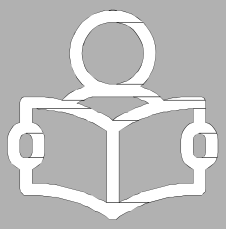
PYTHON

Döngü yapıları

Kapsam



- While Döngüsü
- For Döngüsü
- Continue İfadesi
- İç İçe Döngüler

While Döngüsü

While döngüsü, koşul gerçekleştiği sürece çalışan bir döngü çeşididir. Genellikle döngünün kaç defa çalışacağı belirli değilse while döngüsü tercih edilir. Ancak koşullar verilerek de while döngüsünün belirli sayıda çalışması sağlanabilir. Döngülerde blok yapısı kullanılmaktadır.

Kod

```
sayac=1 #şartın başlangıç değeri
while sayac<6: #sayac 6 dan küçük olduğu sürece
    print("merhaba dünya")
    sayac=sayac+1
```

Sayacın değeri 1 arttırılarak döngünün başına döner ve sayaç değeri 6'ya eşit olana kadar bu durum devam eder.

Döngünün Kapsamı

Döngü koşulu sağlandığı sürece içteki bloklarda bulunan kodlar çalışır. Döngü bittiği zaman Python varsa bir üst katman bloğa dönerek çalışmasına devam eder, yoksa sıradaki satırı işletir.

Kod

```
sayac=1 #şartın başlangıç değeri
while sayac<6: #sayac 6 dan küçük olduğu sürece
    print(sayac)
    sayac=sayac+1
print("döngü sonlandı") #döngü bittiği zaman
```

while döngüsü ile çalışırken sık yapılan hatalardan birisi döngü içerisinde koşulu sağlayan değişkenin değerini arttırma işleminin unutulmasıdır. Bu durumda koşul sürekli sağlanacağı için döngü sürekli çalışır ve dışarıdan bir müdahale ile sonlandırılması gerekir.

Örnek 1

while döngüsü kullanarak 1-100 arasındaki (100 dahil) çift sayıları bularak ekrana yan yana yazan programı yazalım.

Kod

```
a=1
while a<=100:
    if a%2==0:
        print(a, end=",")
    a=a+1
```

Örnek 2

while döngüsü kullanarak 1 - 100 arasındaki sayıların toplamını bulan programı yazalım.

Kod

```
toplam=0
i=1
while i<=100:
    toplam=toplam+i
    i=i+1
print("sayıların toplamı",toplam)
```

While True – Break İfadeleri ve Sonsuz Döngüler

Program yazarken bazen döngünün ne zaman sonlanacağı bilinmeyebilir. Örnek olarak bir markette müşterilerin alışveriş yaparak sepetlerini doldurdukları ve sepette kaç adet ürün olduğu bilinmeyebilir. O müşteriye ait tüm ürünler barkod okuyucu ile okutulmalı ve toplam tutar hesaplanmalıdır.
İşte bu gibi belirsiz durumlar için while döngüsü ile beraber True ifadesi kullanılır. break ifadesi ise döngü sürekli çalışırken istenilen bir anda döngüden çıkmak için kullanılır.

Kod

```
i=1
while True:
    print(i)
    i+=1
    if i==6:
        break
print("döngü sonlandı")
```

While True – Break İfadeleri ve Sonsuz Döngüler

while döngüsü ile kullanılan True ifadesinin yerine farklı kullanımlarla da karşılaşılabilir. Örneğin, bir alışveriş yapıldığını ve sürekli ürün girişi yapıldığını düşünülürse “q” harfi girilene kadar yapılan alışverişler listeye eklensin, eğer “q” harfi girilirse döngüyü sonlandırılsın gibi.

Kod

```
liste=[]
while 2: #buraya 2 yerine herhangi bir sayısal değer de yazılabilirdi.
    ürün=input("ürün adı giriniz:")
    if ürün=="q":
        break
    liste.append(ürün)
print("girdiğiniz ürünler:",liste)
```

Örnek 3

while döngüsü kullanarak sayı tahmin oyunu yapalım. Kullanıcıdan 1-100 arası bir sayı istenmektedir. Girilen sayı, tahmin edilen sayıdan büyükse daha küçük bir sayı girmesi, küçükse daha büyük bir sayı girmesi istensin. Kullanıcı sayıyı bulana kadar bu işlem tekrar etsin. Ayrıca bir sayaç eklenerek kaç defada bilindiğini bulalım.

Kod

```
sayi=45
sayaç=0
print("1-100 arası bir sayı tuttum tahmin et")
while True:
    sayaç+=1
    cevap=int(input("1-100 arası bir sayı girin: "))
    if cevap>sayi:
        print("daha küçük bir sayı girmelisin")
    elif cevap<sayi:
        print("daha büyük bir sayı girmelisin")
    else:
        print("tebrikler tuttuğum sayıyı bildin")
        break
print("tebrikler {} seferde sayıyı bulabildin".format(sayaç))
```

Örnek 4

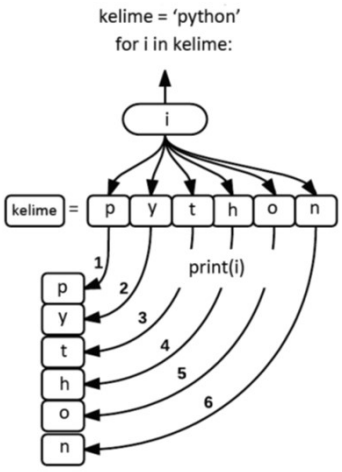
Girilen sayının faktöriyelini hesaplayan programı yazalım.

Kod

```
i=1
f=int(input("faktöriyeli alınacak sayıyı giriniz: "))
sonuc=1
while i<=f:
    sonuc=sonuc*i
    i+=1
print(sonuc)
```

for Döngüsü

for döngüsü, döngünün tekrar sayısı programcı tarafından belirlenmiş veya öngörülmüş ise kullanılır. for döngüsünün iterasyon denilen önemli bir özelliği bulunmaktadır. İterasyon ile karakter dizileri ve listeler üzerinde ilk elemandan son elemana kadar işlem yapılabilmektedir. for döngüsü kullanmak için "in" işlecinden faydalanmak gerekmektedir.




```
kelime="python"
for i in kelime:
   print(i)
```

in İşleci

in işleci bir değerin, bir liste ya da karakter dizisi içerisinde olup olmadığını kontrol eder. Önce karakter dizisi içinde bir karakter olup olmadığına bakar.
Eğer değer karakter dizisi içerisinde varsa True, yoksa False değeri döndürür.

Kod

```
"p" in "python"
liste=[1,2,3,4,5,6]
3 in liste
```

Karakter Dizileri Üzerinde İterasyon İşlemi

Python'da for döngüsü ile karakter dizileri üzerinde kolaylıkla iterasyon işlemi yapılabilir.

Kod

```
isim="Mustafa"
for i in isim:
    print(i, end=",")
```

Kod

```
isim="Mustafa"
i=0
while i<len(isim):
    print(isim[i],end=",")
    i=i+1
```

while döngüsü ile karakter dizileri üzerinde işlem yapılmak istendiğinde hem daha fazla kod yazılması hem de daha karışık bir yapı kullanılması gerekmektedir.
Python'da genellikle listeler veya karakter dizileri üzerinde işlem yapılmak istenildiği zaman veya iterasyon yapılacağı zaman for döngüsü kullanılmaktadır.

Örnek 5

Bir paragraf içerisinde geçen bir harfin kaç defa geçtiğini bulalım.

Kod

```
yazi='''Python üst düzey basit sözdizimine sahip, öğrenmesi
oldukça kolay,modülerliği, okunabilirliği destekleyen, platform
bağımsız nesne yönelimli yorumlanabilir bir script dilidir.'''
harf="a"
sayac=0
for i in yazi:
    if i==harf:
        sayac+=1
print(sayac)
```

Örnek 6

İki farklı karakter dizisi belirleyerek, birinci de olup, diğerinde olmayan karakterleri bulalım.

Kod

```
cumle1='''Python üst düzey basit sözdizimine sahip, öğrenmesi oldukça
kolay, modülerliği, okunabilirliği destekleyen, platform bağımsız
nesne yönelimli yorumlanabilir bir script dilidir.'''
cumle2="Python interaktif yani etkileşimli bir programlama dilidir."
for i in cumle2:
    if not i in cumle1:
        print(i, end=",")
```

Listeler Üzerinde İterasyon İşlemi

Python'da karakter dizilerinde yapılan işlem gibi listeler üzerinde de iterasyon işlemi yapılabilir.

Kod

```
yazi='''Python üst düzey basit sözdizimine sahip, öğrenmesi
oldukça kolay, modülerliği, okunabilirliği destekleyen,
platform bağımsız nesne yönelimli yorumlanabilir bir script
dilidir.'''
yazi=yazi.split()
kelime="üst"
sayac=0
for i in yazi:
    if i==kelime:
        sayac+=1
print(sayac)
```

Örnek 7

İçerisinde sayısal değerler olan bir listedeki değerlerin karesini alarak başka bir listeye atayalım.

Kod

```
sayılar = [1,2,3,4,5]
kareler = []
for i in sayılar:
    kareler.append(i*i)
print(kareler)
```

Örnek 8

Bir liste içerisinde bulunan değerlerin ortalamasını bulalım.

Kod

```
sayılar = [1,2,3,4,5,6,7,8,9]
toplam=0
for i in sayılar:
    toplam=toplam+i
print("sayıların ortalaması: ",toplam/len(sayılar))
```

Örnek 9

Listedeki değerlerden 3'ün katları olan sayıları ekrana yazdıralım.

Kod

```
sayılar = [5,8,12,17,25,36,41,49,60,72]
for i in sayılar:
    if i%3==0:
        print(i,end=",")
```

Örnek 10

İç içe listeler üzerinde gezinme işlemi yapılabilir. [[3,4],[7,8],[10,11],[14,15]] şeklinde bir liste olduğunu varsayalım. Liste elemanları üzerinde klasik bir şekilde gezinme işlemi yapılmak istenirse:

Kod

```
liste = [[3,4],[7,8],[10,11],[14,15]]
for i in liste:
    print(i,end=",")
```

Listeye erişildi, ancak alt listelere erişmek için aşağıdaki kod kullanılır.
Fakat for döngüsündeki iterasyon sayacı sayısı hepsi eşit olan alt liste boyutu kadar olmalıdır.

Kod

```
liste = [[3,4],[7,8],[10,11],[14,15]]
for i,j in liste: #2 sayaç var
    print(i,end=",")
    print(j,end=",")
```

Kod

```
liste = [[3,4],[7,8],[10,11],[14,15]]
for i in liste:
    for j in i:
        print(j,end=",") #bu kod daha iyi
```

range Fonksiyonu ile for Döngüsü Kullanımı

Python'da for döngüsüyle belirli değerler arasında döngü kurmak istenirse range fonksiyonu kullanılmalıdır.
range fonksiyonu bir sayı dizisi oluşturur ve bu sayede oluşturulan sayı dizisi üzerinde for döngüsünün iterasyon yapması sağlanır.

range fonksiyonu, girilen aralık arasında integer değerler oluşturur.
Başlangıç değerinden bitiş değerinin 1 eksiğine kadar değer oluşturur.
Aralık belirtilmediğinde yani sadece bitiş değeri verildiğinde başlangıç değeri 0 alınır.

Kod

```
print(*range(10)) #0 1 2 3 4 5 6 7 8 9
print(*range(5,20)) #5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
print(*range(1,20,3)) #1 4 7 10 13 16 19
```

Örnek 11

for döngüsü ile 0'dan 20'ye kadar olan sayıların toplamını bulalım.

Kod

```
toplam=0
for i in range(21):
    toplam=toplam+i
print("girdiğiniz sayıların toplamı:",toplam)
```

Örnek 12

for döngüsü ile 20'den 1'e kadar olan sayıları yazdıralım.

Kod

```
for i in range(20,0,-1):
    print(i,end=",")
```

Örnek 13

100'e kadar 5'in katları olan sayıyı bulalım.

Kod

```
for i in range(0,101,5):
    print(i,end=",")
```

Continue İfadesi

Hatırlanacağı üzere break ifadesi döngünün dışına çıkılmasını sağlamaktadır. continue ifadesi ise döngünün baştan sona kadar çalışmasını engellemeyen ancak belirli durumlar sağlandığında o adımı atlamamızı sağlayan yapılardır.

Kod

```
for i in range(1,6):
    if i ==2 or i==4:
        continue
    print(i)
```

İç İçe Döngüler

Döngü bloklarının içerisine kodlar eklenebileceği gibi, koşul durumları hatta başka bir döngü koymak bile mümkündür. Bir döngü yapısının içine başka bir döngü yapısının yerleştirilmesi ile elde edilen yapıya iç içe döngü (nested loop) adı verilir.

Kod

```
dersler = ["Ders 1", "Ders 2", "Ders 3"]
konular = ["Konu 1", "Konu 2", "Konu 3"]
for x in dersler:
    for y in konular:
        print(x, y)
```

Örnek 14

İç içe while döngüsü kullanarak, i ve j olarak tanımlanan iki değişkenin değerlerini ekrana yazdıralım.

Kod

```
i=1
while i<4:
    j=1
    while j<4:
        print("i nin değeri: {} j nin değeri: {} ".format(i,j))
        j=j+1
    i=i+1
```