Python Temel Kavramlar

Sayılar, Değişken Tanımlama, Temel Matematik Operatörler

Python' da sayılar: Tam sayılar (integer) ve ondalıklı sayılar (float) Python' da bir veri tipidir.

Temel Matematik operatörler: Toplama (+), Çıkarma (-), Çarpma (*) ve Bölme (/)

Değişken tanımlama: Değişken, bir veri tipinden (sayısal (numeric) ya da sözcük (string)) değer tutan birimlerdir ya da objelerdir. Sayısal bir değişken, **değişkenin ismi = sayı değeri** olarak tanımlanır.

Değişken ismi verilirken aşağıdaki durumlara dikkat edilmelidir.

- Değişken ismi sayı ile başlayamaz.
- Değişken ismi, isim tamlaması gibi ise kelimeler arasında boşluk bulunamaz.
- :'",<>/?|\()!@#$%^&*~-+ simgeleri değişken isminde kullanılamaz. Sadece alt çizgi _ kullanılabilir.
- Python' da tanımlı ***while***, ***not*** gibi özel kelimeler değişken ismi olarak kullanılamaz.

Aslında bu kurallar tüm bilgisayar programlarında geçerlidir.

Python'da veri tipi dinamiktir. ***Yani tamsayı değeri verilmiş bir değişken programın işleyişinde örneğin ondalıklı bir sayı olarak dinamik olarak güncellenebilir.***

Not:Bölme işlemi sonucunda Python'un özelliği olarak sonuçlar float yani ondalıklı olarak görüntülenir. Bu özellik Python 3 ile gelen ve sonuçların daha kesin gösterilmesi için konulmuş bir özelliktir.

Python' da değişken isminde büyük küçük harf duyarlılığı bulunmaktadır. Yani x ve X farklı değişken isimleri olarak kullanılabilir.

Python' da diğer dillerde olmayan güzel bir özelliklerden birisi de iki değişkenin değerinin birbirleri ile değiştirmenin pratikliğidir. Bu işlem diğer programlama dillerinde biraz uzun bir code yazmayı gerektirir. Fakat Python' da bu işlem

```
a,b=b,a
```

işlemi ile kolayca yapılır.

Bir değişkenin değerini bir artırmak için (ki bu işlem döngülerde çok önemlidir) **+1** yapılabilir. Ancak Python' da daha kolay bir yöntem **+=1** tanımıdır. Bu tanım ile değişkenin değeri bir artırılır ve yeni değişken değeri olarak atanır. Sayı iki artırılmak istenirse de +=2 yazılabilir vb.

Bir değişkenin bir sayı ile çarpılıp çarpımın yeni değişken değeri atanması için de *=sayı komutu kullanılabilir. Örneğin x=5 ise ve yeni değeri ***x=x*4*** ise bu, ***x*=4*** olarak yazılabilir.

Ya da bir x değişkenin kendi değerinden örneğin 3 sayısı çıkarılacaksa, ***x-=3*** komut satırı kullanılabilir.

Yorum satırları

işaretinden sonra (Alt Gr+3) tek satırlık yorum satırı eklenebilir.

 # tekli yorum satırı

Çoklu yorum satırları için de

“““

Çoklu yorum satırı

”””

özelliği kullanılır. Yorum satırları Python tarafından işleme alınmaz.

Matematik Operatörleri

// sembolü tamsayı bölmesidir.

% sembolü de bölümden kalanı göstermektedir.

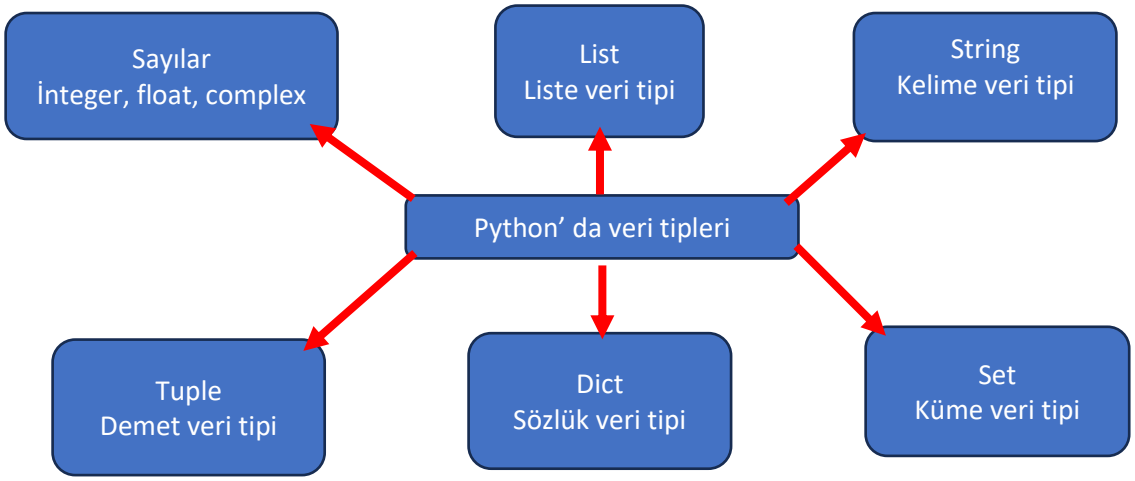
Üs bulma işlemi için iki tane ** işareti peş peşe kullanılır.

Köklü sayılar için de örneğin karekök için **(0.5) yazılarak sayının karekökü belirlenir.

Operatörleri birlikte kullanma ve işlem sırası

Matematiksel operatörler birlikte kullanılırken matematikten bilinen özellikler aynen geçerlidir.

- Her zaman için parantez öncelikli işlem sıradır.
- Çarpma ve bölme toplama ve çıkarma işlemine göre önce yapılır.
- İşlemler soldan sağa değerlendirilir.



Karakter Dizileri (String'ler)

Stringler, karakter dizileri yani harflerde oluşan değişkenlerdir. Örneğin "Can" string değişkeni C, a ve n harflerinden oluşan bir dizidir.

String oluşturma

String oluşturmak için Python' da kullanılabilen üç yöntem bulunmaktadır. İlk yöntem tek tırnak 'Gökhan' arasında stringin tanımlanabilmesidir. Bir tane çift tırnak arasında ya da iç tane çift tırnak arasında yazılan karakterler de bir string oluşturur.

'Kahve'

"Kahve"

""""Kahve""""

String oluşturma da hangi yöntem kullanılacaksa, kurala sadık kalınmalıdır. Yani tek tırnak açıp çift tırnak ile kapatılmamalıdır, vb. Bu nedenle örneğin

'Bugün kahve içmedim'

ifadesinde, ikinci tırnak stringi tamamladığından komut satırı hata verecektir. Bu nedenle bu tip durumlarda örneğin

"Bugün kahve içmedim"

Şeklinde çift tırnak ile string değişkeni tanımlanmalıdır. Bir diğer yöntem de

'Kemal\'in bugün dersi var'

komut yapısını kullanmaktır. İsimden sonra gelen \ işareti stringin henüz bitmediğini dolayısıyla ikinci tırnak işaretinin etkisiz olduğunu belirtir.

Python'da kaçış dizileri bulunmaktadır. Az önce değinilen \ (Alt Gr + ?) kaçış dizisidir ve Python'a "bu, bu değil" ya da "bu, bu anlama gelmiyor" demektir. Yukarıdaki örnek üzerinden \' durumu, Python'a, bu bitiş tırnağı olmadığını söylemektedir.

String değişkeni tanımlamak için x="merhaba" şeklinde bir komut satırı oluşturulabilir.

String indeksleme ve parçalama

Stringler bir kelime dizisi olduğundan bu dizinin içinden ihtiyaca göre istenilen karakter çekilip alınabilir.

u = Disk olarak tanımlanan string bir kelime dizesidir ve dize 0 indisinden başlar. Yani dizinin 0. elemanı, D, 1. elemanı i, 2. elemanı s ve üçüncü elemanı da k' dir. Dizide 4. eleman olmadığından u[4] komutu hata vermiştir. Dizinin elemanları çağrılırken, eleman numarası **köşeli parantez** ile çağırılır.

Stringler, 0' dan başlayarak 0, 1, 2, ... şeklinde indekslendiği gibi, sondan da -1 ile başlayarak indekslenir.

Uzun bir string yapısının sadece belirli bir parçası alınmak istenirse aşağıdaki komut yapısı kullanılır:

[başlangıç indeksi : bitiş endeksi : atlama değeri]

Not: Bitiş indeksi, diziye dahil değildir.

Q = "String parçalıyoruz"

Q[0:3]

0-S
1-t
2-r ‘dir ve 3. Karakter dahil değildir. Sonuç **Str** ’ dir.

Örneğin Q[:14] komutunda, başlangıç indeksi verilmemiştir. Program en baştan başlayarak yani 0. indeksten başlayarak 14. indeksi dahil etmeden stringi parçalıyor. Bunun aksine, bitiş indeksi verilmezse de, başlangıç indeksinden başlayarak stringin son karakterine kadar stringi parçalanabilir.

Q[:] komut satırı da “tüm stringi al” anlamına gelir.

İndeksleme sondan da başlayabiliyordu, -1' den başlayarak. Dolayısıyla yukarıdaki komut satırı -1. indeksi yani son karakteri almaz.

Q[:-1]

Komut satırının sonucu 'String parçalıyoru' olacaktır ve dikkat edilirse -1. yani sonuncu karakteri almamıştır.

Q[::2]

komut satırı, iki iki atlayarak stringi parçala anlamına gelecektir.

Q[::-1]

komut satırı, stringi tersten yaz anlamına gelir. Bu özellik Python' un güzel farklılıklarından birisidir.

Bir stringin uzunluğunu nasıl belirleyebiliriz? Bu amaçla **len()** fonksiyonu kullanılır. (len=length kelimesinin kısaltmasıdır.)

Stringlerin önemli özelliklerinden birisi, bir kere tanımlandıktan sonra bir daha kolayca değiştirilemez. Örneğin

S = “Mertaba”

stringinde 0’ dan başlayarak 3. Karakter yanlış olmuştur. Bu karakter düzeltilmek istenirse

S[3] = h

komut satırı yazılırsa, program hata verecektir. Dolayısıyla stringlerde değişiklik yapmak kolay bir işlem değildir.

Python'daki stringler birbiriyle toplanabilme veya çarpma işlemi özelliğine sahiptir.

Veri Tipi Dönüşümleri

Tam sayının ondalıklı sayıya dönüştürülmesi: **float()** fonksiyonu, tam sayısı ondalıklı sayıya dönüştürür.

```
a= 10
float(a)
10.0
```

olarak, 10 tam sayısı, 10.0 ondalıklı sayısına dönüştürülmüştür. Ondalıklı sayının tam kısmı int() fonksiyonu ile çekilebilir.

Sayıların Stringe dönüştürülmesi

Bu işlem str() fonksiyonu ile sağlanır.

Stringi tam sayıya dönüştürme

int() fonksiyonu kullanılarak string değişkeni tam sayıya dönüştürülür.

Print Fonksiyonu ve Çıkış Formatlama

print() fonksiyonu, ekrana yazı yazdırmak için kullandığımız bir fonksiyondur. Jupyter notebook' un özelliği olarak hücreler çalıştırıldığında otomatik olarak ekrana sonuç yazdırılır fakat bu etkileşimli kabuk olan Jupyter notebook' un özelliğidir ve Phycharm gibi IDE' ler ile çalışıldığında ciddi ve büyük uygulamalarda print() fonksiyonu gereklidir.

print() fonksiyonun içerisinde birden fazla niceliğin aynı satırsa gösterilmesi istenirse nicelikler virgül ile ayrılmalıdır.

```
print(3.14, "çay olsa da içsek", 10**3)
3.14 çay olsa da içsek 1000
```

Dikkat edilirse virgülle ayrılan nicelikler ekrana aralarında boşluk bırakılarak yazdırılmıştır.

\n ve \ t

Stringlerde kullanılan iki özel karakter olan \n ve \t karakterlerine burada print() fonksiyonunda değinilebilir. Bir string değişkeni \n ile yazılırsa \n den sonraki satır bir alt satıra yazdırılır.

```
print("Vatanını\nen\nçok\nseven\ngörevini\nen\niyi\nyapandır\nKemal Atatürk")
Vatanını
en
çok
seven
görevini
en
iyi
yapandır
Kemal Atatürk

print("Vatanını en çok seven\ngörevini en iyi yapandır.\nKemal Atatürk")
Vatanını en çok seven
görevini en iyi yapandır.
Kemal Atatürk
```

\t nicelikler arasında boşluk bırakmak için kullanılır. \t yazıldığında bir TAB kadar boşluk oluşturur.

```
print("Gökhan\tTüreci")
Gökhan        Türeci


print("Gökhan\t\tTüreci")
Gökhan                  Türeci
```

type() fonksiyonu

Bu fonksiyon, içine yazılan niceliğin tipini belirtiyor. Bu nedenle de oldukça faydalı bir fonksiyondur. Bu fonksiyon, içine yazılan niceliğin tipini belirtiyor. Bu nedenle de oldukça faydalı bir fonksiyondur.

```
type(69)
int

type(3.14)
Float

type("yapay zeka")
str
```

print() fonksiyonunun özellikleri

sep parametresi

sep = "istenen karakter" parametresi, ard arada yazdırılmak istenen niceliklerin arasına "istenen karakter" olarak belirtilen karakteri yerleştirir.

```
print(3.14,56,"GS",792.14, sep = "*")
3.14*56*GS*792.14

print(3.14,56,"GS",792.14, sep = "ooo")
3.14ooo56oooGSooo792.14


print("Gökhan","Türeci", sep = "/")
Gökhan/Türeci

print("Gökhan","Türeci", sep = "\n")
Gökhan
Türeci

print("19","05","1919", sep = "/")
19/05/1919
```

Yıldızlı Parametreler

```
print("Python")
Python

print(*"Python")
P y t h o n


print("T","B","M","M", sep =".")
T.B.M.M


print(*"TBMM", sep =".")
T.B.M.M
```

Çıkışı Formatlama

```
# Belirli sayıdaki süslü parantez {} kullanılarak aynı sayıdaki nicelik süslü parantezlerle yer değiştirilerek yazdır
ılır

"{}, {}, {}".format(3.14,9,1001)
'3.14, 9, 1001'

"{}{}{}".format(3.14,9,1001)
'3.1491001'

"{} {} {}".format(3.14,9,1001)
'3.14 9 1001'

x=19
y=409
print("{} + {} 'in toplamı {}'dir.".format(x,y,(x+y)))
19 + 409 'in toplamı 428'dir.

# Süslü parantez içine sayılar yazılırsa, bu indeks numaralarına karşı gelir
# format içindeki dizi, indeks numaralarına göre parantezlerle yer değişir.

"{2} {1} {0} {3}".format("Çalış","Öğün","Türk","Güven")
'Türk Öğün Çalış Güven'

# Ssülü parantezin aşağıdaki gibi kullanımında, ondalıklı sayıların
# belirtilen basamak sayısıyla yazılması sağlanır

"{:.2f} {:.4f} {:.8f}".format(3.14159265358979,3.14159265358979,3.14159265358979,3.14159265358979)
'3.14 3.1416 3.14159265'
```

Print() fonksiyonun daha birçok özelliği bulunmaktadır. Bu detaylı kullanım için https://pyformat.info/ adresinden yadım alınabilir.