ITU ACM Student Chapter Course Program

Python Course

Week 1

Instructor
Hüseyin Averbek

Assistants
Serra Bozkurt
Gökalp Akartepe

Week1

Print() Fonksiyonu

Bir programın en temel fonksiyonlarından birisi ekrana çıktı vermektir. Python'da ekrana output bastırmak için print() fonksiyonu kullanılır:

print() fonksiyonu aracılığıyla ekrana farklı veri tipleri bastırılabilir. Bunlardan en sık kullanılanı stringlerdir. print() fonksiyonu ile ekrana string bastırmak için bastırılacak stringin " " işaretleri arasına yazılması gerekmektedir.

```
print("Hello World!")
```

```
Hello World!
```

print() fonksiyonu ile aynı zamanda matematiksel işlemlerin sonucu da bastırılabilir:

```
print(3+5)
```

```
8
```

Yorum Satırı

Programlamada okunabilirlik kadar yorum satırları da çok önemli bir husustur. Yorum satırları, yazılan kodun bir başkası tarafından anlaşılmasına yardımcı olan açıklamalardır. Hatta bazen kodun yazarının bile kafa karışıklığını gidermeye yarayabilir.

Nasıl Kullanılır

Python "#" işaretini gördüğü satırlarda "#" işaretinden sonraki kısmı yorum satırı olarak algılar ve işlemez.

```
#Burda "Hello World!" yazısını ekrana yazdırdık.
print("Hello World!")
```

```
Hello World!
```

```
print(3+5) # Burda 3 ve 5'in toplamını ekrana yazdırdık.
```

```
8
```

Değişkenler

Programlama yaparken her zaman verilerin kesin değerlerini kullanarak işlemler yapmayız. Bazı veriler değişkendir, mesela hava sıcaklığı. Örneğin Celcius biriminden alınan hava sıcaklığını Kelvin birimine dönüştüren bir program yazıyorsak burada değişkenleri kullanırız. Çünkü hava sıcaklığı değişkendir ve bizim değişkenler üzerinden yazacağımız kod da bu değişkene göre sonuçlar alacaktır.

Değişken Tanımlama Değişken tanımlarken dikkat etmemiz gereken birtakım kurallar var. Öncelikle basit bir değişken tanımlama örneği gösterelim.

```
a = 15 # Burda a değişkenine 15 değerini atadık.
```

```
print(a) # Burda a değerini yazdırdık.
```

```
15
```

Değişkenin adı = Değişkenin alacağı değer

Değişken tanımlamayı bu şekilde özetleyebiliriz. Bu aşamada değişkenimize vereceğimiz ad konusunda dikkatli ve açık olmalıyız. Bu konuda da bazı kurallar mevcut:

1) Değişken adının içinde boşluk bulunamaz.

```
hava sıcaklığı = 34

print(hava sıcaklığı)
```

```
  File "<ipython-input-8-ad46b0ee513e>", line 1
    hava sıcaklığı = 34
         ^
SyntaxError: invalid syntax
```

Bu şekilde adlandırılmış bir değişken çalışmayıp hata verecektir.

Birden fazla kelimeden oluşmasını istediğiniz değişken adlarını bitişik olacak biçimde yazmak durumundasınız.

```
havasıcaklığı = 34

print(havasıcaklığı)
```

```
34
```

Burda kelimeleri bitişik yazdığımızda oluşabilecek kargaşayı önlemek için iki farklı yönteme başvurabiliriz

```
havaSıcaklığı = 34 # Python'da büyük-küçük harf duyarlılığı vardır.

print(havaSıcaklığı)
```

```
34
```

```
hava_sıcaklığı = 34 # Kelimelerin arasına noktalama işareti koyularak␣
↪karışıklık azaltılabilir.

print(hava_sıcaklığı)
```

```
34
```

2) Değişken adı konarken anahtar kelimeler kullanamazsınız.

```
and = "ve" # Burda and adında bir değişkene "ve" string değerine atamaya␣
↪çalışıyoruz.

#Kodumuzu bu haliyle çalıştıralım.
```

```
  File "<ipython-input-12-4badb502b60f>", line 1
    and = "ve" # Burda and adında bir değişkene "ve" string değerine atamaya␣
↪çalışıyoruz.
    ^
SyntaxError: invalid syntax
```

Burda and Python programlama dilinin anahtar kelimelerinden biri olduğundan hata verdi.

3) Değişkenler adlarının içinde sayı bulunabilir fakat değişken adları sayı ile başlayamaz.

```
sayı1 = 18

print(sayı1)
```

```
18
```

```
1sayı = 18

print(sayı1)
```

```
  File "<ipython-input-14-6c8d18513032>", line 1
    1sayı = 18
     ^
SyntaxError: invalid syntax
```

Değişkenler tanımlanırken işlemler de yapılabilir. Mesela bir değişkene iki sayının toplamını atayabiliriz.

```
islem = 3+5
```

```
print(islem)
```

```
8
```

İstersek diğer değişkenler ile yaptığımız işlemlerin sonucunu da bir değişkene atayabiliriz.

```
a = 12
b = 13

c = a+b

print(c)
```

```
25
```

Hatta karışık işlemler bile yapabiliriz.

```
a = 4
b = 6

c = 3*(a+b) - a*b

print(c)
```

```
6
```

Değişkenlere stringler de atayabiliriz.

```
isim = "Hüseyin"

print(isim)
```

```
Hüseyin
```

Stringleri birleştirmek için de toplama işaretini kullanabiliriz.

```
isim = "Hüseyin"
soyisim = "Averbek"

kişi = isim + soyisim

print(kişi)
```

```
HüseyinAverbek
```

Burda "Hüseyin" stringinin sonunda ve "Averbek" stringinin başında boşluk bulunmadığı için ismi ve soyismi bitişik yazdırdı. Bu duruma karşılık da şöyle bir çözüm getirilebilir:

```
isim = "Hüseyin"
soyisim = "Averbek"
```

```
kişi = isim + " " + soyisim

print(kişi)
```

```
Hüseyin Averbek
```

Uyarı: Tırnak işareti kullanılmadan yazılan sayılar ile stringler toplama işaretiyle birleştirilemez!

```
isim = "Hüseyin"
yaş = 21

isim_yaş = isim + " " + yaş

print(isim_yaş)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-44cd13e20d0a> in <module>
      2 yaş = 21
      3
----> 4 isim_yaş = isim + " " + yaş
      5
      6 print(isim_yaş)

TypeError: can only concatenate str (not "int") to str
```

Yukarıdaki hatayı almamızın sebebi [isim_yaş = isim + " " + yaş] satırında bir stringle bir tam sayıyı toplamaya çalışmamız. Bu durumun önüne geçmek için tamsayı değerimiz tırnak içine alınabilir. Böylelikle 2 stringi birleştirebiliriz:

```
isim = "Hüseyin"
yaş = "21"

isim_yaş = isim + " " + yaş

print(isim_yaş)
```

```
Hüseyin 21
```

Veri Tipleri

Python'da farklı verileri tutmak için farklı veri tiplerinde değişkenler kullanılır. En sık kullanılanları harfleri tutmak için string (str), tamsayıları tutmak için integer (int), ve rasyonel sayıları tutmak için float'tır (float).

type() Fonksiyonu Python'da bir değişkenin veya ifadenin hangi veri tipinden olduğunu görmek için type() fonksiyonu kullanılır.

```
a = 3
print(type(a))
```

```
<class 'int'>
```

type() fonksiyonunun değeri değişkene atanmadan da print() fonksiyonu içinde kullanılabilir. Python matematikteki bileşke fonksiyon mantığıyla içten dışa fonksiyonların işlevini yerine getirir. g [f(x)] = (g o f) (x)

```
b = "merhaba"
print(type(b))
```

```
<class 'str'>
```

```
c = 12.5
print(type(c))
```

```
<class 'float'>
```

Input() Fonksiyonu

Bir programın en önemli fonksiyonlarından bir diğeri ise kullanıcıyla iletişime geçebilmesidir. Python'da kullanıcıdan değer girmesini talep etmek için input() fonksiyonu kullanılır.

input() fonksiyonu ile programa dışarıdan veri alınır. Alınan bu veri programda daha sonra kullanılmak üzere bir değişkene atanır. Input() fonksiyonuna parametre olarak girilen değer kullanıcıdan input almadan önce ekrana bastırılır:

```
n = input("Lütfen bir sayı giriniz:")
print(n)
```

```
Lütfen bir sayı giriniz:10
10
```

```
n = input("Lütfen bir string giriniz:")
print(n)
```

```
Lütfen bir string giriniz:ITU ACM Python Kursu
ITU ACM Python Kursu
```

Input fonksiyonuna herhangi bir parametre vermeden de çalıştırabiliriz. Bu durumda kullanıcının ekranın bir bilgilendirme mesajı belirmeyecektir:

```
girilenDeger = input()
print(girilenDeger)
```

```
15
15
```

```
girilenDeger = input()
print(type(girilenDeger))
```

```
15
<class 'str'>
```

Burada görüldüğü gibi input fonksiyonu aldığı inputları her zaman string türünde alır. Yukarıdaki örnekte de görüldüğü gibi girdiğimiz değer bir tamsayı olan 5 olmasına rağmen program bu değerin tipini string olarak algıladı. Tamsayıları, rasyonel sayıları ve stringleri birbirinden ayırmanın yolu ise type casting yöntemidir.

Type Casting

Python'da input halinde gelen değerleri farklı veri tiplerine çevirmek için kullanılır. int(), float(), ve str(); input() fonksiyonundan gelen veriyi istenen veri tipine çevirmek için kullanılır. Type casting yapmadan aldığımız inputları stringten istediğimiz veri tipine dönüştüremeyiz ve bu durumda istenilen işlemleri yaparken hata alabiliriz.

```
n = int(input())
print(type(n))
```

```
15
<class 'int'>
```

Casting yapmadığımız takdirde karşılaşabileceğimiz sorunlara örnek vermek gerekirse:

```
a = input("Birinci sayı:")
b = input("İkinci sayı:")

toplam = a + b
print(toplam)
```

```
Birinci sayı:3
İkinci sayı:5
35
```

Yukarıda da görüldüğü gibi toplama işlemi yapmak istediğimiz halde aldığımız inputları int (tam sayı) veri tipine dönüştürmediğimiz için string birleştirme işlemi gerçekleştirdik. Bu durumun önüne geçmek için önce aldığımız inputları int veri tipine dönüştürmemiz gerekir:

```
a = int(input("Birinci sayı:"))
b = int(input("İkinci sayı:"))

toplam = a + b
print(toplam)
```

```
Birinci sayı:3
İkinci sayı:5
8
```

Burada dikkat edilmesi gereken önemli nokta veri tiplerini karıştırmamaktır. Aksi takdirde hatayla karşılaşabiliriz:

```
n = int(input())
print(n)
```

```
Python3
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-36-93ddc6c45b1a> in <module>()
----> 1 n = int(input())
      2 print(n)

ValueError: invalid literal for int() with base 10: 'Python3'
```

Max-Min Functions

max() fonksiyonu, en yüksek değere sahip öğeyi döndürür.

Değerler string ise alfabetik sıralamaya göre en sonda olan öğeyi döndürür.

```
print(max(3,5))
```

```
5
```

```
maksimum = max(3,5)

print(maksimum)
```

```
5
```

```
print(max("Adana","Mersin"))
```

```
Mersin
```

max() ve min() fonksiyonlarına istenildiği kadar öğe eklenebilir:

```
maksimum = max(6,2,3,5,6,7,9,1,4,2,8)

print(maksimum)
```

```
9
```

min() fonksiyonu, en düşük değere sahip öğeyi döndürür.

Değerler string ise alfabetik sıralamaya göre en önce olan öğeyi döndürür.

```
print(min(3,5))
```

```
3
```

```
minimum = max(3,5)

print(minimum)
```

5

```
print(min("Adana","Mersin"))
```

Adana

```
minimum = min(6,2,3,5,6,7,9,1,4,2,8)

print(minimum)
```

1