PYTHON 101

3. GÜN

BENGÜ YURDAKUL

9 Mart 2022

techcareer.net

Google Developer Students Club
Galatasaray University

GALATASARAY ÜNİVERSİTESİ

GİRDİ (INPUT) ALMA

- Bu noktada artık her şeyi kendimiz yazmayı bırakıyor ve kullanıcıdan veri istemeye başlıyoruz. Input yardımıyla klavyeden/kullanıcıdan bilgi/veri isteyip üstünde işlemler yapacağız.

- Programlama dillerinin olmazsa olmazı olan input fonksiyonu Python'da **input( )** şeklinde yazılır ve her zaman aldığı veriyi **string** tipinden alır.

input( )

- Bu nedenle **sayısal işlemler yaparken** dönüştürmeye ve dönüştüreceğimiz veri tipine dikkat etmeliyiz.
- Burada kullanıcıdan isteklerimiz doğrultusunda bir girdi yapmasını isteyeceğiz ve o girdiyi bir değişkene atayıp kullanacağız.

`a = input( )` dediğimizde program, kullanıcıdan bir veri girmesini isteyecektir ve girilen veriyi a değişkenine atayacaktır.

Örn:

```
a = input()

print(“Merhaba”, a)
```

Çalıştırdıktan sonra ekrana ne yazarsak program ekrana Merhaba yazısının yanında yazdığımız şeyi basacaktır. Mesela girdi olarak Dünya yazsaydım ekranda `Merhaba Dünya` yazısı çıkardı.

input( )

Aynı şekilde şu kodu da deneyebiliriz:

```
a = input("İsminizi giriniz : ")

print(“Merhaba”, a)
```

Bu örneği kendi komut satırımıza yazalım ve çalıştıralım. Ekranda,

```
İsminizi giriniz :
```

yazısının çıktığını göreceğiz. Buraya ismimizi girdikten sonra program bize `Merhaba <isim>` çıktısını verecektir. Yani Bengü yazmış olsaydım çıktı `Merhaba Bengü` şeklinde olacaktı.

Yani input( ) fonksiyonunun içine tırnak işareti ile (“ “) bir şey yazdığınızda program bunu olduğu gibi basacak ve sonunda sizden bir girdi isteyecektir. Ardından da kodunuzun devamına göre işlemleri yapacaktır.

input( ) TİP DÖNÜŞÜMLERİ

- Yukarıda verdiğim örnekte input istendiğinde bir **sayı** girdiyseniz işlemleri yapamadığını ve hata verdiğini görmüşsünüzdür. Bunun sebebi en başta belirttiğim gibi bu fonksiyonun her şeyi string (karakter dizisi) olarak algılaması. Bu nedenden dolayı başka veri tipine ihtiyacımız olduğunda tip dönüşümü yapmamız gerekir.

- Tip dönüşümleri için yapmamız gereken şey dönüştürmek istediğimiz veri tipini yazıp parantez içinde input fonksiyonunu yazmak.

  Örn:

```
int(input( ))

float(input( ))
```

Kullanıcıdan alınan string tipindeki veri tipini istenen(başına yazılan) veri tipine dönüştürecektir, eğer başına int yazarsak alınan veri int tipinde olacaktır.

input( ) TİP DÖNÜŞÜMLERİ

```
a = int(input( ))

b = int(input( ))

print(a * b)
```

kodunu çalıştıralım. Kullanıcıdan iki defa veri girişi isteyecektir. İlk istediği girdiyi int (integer/sayı) tipine dönüştürerek a değişkenine, ikinci istediğini de integer tipine dönüştürerek b değişkenine atayacaktır. Ve ardından a ile b'nin içindeki değerleri çarpacak, print( ) fonksiyonu ile ekrana basacaktır. İlk input isteyişinde 5, ikincisine 6 girersek sonuç 30 olarak çıkacaktır.

Bu kodu sadece input() yazarak deneseydik hata verecekti çünkü girdiğimiz sayıları string (karakter dizisi) tipinde algılayacaktı ve bildiğiniz üzere string veri tiplerinde çarpma işlemi yapılamaz.

input( ) TİP DÖNÜŞÜMLERİ

Aynı işlemler float tipinde de geçerlidir.

```
a = float(input("Dikdörtgenin ilk kenarı : "))
b = float(input("Dikdörtgenin ikinci kenarı : "))
print("Dikdörtgenin alanı : ", a * b)
```

ilk input'a 4.5 ikincisine de 6.3 girersem `28.349999999999999` sonucunu elde ederim.

input( ) TİP DÖNÜŞÜMLERİ

O halde siz de bu kodu komut satırınıza yazarak deneyebilirsiniz :

```
a = int(input("Birinci Sayı:"))

b = int(input("İkinci Sayı:"))

c = int(input("Üçüncü Sayı:"))

print("Toplam:", a+b+c)
```

DÖNGÜLER

- Program yazarken aynı işlemi birden fazla kez tekrarlama ihtiyacı hissedebiliriz.
- Bir koşula bağlı olan tekrarlara döngü denir.
- Python’da for ve while olmak üzere iki farklı temel döngü vardır.

while DÖNGÜSÜ

- Bu döngü yanında verilen şart sağlandığı sürece döner ve içindeki işlemleri yapar, şart sağlanmıyorsa döngüden çıkar.

- Dönebilmesi için değişken(ler)e ihtiyaç duyar. Ancak değişkene bir değer atanmış olmalıdır.

- Python kendisi bu değişkene 0 veya başka değer vermeyecektir. İlk değeri olmayan bir değişken kullanırsanız hatayla karşılaşırsınız. While döngüsünde her tekrarda değişkeni değiştirmemiz/arttırmamız gerekiyor.

while DÖNGÜSÜ

Yazım:

```
while şartİfadesi :

    yapılacak işlemler
```

Örn:

```
i = 5

while i<10 :

    print(i)

    i = i + 1 //her dönüşte i'nin değerine 1 ekleyip yeniden i'ye atıyoruz. Böylece
    her tekrarda i'nin değeri bir artmış/değişmiş oluyor. Bu da while döngüsünün
    başarılı bir şekilde çalışmasını sağlıyor.
```

while DÖNGÜSÜ

- i değişkenine 5 değerini atadık ardından while döngüsünün şartını kontrol ettik.
- Eğer şart doğru ise döngüye girecek, değil ise döngüden çıkacaktır.
- 5, 10'dan küçük olup şartı sağladığı için döngüye girip ekrana i'nin o anki değerini (yani 5) basıyor.
- Sonra i'nin değerini bir arttırıp tekrar while'ın başına gelip şartı kontrol ediyor.
- Bu şekilde şart sağlanmayıncaya kadar dönmeye ve içindeki işlemi yapmaya devam ediyor.

Sonuç olarak çıktımız şu olacaktır :

```
5
6
7
8
9
```

while DÖNGÜSÜ

Eğer i'nin değerini birer arttırma işlemimizi (i=i+1) yapmamış olsaydık program sonsuz döngüye girerdi. Bu nedenle değişkenin üzerindeki istediğiniz artış/azalış miktarını döngünün içinde işlem olarak yapmamız gerekiyor.

Eğer bunu yapmazsak ne olacağını daha iyi görmek için şu kodu çalıştırabilirsiniz :

```
sayi = 0

while sayi == 0:

    print("bilgisayar çıldırdı!")
```

while DÖNGÜSÜ

`sayi` değeriyle oynadığımız bir işlem yapmadığı için sayi değişkeninin içindeki değer hep 0 olarak kalacak ve değişmeyecektir bundan dolayı döngünün şartı olan `sayi==0` şartı sonsuza kadar sağlanacak ve sonsuz döngüye girilecektir. Bunu düzeltmek için `kodla` azcık oynamamız yeterlidir :

```
sayi = 0

while sayi == 0 :

    print("bilgisayar çıldırdı!")

    sayi=sayi+1
```

işte problemi düzelttik !

for DÖNGÜSÜ

- Bir çok farklı yazım şekli olan for döngüsünün hangi koşullar altında ve kaç defa döneceğini biz ayarlıyoruz.

- İlk göz atacağımız for döngüsü için bir değişken kullanmamız gerekiyor ve bu değişken genelde « i » harfi oluyor ama siz başka değişkenler de kullanabilirsiniz.

- For döngüsünde, while'dan farklı olarak her tekrarda Python bizim için i'nin değerini bir arttıryor. Böylece döngünün dönmesi için i'nin değerinin her tekrarda değişmesi şartı ortadan kalkıyor.

for DÖNGÜSÜ

Yazımı şu şekilde olabilir :

```
for i in range (sayıdeğeri) :

    yapılacak işlemler
```

Herhangi bir for döngüsünün yazımına for ile başlanır ve yanına bir değişken konur. Bu değişkenin işlevi, döngünün istendiği kadar dönmesini sağlamaktır. Fark ettiyseniz burada « i » değişkenine bir ilk değer atamadık ve tipini de belirtmedik. Bu noktada Python'ın bize sağladığı bir kolaylıktan faydalanıyoruz. Eğer ilk değeri yoksa Python bu değişkeni 0 olarak düşünecek ve saymaya/dönmeye 0'dan başlayacaktır. Bu yazımda `range` sırayla gitmek anlamına gelir. Yani sona konan sayıya kadar döngü sırayla dönecektir.

(DİKKAT: `sayıdeğeri` dahil değil! Yani o sayıya kadar dönecek.)

for DÖNGÜSÜ

- for döngüsündeki değişkenimiz (bu örnekte « i » oluyor) bir sayaç görevi görür.
- Öncelikle bir değeri olmadığı için i değişkenimiz 0 kabul ediliyor ve döngü çalışmaya başlıyor.
- Her turda i'nin değeri bir (1) artacak ve her turda ekrana i'nin değerini basacak.
- Ancak dikkat edilmesi gereken nokta şu ki, i'nin değeri 10'a kadar artacak. Bu durumda 10 sayısı dahil değil.

```
for i in range (10) :
   print(i)
```

Çıktı:

```
0
1
2
3
4
5
6
7
8
9
```

for DÖNGÜSÜ

Ayrıca döngülerin içerisinde yapılacak işlemleri Python'un anlayabilmesi için girintilemeye dikkat edilmelidir. Eğer herhangi bir girinti olmadan programı yazarsanız döngü hiçbir şey yapmadan sona erecek ve aslında döngünün içinde gerçekleştirilmesini istediğimiz işlemler sadece bir kez yapılacaktır. Yukarıda yazdığımız kodu girintileme olmadan şu şekilde tekrar yazıp çalıştıralım:

```
for i in range (10) :
print(i)
```

Bu durumda hata verecek ve girintileme yapmadınız diye sizi uyaracaktır.

for DÖNGÜSÜ

Başka bir örnek deneyelim. Bu örnekte de 0 dışında belli bir sayı aralığında sırayla ilerlemesini isteyelim. O halde yazımı şu şekilde olacaktır ;

```
for i in range (başlangıçsayısı,ekadargideceğisayı) :

    yapılacak işlemler
```

`başlangıçsayısı` kısmında saymaya/dönmeye hangi sayıdan başlamasını istiyorsak onu belirtiyoruz, `ekadargideceğisayı` kısmında ise hangi sayıya kadar dönmesini/saymasını istediğimizi belirtiyoruz.

for DÖNGÜSÜ

Hemen basit bir örneğe göz atalım :

```
for i in range (4,8) :

    print(i)
```

« i »'nin değeri verilmediği için Python 0 kabul eder demiştik, bu tarz for döngülerinde ise i ilk değer olarak 0'ı değil `başlangıçsayısı` olarak belirttiğimiz değeri alıyor yani bu örnekte 4'ten başlayacak ve 8'e kadar gidecektir. Ancak yine 8 dahil değildir.

Sonuç olarak alacağımız çıktı şu şekilde olacak :

```
4
5
6
7
```

Konuyu pekiştirmek adına yeni öğrendiğimiz input() fonksiyonunu kullanarak bir örnek yapalım :

```
birincisayi = int(input(“Bir sayı giriniz : ”)
ikincisayi = int(input(“Bir sayı giriniz : ”)
for birincisayi in range ikincisayi :
    print(birincisayi)
    print(ikincisayi)
```

for DÖNGÜSÜ

Örnekleri gördüğümüze göre biraz daha detayına inebiliriz. range fonksiyonunu biraz daha tanıyalım. Türkçede karşılığı “aralık” olan bu fonksiyon sayı aralıklarını belirtir. Kullanımı şu şekildedir:

```
range(başlangıç değeri , bitiş değeri, değişim miktarı)
```

`range(10)` : sıfır ile 9 aralığındaki sayıları belirtir. Başlangıç değerini belirtmediğimiz için varsayılan olarak 0, değişim miktarını belirtmediğimiz için ise varsayılan olarak 1 değerini alır. Ve dikkat etmemiz gereken başka bir husus ise bitiş değerini dahil etmemesidir.

for DÖNGÜSÜ

`range(2,20,3)` : 2 ile 19 arasında üçer üçer artarak gider.

```
for i in range (2,20,3) :
    print(i)
```

Alınacak çıktı :

```
2
5
8
11
14
17
```

for DÖNGÜSÜ

`range(20,2,-2)` : 20 ile 2 arasında ikişer ikişer azalarak gider.

```
for i in range (20,2,-2):
    print(i)
```

Alınacak çıktı :

```
20
18
16
14
12
10
8
6
4
```

for DÖNGÜSÜ

Şimdi ise `in`'e bakalım. Türkçe deki karşılığı "içinde" olan `in`, istenilen değer ya da aralık içinde olup olmadığını sorgulama yapmamızı sağlayan araçtır.

`for i in range (20)` : burada sayı değişkeni 0 ile 19 arasındaki(içindeki) bütün değerleri alır.

Sadece tam sayı (integer) değerlerle değil aynı zamanda string (karakter dizileri) ile de for döngüsünü kullanabiliyoruz.

Örn:

```
for i in "Python" :
    print(i)
```

for DÖNGÜSÜ

Yine aynı şekilde ilk değeri olmayan i'yi 0 olarak düşünüp yanında yazılan karakter dizisi boyunca sırayla ilerleyerek her harfi (her döngüde) ekrana basacağız. Ancak dikkat edersiniz ki bu sefer sayı aralığı verilmediği için range fonksiyonu kullanılmıyor. Aynı sebepten dolayı bir eksiğine kadar gitme/dönme durumu da söz konusu değildir. Yanda tırnak işareti arasında verilen karakter dizisinin sonuna gelene kadar dönecek ve her aşamada içindeki işlemleri yapacaktır.

Bu durumda çıktımız şu şekilde olacaktır :

```
P
y
t
h
o
n
```

SIRA SİZDE ☺

Çalışma örneklerini verdiğim programların kodlarını yazmanızı bekliyorum. Kendinizi deneyin, denedikten sonra arka sayfalarda bulunan çözümlere bakabilirsiniz.

SIRA SİZDE #1

Kullanıcıdan girilen iki sayıdan ilkinin, ikincisinin katı olup olmadığını söyleyen bir program yazınız.

Örn:

```
a sayısı giriniz : 22

b sayısı giriniz : 11
```

OUTPUT:

```
a sayısı b sayısının katıdır.
```

SIRA SİZDE #2

Kullanıcının girdiği « n » sayısından 1’e kadar olan bütün sayıları toplayıp ekrana yazan program yazınız.

Örn:

```
Bir n sayısı giriniz : 6
```

OUTPUT:

```
20
```

SIRA SİZDE #3

Kenarı girilen üçgenin, eşkenar üçgen olup olmadığını söyleyen program yazınız.

Örn:

```
Birinci kenarı giriniz : 3

İkinci kenarı giriniz : 5

Üçüncü kenarı giriniz : 4
```

OUTPUT:

```
Eşkenar üçgen değildir.
```

SIRA SİZDE #3 - Versiyon 2

Daha sonra bu programı üçgenin çeşidini de yazacak şekilde geliştiriniz.

Örn:

```
Birinci kenarı giriniz : 4

İkinci kenarı giriniz : 4

Üçüncü kenarı giriniz : 6
```

OUTPUT:

```
Bu bir ikizkenar üçgendir.
```

SIRA SİZDE #4

Kullanıcıdan 4 farklı sayı girmesini isteyen ve girilen sayıların arasından en büyük olanı ekrana basan program yazınız.

Örn:

```
Bir sayı giriniz : 23

Bir sayı giriniz : 8

Bir sayı giriniz : 39

Bir sayı giriniz : 11
```

OUTPUT:

```
En büyük sayı : 39
```

SIRA SİZDE #5

Kullanıcıdan iki basamaklı bir sayı girmesini isteyen ve girilen sayının basamaklarının karelerinin toplamını ekrana basan program yazınız.

Örn:

```
İki basamaklı bir sayı giriniz : 54
```

OUTPUT:

```
Girdiğiniz sayının basamaklarının kareleri toplamı : 29
```

SIRA SİZDE #6

Kullanıcıdan en ve boy değerlerini alan ve her satırda sırasıyla *, o, - olacak şekilde dikdörtgen basan program yazınız.

Örn:

```
Enini giriniz : 4

Boyunu giriniz : 5
```

OUTPUT:

```
****

oooo

----

****

Oooo
```

SIRA SİZDE #1 - Çözüm

```
a = int(input("a sayısı giriniz : "))

b = int(input("b sayısı giriniz : "))


if (a%b==0)

  print("a sayısı b sayısının katıdır. ")

else

  print("a sayısı b sayısının katı değildir. ")
```

SIRA SİZDE #2 - Çözüm

```
n = int(input("Bir n sayısı giriniz : "))

toplam = 0

for n in range (n,1,-1) :

  toplam = toplam + n

print(toplam)
```

SIRA SİZDE #3 - Çözüm

```
birincikenar = int(input("Birinci kenarı giriniz = "))
ikincikenar = int(input("İkinci kenarı giriniz = "))
ucuncukenar = int(input("Üçüncü kenarı giriniz = "))
if (birincikenar == ikincikenar) :
   if (ikincikenar == ucuncukenar) :
       print("Eşkenar üçgendir.")
   else :
        print("Eşkenar üçgen değildir.")
else :
        print("Eşkenar üçgen değildir.")
```

SIRA SİZDE #3 (Versiyon 2) - Çözüm

```
birincikenar = int(input("Birinci kenarı giriniz = "))

ikincikenar = int(input("İkinci kenarı giriniz = "))

ucuncukenar = int(input("Üçüncü kenarı giriniz = "))
```

Devamı

```
if (birincikenar == ikincikenar) :
      if (ikincikenar == ucuncukenar) :
            print("Eşkenar üçgendir.")
      else :
            print("İkizkenar üçgendir.")
elif (birincikenar == ucuncukenar) :
      print("İkizkenar üçgendir.")
elif (ikincikenar == ucuncukenar) :
      print("ikizkenar üçgendir.")
else :
      print("Çeşitkenar üçgendir.")
```

SIRA SİZDE #4 - Çözüm

```
sayi1 = int(input("Bir sayi giriniz : "))

sayi2 = int(input("Bir sayi giriniz : "))

sayi3 = int(input("Bir sayi giriniz : "))

sayi4 = int(input("Bir sayi giriniz : "))

maxsayi = sayi1
```

Devamı

```
if (sayi2>maxsayi):

    maxsayi = sayi2

if (sayi3>maxsayi):

    maxsayi = sayi3

if (sayi4>maxsayi):

    maxsayi = sayi4

print("En büyük sayi : ",maxsayi)
```

SIRA SİZDE #6 - Çözüm

```
a = int(input("İki basamaklı bir sayı giriniz :"))

b = a / 10      //onlar basamağını elde ediyoruz.

c = a % 10      //birler basamağını elde ediyoruz.

b = int(b)    //girilen sayı eğer 10'un katı değilse float(ondalıklı) bir sayı elde edeceğiz.
//Bunun önüne geçmek için integer (tam sayı) veri tipine çeviriyoruz.

d = b * b + c * c

print("sonuç = ",d)
```

SIRA SİZDE #4 - Çözüm

```
en = int(input("Eni ne olsun ? : "))

boy = int(input("Boyu ne olsun ? : "))

a = 0

b = 0
```

Devamı

```
for a in range (boy) : //boyu kadar dönecek

    if b % 3 == 0 : //hangi satırda olduğumuzu
buluyoruz

        print("*" * en) //eni kadar basacak

    elif b % 3 == 1:

        print("o" * en)

    else :

         print("-" * en)

    b = b + 1  //satırı bir arttırıyoruz
```

EKSTRA

Break ve Continue

- Döngüler, koşul sağlandığı sürece belli işlemleri yapmak için kullanılır ve bizi birçok satır yazmaktan kurtarır. Ancak bazen farklı bir koşulun gerçekleşmesi durumunda döngüyü sonlandırmak, ya da başa döndürmek (yinelemek) isteyebiliriz.

İşte bu anlarda yardımımıza Break ve Continue koşar.

BREAK

- Break ifadesi, içinde bulunduğu döngüyü sonlandırır.

```
for i in range (10):
   if(i == 5):
      break
print(i)
```

Output:

```
5
```

Burada 0'dan 10'a kadar dönen for döngüsünü görüyoruz. İçinde if koşullu ifadesi bulunuyor. i'nin değerinin 5'e eşit olup olmadığını kontrol ediyor. Eğer bu ifade doğru ise/koşul sağlanıyorsa, if'in içine giriyor ve break komutu ile döngüden çıkıyor.

Sonuç olarak döngü sadece 5 sayısına kadar dönebiliyor çünkü break ile döngüyü kırdık.

CONTINUE

- İçinde birden fazla işlem olan bir while döngüsü hayal edelim. Bu işlemlerden bazıları, eğer bir koşulun sağlanması durumunda çalışmasın. Böyle bir şeyi başarmak istiyorsak continue kullanıyoruz.

```
for i in "Merhaba Dunya":
	if(i == "a"):
		continue
	print(i)
```

CONTINUE

“Merhaba Dunya” yazısında harf harf dönen bir for döngüsü oluşturduk. if koşullu ifadesi ile o harfin “a” harfi olup olmadığını kontrol ediyoruz. Eğer “a” harfi ise continue yardımı ile alttaki işlemleri es geçip döngüye geri dönüyor ve bir harf ilerliyoruz.

Eğer “a” harfi değilse print yardımı ile ekrana basıyoruz.

Böylece Output: Merhb Duny oluyor.

ÇOK TEŞEKKÜR EDERİZ ☺

Daha fazla çalışabilmeniz için kaynaklar;

- Youtube kanalımız DSC Galatasaray'daki Python eğitim serimize göz atabilirsiniz.
- Döngüler (Bu slaytta detaylı yer almayan Break ve Continue 'a buradan bakabilirsiniz.)
- Sorularınız olması durumunda bizlere Instagram sayfamızdan ulaşabilirsiniz.

Hazırlayan: Bengü YURDAKUL

Bu eğitimde kullandığımız slaytların tasarımını yapan Selin Turan'a teşekkürler