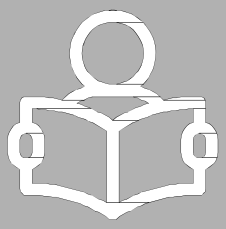
PYTHON

İleri Seviye Fonksiyonlar

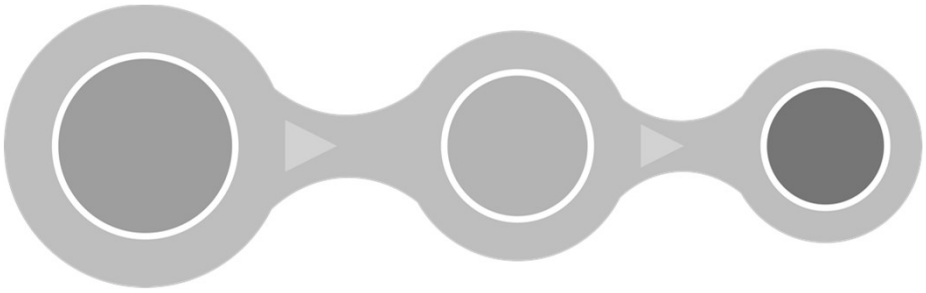
Kapsam



- List Comprehension
- Özyinelemeli Fonksiyonlar
- Lambda Fonksiyonları
- Map Fonksiyonu
- Reduce Fonksiyonu
- Filter Fonksiyonu

İleri Seviye Fonksiyonlar

Python'un en belirgin özelliklerinden biri de tek bir kod satırında güçlü ve işlevsel özellikler oluşturabilmemize olanak vermesidir.



Bu özelliklere sahip olan ileri seviye fonksiyonlardan List Comprehension, lambda, map, filter ve reduce fonksiyonları işlenecektir.

Bu fonksiyonlar python'ca (phytonic) yapılma yöntemlerini ifade ederler.

Birkaç satırda yapılacak işlemi tek satırda yapabilme kolaylığı ve zevkini sunarlar.

List Comprehension

List Comprehension, bir çırpıda liste oluşturma manasına gelmektedir.

Kod

```
liste=[1,2,3]
liste=[]
liste=list()
```

Yukarıdaki yöntemlerin üçü de kullanılarak liste oluşturulabilir.
Bunun yanında liste oluştururken döngülerden de faydalanabiliriz. Sayı dizilerinden listeler oluşturmak isteyeceğimiz gibi, bir listeden farklı bir liste türetmek için de Python'da for döngüsünden faydalanabiliriz. Şöyle ki;

Kod

```
liste1=[1,2,3,4,5]
liste2=[]
for i in liste1:
    liste2.append(i)
print(liste2)
```

List Comprehension

Aynı işlem List Comprehension metodu sayesinde daha az kodla yapabilir.

Kod

```
liste1=[1,2,3,4,5]
liste2=[i for i in liste1]
print(liste2)
```

Bu işlem ile mevcut liste üzerinde işlem de yapılabilir. Şöyle ki; örneğin liste elemanlarının karesini alan list comprehension uygulaması yapalım.

Kod

```
liste1=[3,4,5,6,7]
liste2=[i**2 for i in liste1]
print(liste2)
```

List Comprehension

list comprehension metodu ile belirli bir aralık kadar da liste oluşturulabilir.

Kod

```
liste1=[i for i in range(0,10)]
```

Aynı işlemler string'ler üzerinde de uygulanabilir.

Kod

```
yazi="python"
liste=[i*2 for i in yazi]
print(liste)
```

List Comprehension

list comprehension metodu ile iç içe listeleri birleştirerek tek bir liste üzerinde birleştirme yapabiliriz.

Kod

```
liste1=[[1,2,3],[4,5,6],[7,8,9]]
liste2=[j for i in liste1 for j in i]
print(liste2)
```

list comprehension metodu ile çift sayıları bulan uygulama

Kod

```
liste=[i for i in range(1,20) if i%2==0]
print(liste)
```

Özyinelemeli Fonksiyonlar

Özyinelemeli (recursive) fonksiyonlar kendilerini tekrar kullanan /çağıran fonksiyonlardır. Bu fonksiyonlar, zor ve uzun problemleri daha kolay parçalara ayırarak adım adım çözme mantığına dayanırlar. Bilgisayar bilimlerinde böl ve yönet (Divide and Conquere) metodunun uygulamalarından biridir.
Problem çözmede farklı bir yaklaşım olarak ta görülebilir.
Örneğin faktöriyel hesaplama işlemi nasıl yapılır? Akla ilk gelen yöntem birden sayıya kadar sayıların çarpımını kullanmak.

Kod

```
faktoriyel=1
sayı=int(input("faktöriyelini hesaplamak istediğiniz sayıyı giriniz"))
if sayı>=0:
    for i in range(1,sayı+1):
    faktoriyel*=i
print (faktoriyel)
```

Özyinelemeli Fonksiyonlar

Faktöriyel hesaplamayı bir fonksiyon olarak tanımlamak istersek:

Kod

```
def faktoriyel_hesapla(sayı):
    faktoriyel=1
    if sayı>=0:
        for i in range(1,sayı+1):
            faktoriyel*=i
    return faktoriyel

sayı=int(input("faktoriyelini hesaplamak istediğiniz sayıyı giriniz"))
print (faktoriyel_hesapla(sayı))
```

Özyinelemeli Fonksiyonlar

Özyineleme kullanmaktaki amaç problemi daha küçük parçalara ayırarak çözmektir.
n!'in aslında n! = n*(n-1)! olarak görülürse ve en temel parçaya kadar bu yaklaşım benimsenir ise problem çözülmüş olur.
Örnekte en küçük birim 1! (bir faktöriyel) olacağından tüm işlem bir faktöriyele ulaşana kadar her sayının bir alt sayı faktöriyeli ile çarpılması olacaktır.

Kod

```
def faktoriyel_hesapla(sayı):
    faktoriyel=1
    if sayı==1:
        return 1
    else:
        return sayı*faktoriyel_hesapla(sayı-1)

sayı=int(input("faktoriyelini hesaplamak istediğiniz sayıyı giriniz: "))
print (faktoriyel_hesapla(sayı))
```

Özyinelemeli Fonksiyonlar

Özyinelemeye başka bir örnek ise fibonacci terim sayısının hesaplamasıdır.

Kod

```
def fibonacci(n):
    if n <= 2:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

sayı=int(input("istediğiniz fibonacci terim sayısını giriniz: "))
print(fibonacci(sayı))
```

Örnek 1

Birden girilen sayıya kadar olan sayıların toplamını hesaplayan programın özyineleme kullanılarak nasıl yapılabileceğini inceleyelim.

Kod

```
#for döngüsü ile toplam hesaplama
sayı=int(input("sayıyı giriniz: "))
toplam=0
for i in range(1,sayı+1):
    toplam+=i
print(toplam)

#Özyineleme yöntemi ile toplam hesaplama
def topla(sayı):
    if sayı==1:
        return 1
    else:
        return sayı+topla(sayı-1)
sayı=int(input("sayıyı giriniz: "))
print(topla(sayı))
```

Lambda Fonksiyonları

Lambda fonksiyonları küçük anonim fonksiyonlar olarak isimlendirilirler.
Lambda fonksiyonlarını işlevsel tek satırlık fonksiyonlar olarak düşünebilirsiniz.
Normal bir fonksiyon tanımlarken def anahtar kelimesi kullanılarak fonksiyon tanımlanır ve return anahtar kelimesi ile sonuç döndürülür.
Lambda yöntemi ile bu ikisinin de kullanılma ihtiyacı ortadan kalkar.
Böylece lambda kullanarak programlamada işimizi görecek pratik fonksiyonlar kullanabiliriz.
Argümanlar giriş, ifadeler çıkış değerleridir.
Kullanım şekli:
lambda argüman: ifadeler

Kod

```
x=lambda a: a+15
print(x(5))
```

Lambda Fonksiyonları

Lambda fonksiyonu ile hipotenüs hesaplama

Kod

```
import math
hipotenus=lambda x,y: math.sqrt(x**2+y**2)
print(hipotenus(3,4))
```

Lambdanın direkt kullanım şekli;

Kod

```
print((lambda x,y: 3*x+5*y)(2,5))
```

Map Fonksiyonu

map fonksiyonu bir işlemin veya fonksiyonun birden çok veriye tek satırda uygulanmasını sağlayan pratik fonksiyonlardan biridir.

map fonksiyonu for döngüsü kullanarak yapılabilecek işlemlerin tek satırda çözülmesine olanak tanır.

map fonksiyonun kullanımı şu şekildedir:

map(fonksiyon, iterasyon yapılabilecek veri tipi (liste, demet vb.))

Map Fonksiyonu

Davetliler listesi ile ilgili standart bir davet mesajı uygulaması üzerinden map fonksiyonunu anlamaya çalışalım. Sabit metne listedeki isimleri ekleyerek kişiye özgü davetiye metni oluşturma uygulaması yapılacaktır.

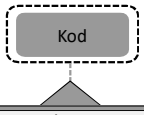
```
def davetiye_metni(isim):
    gonderilecek_metin ="Sayın "+isim+"\n"+" Bu mutlu günümüzde sizleri de aramızda
görmekten mutluluk duyarız."
    return gonderilecek_metin
isimler=["Ali","Ayşe","Murat","Elif"]
for birey in isimler:
    print(davetiye_metni(birey))
```

map fonksiyonu ile kişiye özel davetiye metni oluşturma

```
def davetiye_metni(isim):
    gonderilecek_metin ="Sayın "+isim+"\n"+" Bu mutlu günümüzde sizleri de aramızda
görmekten mutluluk duyarız. \n"
    return gonderilecek_metin
isimler=["Ali","Ayşe","Murat","Elif"]
davetiye=map(davetiye_metni,isimler)
print (davetiye)#davetiye değişkeninin bellekteki konumunu yazdırır
print (*davetiye)#davetiye değişkeninin değerini yazdırır
```

Map Fonksiyonu

map ve lambda fonksiyonlarının birlikte kullanımı




```
isimler=["Ali","Ayşe","Murat","Elif"]
yeni_liste =map(lambda x:"Sayın "+x+"\n Bu mutlu günümüzde sizleri
de aramızda görmekten mutluluk duyarız.\n", isimler)
print(*yeni_liste)
```

Reduce Fonksiyonu

reduce( ) fonksiyonu değer olarak aldığı fonksiyonu soldan başlayarak listenin ilk 2 elemanına uygular.
Daha sonra çıkan sonucu listenin 3. elemanına uygular. Bu şekilde devam ederek liste bitince bir tane değer döner.
Reduce fonksiyonu functools modülünde olduğundan bu modülü dahil ediyoruz.

Kod

```
from functools import reduce
print(reduce(lambda x,y : x + y , [1,2,3,4]))
```

Örnek incelediğinde lambda fonksiyonu ile 2 sayının toplandığı ardından üretilen sonuç ile listede bir sonraki sayının aynı toplama işlemine dahil edilerek bir sonuç üretildiği görülmektedir.

Reduce Fonksiyonu

reduce ile faktöriyel hesaplama

Kod

```
from functools import reduce
sayı=int(input("faktöriyelini hesaplamak istediğiniz sayıyı giriniz"))
liste=[i for i in range(1,sayı+1)]
print(reduce(lambda x,y :x*y, liste))
```

Filter Fonksiyonu

filter( ) fonksiyonu liste veri tipindeki yapılar için kullanılır.
Kullanım şekli:
filter(fonksiyon1, liste1)
liste1 liste tipinde değerler, fonksiyon1 önceden tanımlanmış fonksiyonu ifade etmektedir. İçinde kullanılan fonksiyonun döndürdüğü değerin doğru (True) olduğu durumlara göre liste veri tipinde çıktı üretir. Diğer bir değişle verilen liste üzerinde fonksiyon işlemlerinin çıktısı kullanılarak filtreleme işlemi yapılır.

Kod

```
def buyuk_harf(isim):
    return isim.isupper( )
liste=["Ali","ali","Selim","SELİM","selim"]
sonuc=filter(buyuk_harf,liste)
print(*sonuc)
```

Filter Fonksiyonu

filter ve reduce fonksiyonlarının birlikte kullanılarak verilen sayılardan sadece üçe tam bölünebilen sayıların ekrana yazdırıldığı uygulama

Kod

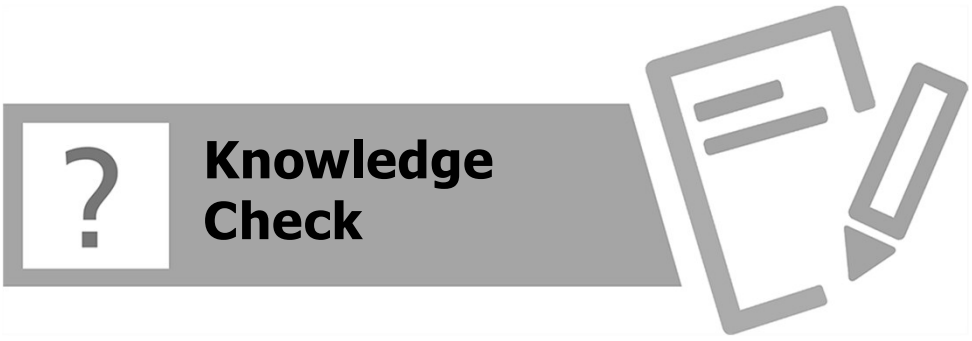
```
sayılar = [10, 15, 4, 29, 402, 249, 210, 55, 40, ]
sonuc = filter(lambda x: (x % 3 == 0), sayılar)
print(*sonuc)
```

Filter Fonksiyonu

Bir bankaya ait müşteri ve mevduat listesinden bakiyeleri 150'den büyük olanların seçilip ekrana yazdırılması

Kod

```
#müşteriler listesi müşteri no ile bakiye miktarı
musteriler= [[1, 12], [2, 600], [ 3, 500], [4,150]]
sonuc = filter(lambda x: (x[1] > 149),musteriler )
print(*sonuc)
```



1	**5'ten başlayıp 20'ye kadar olan çift sayılardan oluşan bir listeyi list comprehension kullanarak yapınız.**

1	**5'ten başlayıp 20'ye kadar olan çift sayılardan oluşan bir listeyi list comprehension kullanarak yapınız.**

```
liste=[ i for i in range(5,21,2)]
print(liste)
```

Doğru Cevap

2	**liste1=[“ahmet”,“ayşe”,“metin”] listesinden list comprehension metodu ile verilerin başına sayın ifadesi gelecek şekilde ([‘Sayın ahmet’, ‘Sayın ayşe’, ‘Sayın metin’]) liste2 adında bir liste oluşturunuz.**

2	**liste1=[“ahmet”,“ayşe”,“metin”] listesinden list comprehension metodu ile verilerin başına sayın ifadesi gelecek şekilde ([‘Sayın ahmet’, ‘Sayın ayşe’, ‘Sayın metin’]) liste2 adında bir liste oluşturunuz.**

```
liste1=["ahmet","ayşe","metin"]
liste2=["Sayın "+i for i in liste1]
print(liste2)
```

Doğru Cevap

3	Elinizde bir teste ait cevap anahtarı ve öğrenci cevaplarının olduğu listeler var. siz bu cevapları anahtar ile kontrol ederek sonuçları doğru veya yanlış olarak çıktı almak istiyorsunuz. Bunu yapacak kodları lambda ve map fonksiyonları ile nasıl yaparsınız.

Örnek: cevap_anahtarı=[“a”,“c”,“e”,“b”,“a”] ogrenci_1=[“a”,“d”,“d”,“b”,“a”]
İstenen çıktı Çıktı: [‘doğru’, ‘yanlıs’, ‘yanlıs’, ‘doğru’, ‘doğru’]

3	**Elinizde bir teste ait cevap anahtarı ve öğrenci cevaplarının olduğu listeler var. siz bu cevapları anahtar ile kontrol ederek sonuçları doğru veya yanlış olarak çıktı almak istiyorsunuz. Bunu yapacak kodları lambda ve map fonksiyonları ile nasıl yaparsınız.** **Örnek: cevap_anahtarı=[“a”,“c”,“e”,“b”,“a”] ogrenci_1=[“a”,“d”,“d”,“b”,“a”]** **İstenen çıktı Çıktı: [‘doğru’, ‘yanlıs’, ‘yanlıs’, ‘doğru’, ‘doğru’]**

```
cevap_anahtari=["a","c","e","b","a"]
ogrenci_1=["a","d","d","b","a"]
liste2=list(map(lambda x,y: "doğru" if x==y else "yanlıs",cevap_anahtari,ogrenci_1))
print(liste2)
```

Doğru Cevap

Teşekkürler