
	print("İçecekler: {}".format(icecek))
	print("Tatlılar: {}".format(tatlilar))
	print()

	def __init__(self, ucret = 0):
		self.hesap = ucret
		print("Hesap: {} ₺".format(self.hesap))
		print()

	def YiyecekSec(self, secilenYiyecek):
		self.hesap += self.yiyecek[secilenYiyecek]

	def IcecekSec(self, secilenIcecek):
		self.hesap += self.icecek[secilenIcecek]

	def TatliSec(self, secilenTatli):
		self.hesap += self.tatlilar[secilenTatli]

	def Hesap(self):
		print("Ödenecek miktar: {} ₺".format(self.hesap))


Ali = Restorant()

Ali.YiyecekSec(input("Yiyecek: "))
Ali.IcecekSec(input("İçecek: "))
Ali.TatliSec(input("Tatlı: "))

Ali.Hesap()
```

Kaynaklar:

1- Her yönüyle Python, Fırat Özgül, KODLAB