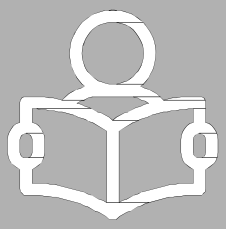
PYTHON

Yorum Satırları, Değişkenler, Veri Tipleri ve Operatörler

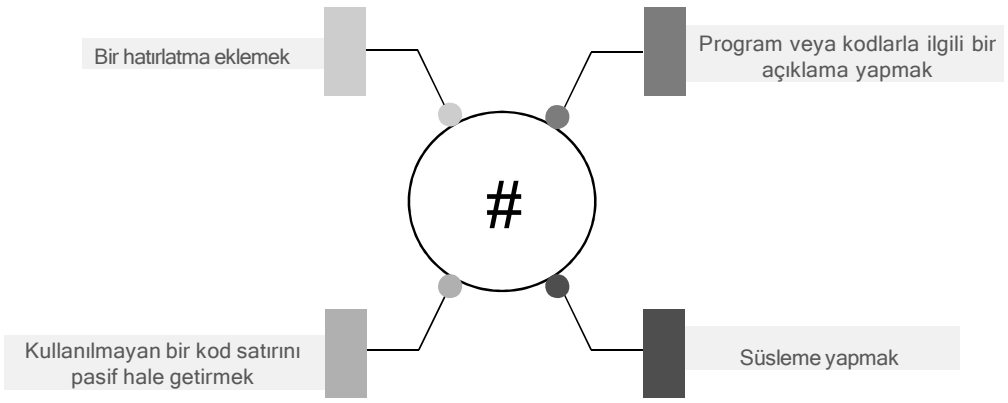
Kapsam



- Yorum Satırlarını Kullanma
- Değişkenler
- Veri Tipleri
- type( ) Metodu Kullanımı
- Veri Tiplerini Dönüştürmek
- Operatörler

Yorum Satırlarını Kullanma

Yorum satırları, Python yorumlayıcısı tarafından dikkate alınmayan ve yorumlanmayan ifadelerdir. Python'da yorum satırları genel olarak aşağıdaki işlemler için kullanılır:



Bu tür açıklama satırları kodun başkaları tarafından daha iyi anlaşılmasını sağlar. Python'da tek satırlık açıklama için "#" işareti kullanılır. "#" işareti kullandığınızda o satırdaki metin kod olarak işlenmez.

Örnek 1

Aşağıdaki yorum satırları Python tarafından dikkate alınmamaktadır.
Bu nedenle sadece “print()” komutu çalışır.

Kod

```
#Bu kod ekrana yazı yazılmasını sağlamaktadır.
print("Konu: Yorum satırlarını kullanma")
#Her satırın başına
#"#" işareti eklenerek
#alt alta yorum satırları oluşturulabilir.
```

Örnek 2

Yorum satırlarını kod satırının devamında aynı satırda kullanabilirsiniz. Bu kullanımda önce kod gelir, devamında yorum satırı “#” işareti ile başlar (öndeki kod çalışır), daha sonra satır sonuna kadar yorum satırı olarak dikkate alınmaz.

Kod

```
print(2+3) # Bu kod satırı ekrana 2 sayının toplamını yazar.
```

Örnek 3

Birden fazla yorum satırı kullanılacaksa yorumlar üçlü tek tırnak veya üçlü çift tırnak blokları arasına yazılır.

Kod

```
'''Python'da birden fazla açıklama satırı kullanmak için üçlü tek tırnak veya çift tırnak
kullanılır. Açıklama satırını bitirmek için aynı işaretler kullanılır.'''

"Python'da birden fazla açıklama satırı kullanmak için üçlü tek tırnak veya çift"
"tırnak kullanılır. Açıklama satırını bitirmek için aynı işaretler kullanılır."
```

Örnek 4

Aşağıdaki kod satırları yorum satırı olarak değerlendirilmez.

Kod

```
"#!/usr/bin/env python3" kodu Python 3 yorumlayıcısının Linux için dosya konumunu
belirtir.
"#!c:/Python/python.exe" kodu Python 3 yorumlayıcısının Windows için dosya konumunu
belirtir.
"# -*- coding: utf-8 -*- " Kullanacağınız karakter kodlamasını belirtmek için
kullanılır. utf-8 Türkçe alfabeyi de destekleyen bir karakter kodlama sistemidir.
```

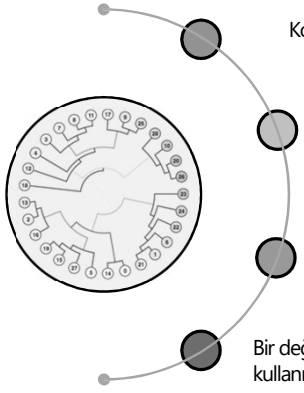
Örnek 5

Yorum satırları süsleme amacıyla da kullanılabilir. Bazı program dosyalarında aşağıdaki gibi süslü açıklamalar, etiketler görülebilir.

Kod

```
'''
#########################################################################
#*********************************************************************##
# Python Öğreniyorum ##
# Python 3 ##
#*********************************************************************##
#########################################################################
'''
```

Değişkenler



Kod yazarken sadece sabit değerler üzerinden işlemler yapılmaz. Kullanıcıdan veya başka kaynaklardan veri alınması gerekir.

İşlem yapabilmek için bu girdiler (değerler) bellekte tutulmalıdır. Bu girdileri depolamak amacıyla bellekte belirli bir yer ayrılması gerekir. Bir değişken tanımlandığında yorumlayıcı, veri türüne bağlı olarak bellekte yer ayırır ve ayrılan bölümde hangi türden verilerin saklanabileceğini belirler.

Değişkenlere farklı veri tiplerinde değerler atanabilir. Değer atama ile (bir değişkeni bir değere eşleyerek) tam sayı, ondalık sayı, dizi veya karakter dizisi türünde değerler değişkenlerde tutulabilir. Python, bu konuda çok esnektir. Python'da değişkenlerinin veri tiplerini açıkça bildirmeye gerek yoktur. Aynı değişkene önce sayı, sonra bir metin daha sonra başka türde bir değer atanabilir.

Bir değişkene değer atandığında veri tipi otomatik olarak tanımlanır. Eşittir operatörü "=" değişkenlere değer atamak için kullanılır. "=" operatörünün solunda değişkenin adı ve "=" operatörünün sağında ise bu değişkene atanacak değer yer alır.

Örnek 6

Değişken oluşturma ve değer atamaya ilişkin örnekler:

Kod

```
#sayi değişkenine ilk olarak 5 sayısı atandı.
sayi=5
print('Değişken: ', sayi)
sayi=10
print('Değişken: ', sayi, 'oldu')
sayi='Python'
print ('Değişken: ', sayi, 'oldu')
sayi=10.5
print ('Değişken: ', sayi, 'oldu')
```

Örnek 7

Aşağıda 3 değişkene de tek satırda 1 değeri atanmıştır.

Kod

```
a = b = c = 1
print ('1. sayı=', a)
print ('2. sayı=', b)
print ('3. sayı=', c)
```

Örnek 8

Değişkenler aralarına virgül eklenerek yan yana yazılır. Değerleri de aynı sıralama ile karşılarına yazılır.

Kod

```
adi, soyadi, yasi='Ali', 'CAN', 34
print ("Adı=", adi)
print ("Soyadı=", soyadi,)
print ("Yaşı=", yasi)
```

Örnek 9

Değişkenlere değer atamak için başka bir yöntem aralarına noktalı virgül ";" ekleyerek değişken - değer ikilileri şeklinde yazmaktır.

Kod

```
adi='Aziz'; soyadi='SANCAR'; yasi=74
print ("Adı=", adi)
print ("Soyadı=", soyadi,)
print ("Yaşı=", yasi)
```

Örnek 10

Değer atanmayan ve/veya tanımlanmamış bir değişken kullanılırsa Python hata verir.

Kod

```
print (yenisayi)
```

Değer atamadan tanımlamak için yenisayi=int( ) kodu kullanılabilir. Bu durumda değişkene ilk değer olarak “0” sıfır atanır.

Kod

```
yenisayi=int( )
print(yenisayi)
```

Örnek 11

Değişkenler veri tiplerine göre kullanılmazsa Python hata verir.

Kod

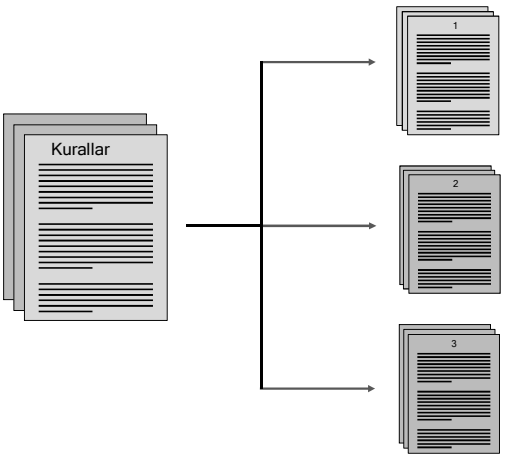
```
sayi1=1
metin1='deneme'
print(sayi1+metin1) #Bir sayı ile bir kelime toplanamaz.
```

Değişken Adlandırmada Kurallar

Değişken adı verilirken uyulması gereken bazı kurallar ve kurallar kadar katı olmasa da yararlı kullanım önerileri vardır.

abc ✓ _abc ✓ 3a ✗ @abc ✗

a100 ✓ _a984_ ✓ a9967$ ✗ xyz-2 ✗



Değişkenler bir harf (a - z, A - Z) veya alt çizgi (_) ile başlamalıdır. Bunların dışında sayı veya başka bir karakter ile başlayamaz.

Değişken adında rakam, alt çizgi(_), büyük veya küçük harf olabilir.

Değişken adları herhangi bir uzunlukta olabilir.

Python, değişken adlandırmada Türkçe karakterlerin (ç , ğ, ı, ö, ş ve ü) kullanımına izin vermektedir.

Değişken Adlandırmada Kurallar

Python programlama dilinde ayrılmış sözcükler kullanılamaz. Bu özel sözcüklerin listesini görmek için aşağıdaki kod kullanılabilir.

Kod

```
import keyword
keyword.kwlist
```

Not: Büyük harf ve küçük harf kullanılarak tanımlanan değişkenlerin adı aynı olsa bile farklı değişkenler olduğu unutulmamalıdır.

Örnek 12

Python dilinde uygun değişken adı örnekleri:

Kod

```
sayi1=1
Sayi1=2
sayı1=3
```

Sayi1 ekrana yazdırılırsa çıktı ne olur?

Kod

```
print (sayi1)
print(Sayi1)
print (sayı1)
```

Örnek 13

Python'da hatalı değişken adı kullanımı örneği:

Kod

```
1sayi=5
```

Değişken Adlandırma için Standartlar

Bu standartlar değişken adının ve içeriğinin anlaşılmasına yardımcı olarak programcıların daha kolay çalışmasını sağlar. Değişken adı, onun içeriği hakkında bilgi verirse kodun anlaşılması kolaylaşır.

Snake

- Birden fazla kelimenin kullanılacağı değişken adlarında *kelimelerin arasına alt çizgi (_) konur*.
  - degisken_adi
  - dogum_yılı
  - medeni_durum

Camel

- Birden fazla kelimenin kullanılacağı değişken adlarında *ilk kelime hariç sonraki kelimelerin ilk harfi büyüktür*.
  - degiskenAdi
  - dogumYili
  - medeniDurum

Örnek 14

Camel standardı ve snake standardı örneği:

Kod

```
adıSoyadı='Ali Can'
adı_soyadı='Veli Can'
```

Not: Farklı standartlar kullanılarak tanımlanan değişkenlerin adı aynı olsa bile farklı değişkenler olduğu unutulmamalıdır.

Örnek 15

Aşağıda bir sayı ile bir karakter dizisi
üzerinde operatörleri kullanarak işlemler yapılmıştır.

Kod

```
sayi1=5
sayi2='3'
print (sayi1+sayi2)
```

Kod çalıştırıldığında bir hata mesajı alınır. Bir aritmetik operatörü kullanılırken bir sayı ile bir karakter dizisi (ikinci değişkenin adı sizi yanıltmasın) toplamaya çalışıldığı için Python hata verir. Veri tipleri, değişkenler üzerinde yapılabilecek işlemleri belirler.

Örnek 16

Operatörler konusunda * operatörü ile ilgili bir ayrıntıdan bahsedilmiştir.

Kod

```
sayi1=5
sayi2='3'
print (sayi1*sayi2)
```

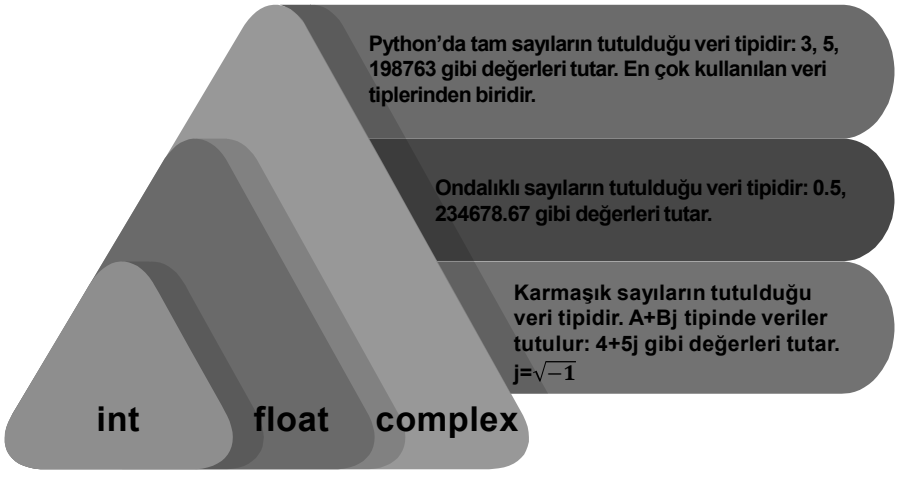
Örnekte görüldüğü gibi kod "çarpma" işlemi yapamamıştır. Çünkü ortada iki sayısal değer yoktur. Bu örnekte 3 sayısını bir karakter olarak 5 defa yazmıştır.

Python'da değişkenlere değer atanırken veri tipleri belirtilmez. Python, atanan değere göre veri türünü kendisi belirler. Ancak programlama yaparken veri tiplerini bilmek ve ona göre kullanmak gerekir.

Python'da Veri Tipleri

Veri Tipi	Sınıfı	Açıklama
Integer	int	Tam sayı. (Örnek: 3, 5, 369963)
Float	float	Ondalıklı sayı. (Örnek: 10.45)
Complex	complex	Karmaşık sayılar A + Bj şeklinde kullanılır. (Örnek: 4+5j)
Karakter dizisi (String)	str	Karakter dizilerini (metinleri) göstermek için kullanılır. Çift tırnak veya tek tırnak içinde gösterilir. "Merhaba Dünya"
Boolean	bool	Sadece True veya False değeri alır. int(True)=1 iken int(False)=0 dır.
Liste	list	Farklı veri türleri içerebilir. listem=['Çınar', 24, 'Mühendis', True]
Demet (tuple)	tuple	Farklı veri türleri içerebilir. demet1=('Çınar', 24, 'Mühendis', True)
Sözlük (dictionary)	dict	Farklı veri türleri içerebilir. sozluk={'adi': 'Çınar','yasi'=24, 'meslekUnvani':'Mühendis', 'askerlikDurumu': True}

Sayısal (int, float ve complex) Veri Tipleri



Örnek 17

Python'da bir değişkene değer atandığında veri tipleri atanan değere göre otomatik olarak belirlenir.

Kod

```
piSayisi=3.14 #float tipinde bir veri
rCm=2 #integer tipinde veri
karmasikSayi=4+5j
```

Karakter Dizisi (string) Veri Tipi

Karakter dizisi, kullanıcıdan alınan değerlerin metin formatında tutulduğu veri tipleridir. Python karakter dizisi oldukça kullanışlı işlevlere sahiptir.

Bir karakter dizisi ekrana yazdırılabilir, başka bir karakter dizisiyle birleştirilebilir. "len( )" metodu bir karakter dizisinin uzunluğunu vermektedir.

Örnek 18

Aşağıda karakter dizisi ile ilgili örnek verilmiştir.

Kod

```
metin1 = 'Merhaba '
metin2 = 'Python'
print (metin1) #karakter dizisinin tamamını yazar
print (metin1 * 2) #karakter dizisini 2 defa yazar
print (metin1 + metin2) #iki karakter dizisini birleştirir
print ('metin1 adlı değişkendeki değerin uzunluğu: ', len(metin1))
```

Boşluğun da bir karakter olduğunu gözden kaçırmayınız!

Karakter Dizilerinde (string) Dilimleme İşlemleri

Bir karakter dizisinin içindeki karakterlere tek tek veya belirli bir aralıkta erişilebilir. Köşeli parantez içinde [ ] tek bir sayı verildiğinde bu karakter dizisinin indisini ifade eder. İndis numarası "0" dan başlayarak karakterin metindeki kaçıncı sırada yer aldığını gösterir. metin[0] ifadesi metin değişkenindeki 1. karakteri verir.

Belirli bir aralıktaki karakteri alırken "[başlangıç indisi:bitiş indisi]" şeklinde ifade edilir. metin[0:5] metin değişkeninde indisi 0, 1, 2, 3 ve 4 olan karakterleri dilimler. Bitiş indisi dilimlemeye dahil edilmez.

Başlangıç indisi verilmeyen karakter dizisinde örnek: metin[:7] başlangıç indisi otomatik olarak sıfır (0) olur; 0, 1, 2, 3, 4, 5 ve 6. karakterlerden oluşan bir metin verir. Bitiş indisi değeri verilmezse başlangıç indisinden başlanarak son karakter dâhil dilimleme işlemi yapılır.

İndislerin "0" dan başladığı unutulmamalıdır.

Negatif İndis Sayıları

Karakter dizisine sondan başlandığını ifade eder.

metin[-1] karakter dizisinin en sonundaki karakteri verir.
metin[-2] ise sondan ikinci karakteri verir.

Karakter dizilerinde indisler ritmik atlanarak dilimleme yapılabilir.

Bunun için [başlangıç indisi: bitiş indisi: ritmik artış miktarı] şeklinde kullanılır.
"metin[0:8:2]" : 0'dan başlayarak 0, 2, 4 ve 6 indis numaralı karakterleri dilimler.

Örnek 19

Karakter dizilerinin kullanımı ile ilgili örnekler:

Kod

```
metin='Merhaba Python'
print (metin[0]) # ifadenin ilk karakterini yazar.
print (metin[4:7]) # ifadenin 5, 6 ve 7. karakterlerini yazar.
print (metin[8::]) # 9. karakterden sonuncu karaktere kadar yazar.
print (metin[-2]) # karakter dizisinin en sondan ikinci karakterini yazar.
print (metin [:7]) # indisi 0' dan 7'ye kadar olan (7 dahil değil) karakterleri
yazar.
print (metin[8:]) # başlangıç indisinden sonra tüm karakterleri yazar.
print(metin[0:8:2]) # 0, 2, 4 ve 6 indis numaralı karakterleri dilimler.
```

type( ) Metodu Kullanımı

Python, her ne kadar veri tiplerini otomatik olarak verse de bir değişkenin veri tipini kontrol etmek ve kullanım amacına göre değiştirmek gerekebilir. Bir değişkenin veri tipini öğrenmek için "type( )" komutu kullanılır. type(degiskenAdi) şeklinde yazılır.

Örnek 20

Kod

```
metin = 'Merhaba'
sayi1 = 23
sayi2 = 6.34
sayi3 = 2+3j
type(metin)
type(sayi1)
type(sayi2)
type(sayi3)
```

Veri Tiplerini Dönüştürmek

Veri tipi, kullanılan değerler ile uyumlu olmalıdır. Veri tipi dönüşümünde bazı değerlerde kayıplar olabilir.

int( )
Veri tipini integer'a çevirir.

int(4.3) #doğru kullanım
int("45") #doğru kullanım
int("Ali") #hata verir
int("7.5") #hata verir

"integer" tipinde bir sayıyla "float" tipinde bir sayı çarpıldığında sonuç "float" tipinde olacağı için Python bu veri tipini otomatik olarak belirler.

float()
Veri tipini float'a çevirir.

float(5) #doğru kullanım
float("4.6") #doğru kullanım
float("Ali") #hata verir
float("7") #doğru kullanım

str()
Veri tipini karakter dizisine çevirir.

str(3) #doğru kullanım
str(3.3) #doğru kullanım

Örnek 21

Aşağıdaki örnekte iki sayının da tırnak içinde verilmiş olduğuna dikkat ediniz.

Kod

```
metin1='5'
metin2='3'
print (metin1+metin2)
```

Yukarıda kullanılan değerler tırnak içinde verildiğinden karakter dizisi veri tipindedir. Kod sonuç olarak iki karakteri yan yana yazacaktır.

Örnek 22

İki değişkenin veri tipini dönüştürünüz. Veri tipi dönüştürüldüğünde karakter dizisi sayıya çevrilmiş olacaktır. Böylece iki sayısal değer üzerinde toplama işlemi yapılabilir.

Kod

```
metin1='5'
metin2='3'
print(int(metin1)+int(metin2))
```

Örnek 23

Aşağıdaki örnekte pi değerinin “float” ve “integer” veri tiplerinde kullanıldığında çıkan sonuca dikkat ediniz.

Kod

```
piDegeri=3.14
type(piDegeri)
yariCap=5
daireninAlani=(2*piDegeri*yariCap)
print(daireninAlani)
piDegeriInt=int(piDegeri)
print(piDegeriInt)
daireninAlani=(2*piDegeriInt*yariCap)
print(daireninAlani)
```

Örnek 24

Boolean veri tipini sayısal veri tipine ve string veri tipine dönüştürebilirsiniz. Bu işlemin tersini de yapabilirsiniz.

Kod

```
int(True) #1 verir
int(False) #0 verir
bool(123)#True verir
bool(-123)#True verir
bool(0)#False verir
str(True)#’True’ verir
str(False) #’False’ verir
bool("Ali")#True verir
bool("98989")#True verir
```

Aritmetik Operatörler

Toplama, çıkarma, çarpma ve bölme gibi işlemler başta olmak üzere aritmetik işlemleri yapmak için kullanılan operatörlerdir.

İşaret	İşlem	Örnek	Sonuç
+	Toplama	5+3	8
-	Çıkarma	5-3	2
*	Çarpma	5*3	15
/	Bölme	5/3	1.6666666666666667
**	Kuvvet	5**3	125
//	Tam sayı bölme	5//3	1
%	Mod	5%3	2

Toplama Operatörü

Bu operatör iki veya daha fazla sayısal değeri toplamak için kullanılır.
Sayısal değerler “integer” , “float” veya “complex” tipinde olabilir.

Örnek 25

Kod

```
print(5+3)
print(5+3+3)
sayi1=10
print(sayi1+5)
sayi2=10.34 #Aynı şekilde değişken kullanarak da yapabiliriz.
print(sayi1+sayi2+5.5)
```

Toplama Operatörü

Toplama operatörünün karakter dizilerinde farklı bir kullanımı vardır: İki veya daha fazla karakter dizisini birleştirmek için kullanılabilir.

Örnek 26

Kod

```
print ('Merhaba ' + 'Python')
metin1='Merhaba ' + 'Python' + ' nasılsın?'
print(metin1)
```

Çıkarma Operatörü

Bu operatör, sayıların farkını bulmak için kullanılır.

Örnek 27

Kod

```
print(5-3)
sayi1=5
sayi2=3
print(sayi1-sayi2)
print(3-4-5)
```

Çarpma Operatörü

Çarpma operatörü iki veya daha fazla sayıyı çarpmak için kullanılır.

Örnek 28

Kod

```
print(5*3)
print(5*3*2)
sayi1=5
sayi2=3
print(sayi1*sayi2)
```

Çarpma Operatörü

Çarpma operatörü, toplama operatöründe olduğu gibi karakter dizilerinde farklı bir amaç için kullanılmaktadır.
Bir karakter dizisini istediğiniz kadar yazdırmak için çarpma operatörünü kullanabilirsiniz.

Örnek 29

Kod

```
print (20*'BA')
```

Bölme Operatörü

Bu operatör, iki sayıyı bölmek için kullanılır.

Örnek 30

Kod

```
print (5/3)
```

Bölme Operatörü

Bölme operatörünü ikiden fazla sayıyı bölmek için de kullanabilirsiniz.

Örnek 31

Kod

```
print(18/3/2)
```

Bu tür durumlarda Python işlemleri soldan başlayarak sırayla yapar. Önce 18/3 işlemini yapar ve sonucu 2'ye böler.

Kuvvet Alma Operatörü

Bir sayının kuvvetini almak için kullanılır. Bir sayının kuvveti o sayının kendisiyle kuvvet kadar çarpılmasıyla bulunur.

Örnek 32

Kod

```
print(5**3)
```

Kuvvet Alma Operatörü

Kuvvet alma operatörü bir sayının kökünü almak için “1/kuvvet” olarak kullanılabilir.

Örnek 33

Kod

```
print(49**(1/2))
```

Tam Bölüm Operatörü

İki sayının birbirine tam bölüm sonucunu verir. Bölüm sonucu ondalıklı sayı ise ondalıklı kısmını almaz.

Örnek 34

Kod

```
print (121//3)
```

Mod Alma Operatörü

Bir sayının diğer bir sayıya bölümünden kalanını verir.

Örnek 35

Kod

```
print (5%3) #Sayının 3 ile bölümünden kalan 2.
print (9%2) #Kalan 0 ise sayı çifttir.
```

İlişkisel Operatörler

İlişkisel operatörler, değerler arasındaki ilişkiyi kontrol ederek sonucu "boolean" bir değer olarak (True, False) döndürür. "True" değeri şartın, ilişkinin veya koşulun sağlandığı anlamına gelirken, "False" değeri ise ilişkinin sağlanmadığı anlamına gelir.

İşaret	İşlem	Örnek	Sonuç
==	Eşit mi?	5==3	False
!=	Farklı mı?	5!=3	True
>	Büyüktür?	5>3	True
<	Küçüktür?	5<3	False
>=	Büyük veya eşittir?	3>=3	True
<=	Küçük veya eşittir?	5<=3	False
is	Değerler eşit mi?	'Elif' is 'elif'	False
is not	Değerler farklı mı?	'Elif' is not 'elif'	True
in	İçeriyor mu?	'bil' in 'bilişim'	True
not in	İçermiyor mu?	'bil' not in 'bilişim'	False

Eşittir Operatörü

"==" operatörü iki değerin birbirine eşit olup olmadığını anlamak için kullanılır. İki değer birbirine eşitse "True", eşit değilse "False" değeri verir.

Örnek 36

Kod

```
print(5==3)
#değişkenlere atadığınız değerleri de aynı şekilde kontrol edebilirsiniz.
sayi1=5
sayi2=3
print(sayi1==sayi2)
print(sayi1==5)
```

Eşittir Operatörü

"==" operatörü karakter dizilerinde de değerlerin eşitliğini kontrol etmek için kullanılır.

Örnek 37

Kod

```
print('Emre'=='emre')
# Küçük büyük harf duyarlılığına dikkat ediniz.
metin1='Emre'
metin2='emre'
print(metin1==metin2)
print(metin1=='Emre')
```

Eşit Değildir Operatörü

"!=" operatörü iki değerin birbirinden farklı olup olmadığını anlamak için kullanılır. "==" operatörünün tersine değerler birbirine eşitse "False", eşit değilse "True" değerini döndürür.

Örnek 38

Kod

```
print(5!=3)
sayi1=5
sayi2=3
print(sayi1!=sayi2)
print(sayi1!=5)
```

Eşit Değildir Operatörü

"!=" operatörü karakter dizilerinde de değerlerin farklılığını kontrol etmek için kullanılır.

Örnek 39

Kod

```
print('Emre'!='emre')
# Küçük büyük harf duyarlılığına dikkat ediniz.
metin1='Emre'
metin2='emre'
print(metin1!=metin2)
print(metin1!='Emre')
```

Büyüktür Operatörü

Büyüktür ">" operatörü iki değeri karşılaştırmak için kullanılır. 1. sayı 2. sayıdan büyükse "True" değilse "False" değerini döndürür.

Örnek 40

Kod

```
sayi1=6.06
sayi2=6.07
print(sayi1>sayi2)
```

Küçüktür Operatörü

Küçüktür “<” operatörü iki değeri karşılaştırmak için kullanılır. 1. sayı 2. sayıdan küçükse “True” değilse “False” değerini döndürür. Büyüktür operatörünün tersi işlevini görür.

Örnek 41

Kod

```
sayi1=6.06
sayi2=6.07
print(sayi1<sayi2)
```

Karakter dizileri, alfabetik olarak sıralandığında sonra gelen ifade daha büyük olarak değerlendirilir.
print ('z'>'a') #True
print ('a'<'z') #True

Büyük Eşittir (>=) ve Küçük Eşittir Operatörleri

Büyük eşittir “>=” operatörü iki değeri karşılaştırmak için kullanılır. Birinci değer ikinciden büyükse veya ikinciye eşitse “True” değilse “False” değerini döndürür.
Küçük eşittir “<=” operatörü ise birinci değer ikinci değerden küçükse veya ikinci değere eşitse “True” değilse “False” değerini döndürür.

Örnek 42

Kod

```
sayi1=6.06
sayi2=6.06
print(sayi1>=sayi2)
print (sayi1<=sayi2)#iki operatör de True değeri döndürür.
sayi2=6.07
print(sayi1>=sayi2)
print(sayi1<=sayi2)#sadece <= operatörü True değerini döndürür.
sayi2=6.05
print(sayi1>=sayi2)
print(sayi1<=sayi2)#sadece >=operatörü True değeri döndürür.
```

"is" ve "is not" Operatörleri

"is" operatörü ve "==" operatörü benzer işleve sahiptir ve iki değerin eşit olup olmadığını kontrol etmek için kullanılır. Değerler eşitse "True" değilse "False" değerini döndürür.
"is not" operatörü ise "is" operatörünün tersi işlev görür. "is not" operatörü değerler farklı ise "True" değerini değerler aynıysa "False" değerini döndürür.

Örnek 43

Kod

```
sayi1=5
print (sayi1 is 5)
print (sayi1 is not 5) #is operatörünün tersini verir.
```

NOT

"==" operatörü değerlerin eşitliğini kontrol ederken "is" aynı zamanda her iki değerin aynı nesneyi referans gösterip göstermediğini kontrol eder.

"is" ve "is not" Operatörleri

"is" operatörü karakter dizisinde de kullanılabilir.

Örnek 43

Kod

```
print ('elif' is 'Elif')#büyük harf küçük harf duyarlılığını hatırlayınız.
adi='Elif'
print (adi is 'Elif')
print (adi is not 'Elif') #is operatörünün tersini verir.
```

"in" ve "not in" Operatörleri

"in" operatörü bir karakter dizisinin başka bir karakter dizisinde yer alıp almadığını kontrol etmek için kullanılır. Karakter dizisi, diğer karakter dizisi içinde yer alıyorsa "True" değeri, yer almıyorsa "False" değeri döndürür.
"not in" operatörü ise içinde yer almıyorsa "True" yer alıyorsa "False" değeri döndürür.

Örnek 44

Kod

```
print ('Bil' in 'Bilişim')#2. karakter dizisi içinde 1. karakter dizisi var mı?
print ('Bil' not in 'Bilişim')# in operatörünün tersi sonuç verir.
```

NOT

"in operatörünün" for döngüsünde işlevsel bir kullanımı vardır. "Döngüler" konusunda bu kullanımına yer verilmiştir.

Atama Operatörleri

Atama operatörleri değişkene değer atamak için kullanılır. Değişkeni başka bir değerle işleme alarak sonucun yine aynı değişkene atanmasını sağlar.

İşaret	İşlem	Örnek	Sonuç
+=	Artırarak atama	sayi1+=3	8
-=	Eksilterek atama	sayi1-=3	2
*=	Çarparak atama	sayi1*=3	15
/=	Bölerek atama	sayi1/=3	1.6666666666666667
**=	Kuvvet alarak atama	sayi1**=3	125
//=	Tam sayı bölerek atama	sayi1//=3	1
%=	Mod alarak atama	sayi1%=3	2

NOT

sayi1=5 olarak atanmıştır.

Artırarak Atama Operatörü

Bir değişkenin üstüne sayısal değerleri toplayarak sonucun aynı değişkene atanmasını sağlar.
Aşağıdaki kod ”sayi1=sayi1+3” koduyla aynı işlevi görür.

Örnek 45

Kod

```
sayi1=5
sayi1+=3
print(sayi1)
metin1='Merhaba '
metin1+='Python' #metin1=metin1+"Python" koduyla aynı işlevi görür.
print(metin1)
```

Eksilterek Atama Operatörü

Bir değişkenden bir sayısal değeri eksilterek sonucun aynı değişkene atanmasını sağlar.
Aşağıdaki kod “sayi1=sayi1-3” koduyla aynı işlevi görür.

Örnek 46

Kod

```
sayi1=5
sayi1-=3
print(sayi1)
```

Çarparak Atama Operatörü

Bir değişkeni bir sayısal değer ile çarparak sonucun aynı değişkene atanmasını sağlar.
Aşağıdaki kod "sayi1=sayi1*3" koduyla aynı işlevi görür.

Örnek 47

Kod

```
sayi1=5
sayi1*=3
print(sayi1)
#Aynı şekilde karakter dizilerinde de kullanabilirsiniz.
metin1='Merhaba '
metin1*=3 #metin1=metin1*3 koduyla aynı işlevi görür.
print(metin1)
```

Bölerek Atama Operatörü

Bir değişkeni bir sayısal değere bölerek sonucun aynı değişkene atanmasını sağlar.
Aşağıdaki kod "sayi1=sayi1/3" koduyla aynı işlevi görür.

Örnek 48

Kod

```
sayi1=5
sayi1/=3
print(sayi1)
```

Kuvvet Alarak Atama Operatörü

Bir değişkenin kuvvetini alarak sonucun aynı değişkene atanmasını sağlar.
Aşağıdaki kod “sayi1=sayi1**3” koduyla aynı işlevi görür.

Örnek 49

Kod

```
sayi1=5
sayi1**=3
print(sayi1)
```

Tam Sayı Bölerek Atama Operatörü

Bir değişken sayıya bölündüğünde sonucun (ondalık kısmını atarak) aynı değişkene atanmasını sağlar.
Aşağıdaki kod “sayi1=sayi1//3” koduyla aynı işlevi görür.

Örnek 50

Kod

```
sayi1=5
sayi1//=3
print(sayi1)
```

Mod Alarak Atama Operatörü

Bir değişkenin bir sayıya bölümünden kalanın aynı değişkene atanmasını sağlar.
Aşağıdaki kod "sayi1=sayi1%3" koduyla aynı işlevi görür.

Örnek 51

Kod

```
sayi1=5
sayi1%=3
print(sayi1)
```

Mantıksal Operatörler

İfadeleri mantıksal olarak bağlamak için kullanılan: "or", "and" ve "not" operatörleridir.

Aritmetik operatörler

"or" Operatörü

"or" operatörü "veya" anlamındadır. Belirtilen koşullardan birinin sağlanması durumunda "True" değeri döndürür. Bir sayı 6'dan küçük veya 10'dan büyükse koşulunu düşünelim. Sayımız 5 ise 6'dan küçük olduğu için bu şartı sağlayacaktır. Sayımız 6, 7, 8, 9 veya 10 olursa şartların her ikisini de sağlamadığı için "False" değeri döndürülür.

Örnek 52

Kod

```
sayi1=5
print(sayi1<6 or sayi1>10)
adi='Elif'
print (adi=='Elif' or adi=='Emre')#Adı Elif veya Emre ise True değerini döndürür.
```

"and" Operatörü

Bu operatör "ve" anlamındadır. Belirtilen koşulların hepsinin sağlanması durumunda "True" değerini döndürür. Bir öğrencinin ders puanı 50'den büyük ve 60'tan küçükse koşulunu düşünüldüğünde öğrencinin puanı 50 ise her iki şartı da sağlayacaktır ve "True" değerini döndürecektir.

Örnek 53

Kod

```
ogrenciDersPuani=50
print (ogrenciDersPuani>50 and ogrenciDersPuani<60)
adi='Emre'
yasi=24
print(adi=='Emre' and yasi>=20)#Adı Emre ve yaşı en az 20 ise True değerini döndürür.
```

Tam Sayı Bölerek Atama Operatörü

Bir değişken sayıya bölündüğünde sonucun (ondalık kısmını atarak) aynı değişkene atanmasını sağlar.
Aşağıdaki kod "sayi1=sayi1//3" koduyla aynı işlevi görür.

Örnek 50

Kod

```
sayi1=5
sayi1//=3
print(sayi1)
```

"not" Operatörü

Bu operatör "değil" anlamındadır. Belirtilen koşulun tersi doğruysa "True" değeri verir. Bir öğrencinin puanı 45'ten küçük değilse ifadesi düşünüldüğünde öğrencinin puanı 50 ise "True" değerini döndürür.

Örnek 54

Kod

```
ogrenciDersPuani=50
print(not(ogrenciDersPuani<45))
print(ogrenciDersPuani>45) #Yukarıdaki ifade ile aynı işlevi görür.
```

Operatörlerde Öncelik Sırası

Farklı operatörler birlikte kullanılabilir. Operatörler birlikte kullanılırken hangi işlemin önce yapılacağına dair bir sıralama vardır.

Parantez içindeki işlemler her zaman öncelikli olarak yapılır.

Çarpma ve bölme işlemleri toplama ve çıkarma işlemine göre önce yapılır.

Aynı derecedeki operatörlerde işlem sırası önceliği soldan sağa doğrudur.

İşlemlerde öncelik sırasını belirtmek için en iyi yöntem operatörleri parantez içinde kullanmaktır.

Örnek 55

Kod

```
print((3+5)*2) #Öncelikle parantez içi yapıldığında 8*2=16
print (3+5*2) #Öncelikle çarpma işlemi yapıldığından 3+10=13
print (3**2*2)
print (6*7/7)
print (6*3/2+8/2*3)
```