2021

Yazılım Notları

PYTHON

TARIK MİRAÇ AYKAN

SAMSUN ARTIFICIAL INTELLIGANCE COMMUNITY

İçindekiler

Giriş

- ✓ Python, Guido van Rossum tarafından geliştirilmeye başlanmış yüksek seviyeli, yorumlanabilir, interaktif ve nesneye yönelik bir programlama dilidir.

Değişkenler

- ❖ Değişkenleri isimlendirme için kullanıyoruz. Bir ifadenin değişken sayılabilmesi için harfle başlaması yeterlidir, değişkenler rakamla başlayamaz.
- ❖ Başlıca 3 tip değişkenimiz var. Bunlar integer(int), stranger(str) ve float.
- ❖ İnteger veri tipi tam sayıları ifade eder.
- ❖ Float veri tipi kesirli sayıları ifade eder.
- ❖ Stranger veri tipi tüm karakterleri kapsar, kullanımı diğer iki değişkene göre farklılık gösterir.
- ❖ Integer ve float değişkenlerini tanımlarken değeri tırnak içerisinde yazmamıza gerek yoktur. Stranger veri tipinde ise değeri tırnak içinde yazmamız gereklidir.

```
A = 5
B = 9.8
C = "Tarık Aykan"
```

A integer, B float ve C stranger'dir.

```
D = "6"
```

D veri tipi integer olarak gözükse de stranger'dir çünkü tırnak içerisinde yazılmıştır.

Tırnak içerisinde yazılan değerleri stranger olarak kabul ederiz.

```
F = "  "
```

F değişkeninde tırnak içerisine boşluk bıraktık. Boşluk bize boş olarak gözükse de bilgisayar dilinde bir değere sahiptir ve bu yüzden F de bir stranger değişkendir.

```
H = 7
H = 10.5
```

Yukarıdaki kodda H değişkeni başta 7 olarak tanımlandı ancak alttaki satırda 10.5 olarak tekrar tamamlandı. 7 ile 10.5 arasında H'nin değeri 7, 10.5 den sonra ise 10.5'dir. Bu da değişkenlerin değişebilir olduğunun göstergesidir.

```
A = 8
B = 6.9
C = "78"
print(A+B+float(C))
```

⇨ 92.9

Yukarıdaki kodda C değişkeni bir stranger ancak print içinde onu float olarak değiştirdiğimiz için ekrana çıktı alırken onu float'a çevirip toplama işleminin sonucunu yazacaktır.

Print() Fonksiyonu

```
print(“Merhaba Dünya!”)
```

⇨ Merhaba Dünya !

- Örnekte gözüktüğü gibi print() fonksiyonun da ekrana çıkmasını istediğimiz metni tırnak içerisinde girmemiz yeterli oluyor.
- Print fonksiyonunun bir diğer kullanım alanı ise değişkenleri ekrana yazdırmak.

```
A = 5
print(A)
```

⇨ 5

Fonksiyonunu çalıştırdığımız zaman “A” değişkeninin değerini çıktı olarak verir. A değişkenini yukarıda 5 olarak tanımladığımız için çıktı 5 olacaktır.

```
A = 5
B = 9.8
C = “Tarık Aykan”
print(A,B,C)
```

⇨ 5 9.8 Tarık Aykan

Yukarıdaki örnekte olduğu gibi print fonksiyonunda tek print kullanarak birden fazla fonksiyonu çağırmak için virgül kullanılır.

```
A = 7
print(“Yedi”,A)
```

⇨ Yedi 7

Yukarıdaki örnekte ekrana istediğimiz yazıyı çıkartıp yanına değişkenimizi çağırabiliriz.

- Print fonksiyonu ile ekrana değişkenlerin 5 işlemini (toplama”+”, çıkarma”-“, çarpma”*”, bölme”/” ve üs alma”**”) yaptırabiliriz.

```
A = 8
print(A)
A*= 2
print(A)
```

⇨ 8

⇨ 16

- Yukarıdaki kodda A değişkeninin değerini 2 ile çarpıp çıktısını almasını söyledik. Bu metodu diğer 5 işlem içinde kullanabiliriz.

```
A = 5
B = 9.8
C = "Tarık Aykan"
print(A+B)
==>14.8
print(A-B)
==>-4.8
print(A*B)
==>49
print(A/B)
==>0.51
print(A**B)
==>7.077.926,4
```

- ❖ Python'da diğer yazılım dillerinden farklı olarak çıkarma işleminde negatif sonuç alabiliyorsunuz.
- ❖ Değişkenlerin yerleri değişebilir.
- ❖ Integer ile float arasında toplama, çıkarma, çarpma, bölme, üs alma yapılabilir.
- ❖ Integer ile stranger arasında sadece çarpma işlemi yapılabilir.
- ❖ Aynı türler kendi içerisinde 5 işlem yapabilir.
- ❖ Print fonksiyonun içine birden fazla satırda yazı yazmak istiyorsak alttaki örnekte ki gibi 3 tane çift tırnak konur.

```
print("""Tarık
Miraç

Aykan""")
```

➪
```
Tarık
Miraç

Aykan
```

- ❖ Çıktımızın her harf arasına boşluk koymasını istersek, her harfi ayrı ayrı tırnak içine alıp aralarına virgül koymamız yeterlidir.

```
print ("S", "A", "I", "C")
```

➪ S A I C

Sep komutu

- ❖ Eğer çıktımızın harfleri arasına boşluk değil de nokta ya da başka bir sembol koymak istersek sep komutunu kullanırız.

```
print("S", "A", "I", "C", sep = ".")
==========================================================================
==> S.A.I.C
```

Sep komutu kendinden önce yazdığımız sembolü eşitlediğimiz sembol ile değiştirir. Sep komutunun kullanım örneği yukarıdaki gibidir.

- ❖ Nokta yerine herhangi başka bir sembol koyabiliriz. Onun için sep'ten sonraki tırnak içerisine istediğimiz sembolü yazmamız yeterlidir.

Menü Örneği

- ❖ Kendimize bir menü hazırlayalım.

```
İsim = "Tarık"
Soy_isim = "Aykan"
Hobi = "Yapay zeka geliştirme ve siber güvenlik"
Felsefe = "Hayat umursamayana güzeldir 😊"
print("İsim           : ",İsim)
print("Soy İsim       : ",Soy_isim)
print("Hobisi         : ",Hobi)
print("Hayat felsefesi: ",Felsefe)
```

```
⇨ İsim           : Tarık
  Soy İsim       : Aykan
  Hobisi         : Yapay zeka geliştirmek ve siber güvenlik
  Hayat felsefesi : Hayat  umursamayana güzeldir😊
```

Yukarıdaki menü örneğinde print içinde tırnak içerisinde verilerimizi adlandırdık. Sonrasında virgülle ayırarak değişkenlerimizi çağırdık.

Input () Fonksiyonu

- ❖ Input fonksiyonu kullanıcıdan veri almamızı sağlar.

```
input ("Bir sayı girin: ")
```

Kullanıcıdan bir sayı istedik ve açılan konsol ekranına değer girmesi gerekti.

Şimdi kullanıcıdan aldığımız değeri bir değişkene eşitleyip onun çıktısını alalım.

```
A = input ("Bir sayı girin")
print(A)
```

```
⇨ kullanıcın girdiği sayı
```

- ❖ Kullanıcı sayı dışında bir değer girdiği takdirde program hatasız şekilde çalışır. Buna programın açığı denir.
- ❖ Bunu engellemek için input'un veri tipini belirlememiz gerekir.

```
A = int(input("Bir tam sayı girin:  "))
B = float(input("Bir kesirli sayı girin: "))
print(A+B)
```

```
⇨ Kullanıcıdan alınan A değişkeni ile B değişkeni toplamı.
```

- ❖ Kullanıcıdan input fonksiyonu ile sadece sayı değeri alınmaz. Input fonksiyonunu sınırlandırmadığımız zaman tüm değerleri stranger olarak alır.

```
isim = input("İsminizi girin : ")
print("İsminizin", isim," olduğuna emin misiniz ?")
```

Yukarıda ki kodda kullanıcıdan aldığımız veriyi kullanıcıya farklı bir cümle içerisinde sunduk.

Type() Fonksiyonu

- Type fonksiyonu değişkenin tipini gösterir.

```
A = 8
print(type(A))
```

⇨ 'class integer'

Şeklinde kullanılır.

- Class değişkenin sınıfını belirtir.

5 İşlem Makinesi Örneği

```
Sayi_1 = float(input("Bir sayı giriniz : "))
Sayi_2 = float(input("Bir sayı giriniz : "))
```

İki tane değişken oluşturduk ve bunları kullanıcıdan alırken float olmasını istedik. Integer olmasını istesek kullanıcı kesirli sayı girmek istediği zaman program çöker.

```
print("Toplama : ",Sayi_1 + Sayi_2)
print("Çıkarma: ",Sayi_1 - Sayi_2)
print("Bölüme   : ",Sayi_1 / Sayi_2)
print("Çarpma : ",Sayi_1 * Sayi_2)
print("Üs alma : ",Sayi_1 ** Sayi_2)
```

5 işlemi de yapıp ekrana hangi işlem olduğunu yazarak çıkartmasını istedik.

```
Sayi_1 = float(input("Bir sayı giriniz : "))
Sayi_2 = float(input("Bir sayı giriniz : "))

print("Toplama : ",Sayi_1 + Sayi_2)
print("Çıkarma : ",Sayi_1 - Sayi_2)
print("Bölüme  : ",Sayi_1 / Sayi_2)
print("Çarpma  : ",Sayi_1 * Sayi_2)
print("Üs alma : ",Sayi_1 ** Sayi_2)
```

Örneğin tam hali yukarıdaki gibidir. Çıktının güzel gözükmesi için print içindeki iki noktaların aynı hizada kalmalarına özen gösterdik.

Bool() Fonksiyonu

- Bool fonksiyonu değişkenimizin değeri olup olmadığını ölçer, yani değişkene bir değer atanmışsa True, atanmamışsa False çıktısı verir.

Kullanımı aşağıdaki gibidir.

```
A = 9
B = 0
print(bool(A,B)
```

- ⇨ True
- ⇨ False

Yukarıdaki örnekte A değişkenime 9, B değişkenime 0 atadık. Değişkenleri çağırırken aralarına virgül koyarak ikisinin aynı anda ekrana çıktı olarak çıkmasını sağladık. İki defa print kollanmak yerine tek print kullanarak hem kodumuzun temiz gözükmesini hem de hızlı çalışmasını sağladık. Önce True sonra False vermesinin sebebi A değişkenin bir değere sahipken B değişkeni 0 değerine sahip. 0 değeri bilgisayar dilinde hiçlik demektir yani boştur. Bu yüzden ilk çıktı A değişkenin yani True, ikinci çıktı B değişkenin ve False'dır.

- Bool fonksiyonunu kullanıcıdan aldığımız input'u kontrol amaçlıda kullanabiliriz.

```
A = input ("Bir değer girin :")
print(bool(A))
```

Yukarıdaki kod kullanıcının bir değer girip girmediğini görmemizi sağlar. İlerde bunu kullanarak örnek yapacağız.

```
A = input("Bir değer girin: ")
print(type(bool(A)))
```

Yukarıdaki kod kullanıcının bir değer girip girmediğini ve girdiyse tipinin ne olduğunu göstermez.

- Çünkü input ile alınan her değer stranger'dır. Biz bunu integer ya da float diye belirterek değiştirebiliriz. Daha önce ki örnekte de yaptığımız gibi integer almak istiyorsak başına int, float değer almak istiyorsak başına float yazıyoruz.

Koşula Bağlı İşlem

- Kontrol mekanizmalarına giriş yapıyoruz. Kontrol mekanizmaları programın akışına yön veren komut dizinidir.

If, Elif ve Else Komutu

- If komutu koşul koymamızı sağlayan komuttur. Türkçe deki "eğer'in" karşılığıdır.
- Eğer if sağlamazsa elif'e geçer.
- If bir defa kullanılır, diğer olasılıklar için elif kullanılır.
- Else hiçbir koşul sağlanmazsa devreye girer.

```
if a == 5
```

Yukarıdaki kodun Türkçesi eğer a 5' eşit isedir.

If komutunu bool fonksiyonu ile kullanımına örnek verelim.

```
A = input("Bir şey yazın")
if not bool(a):
      print("Lütfen bir şey girin!")

else:

      print("adamsın!")
```

A değişkenine atadığımız input'ta kullanıcı değer girmezse lütfen bir şey girin çıktısı alınır. Değer girerse adamsın çıktısı alınır.

Anket Örneği

```
Şehirler = ("""
1)Ankara
2)Bursa
3)Samsun
4)Ordu""")
A = int(input("Hangi şehirde yaşıyorsunuz : ")
if A == 1:
      print("Ankara'yı seviyor musunuz?")
elif A == 2:
      print("Bursa'yı seviyor musunuz ?")
elif A == 3:
      print("Samsun'u seviyor musunuz?")
elif A == 4:
      print("Ordu'yu seviyor  musunuz?")
else:
      print("Lütfen geçerli bir numara girin")
```

Yukarıdaki kodda Şehirler adında değişkenimizi oluşturduk ve şehirlerimizi numaralarıyla girdik.

Kullanıcının girdiği numaraya göre ekrana çıktı gelecek. Örneğin kullanıcı 1 girerse ekrana Ankara'yı seviyor musunuz? çıktısını alacak

Hesap Makinesi Örneği

```
ifade = input("Yapılacak işlem :")
sayi_1 = float(input("Sayı giriniz :"))
sayi_2 = float(input("Sayı giriniz :"))

if ifade == "+":
   print("Sonuç :", sayi_1+sayi_2)
elif ifade == "-":
   print("Sonuç :", sayi_1-sayi_2)
elif ifade == "/":
   print("Sonuç :", sayi_1/sayi_2)
elif ifade == "*":
   print("Sonuç :", sayi_1*sayi_2)
elif ifade == "**":
   print("sonuç :", sayi_1**sayi_2)
else:
   print("hatalı işlem")

   print("Sonuç :", sayi_1*sayi_2)
elif ifade == "**":
   print("sonuç :", sayi_1**sayi_2)
else:
   print("hatalı işlem")

   print("sonuç :", sayi_1**sayi_2)
else:
   print("hatalı işlem")
```

Yukarıdaki hesap makinesi örneğinde ilk olarak kullanıcıdan yapılacak işlemi girmesini istedik. Ardından hesaplamak istediği 2 rakamı girmesini istedik. Seçtiği işlemin koşuluna bağlı olarak istenilen sonuç ekrana çıktı olarak alındı. Kullanıcının hatalı işlem seçme durumuna karşı da else komutumuzu oluşturduk.

- Sayı değişkenlerinin sayısı arttırılarak 4, 5 değerli işlemler yapılabilir.

Operatörler

- Az önceki konuda kullandığımız eşit ise (==) bir operatör örneğidir. Bu bölümde diğer operatör çeşitlerini göreceğiz.

Eşit Değilse

- Bir değişkenin bir duruma eşitliğini kontrol etmek için kullanılır.
- ! = şeklinde ifade edilir.

Oturum Açma Örneği

Bir kullanıcı adı ve şifre belirleyelim ve kullanıcı bunlar dışında veri girdiğinde hata vermesini sağlayalım.

```
Kullanıcı_adi = input("Kullanıcı adını giriniz : ")
Sifre = input("Şifrenizi giriniz : ")
A = "Frzze"
B = 1234
if Kullanıcı_adi != A:
     print("Hatalı kullanıcı adı!")
elif Sifre != B:
     print("Hatalı Şifre!")
else:
     print("Giriş başarılı")
```

Yukarıdaki örnekte Kullanıcı adını ve şifreyi A ve B değişkenleri olarak tanımladık. Daha sonra kullanıcıdan aldığımızın verilerle karşılaştırdık ve eşit değilse ekrana vermesi gereken çıktıyı belirledik.

Büyüklük Kıyaslama

- Bir değişken değerinin diğer değişken değeri ile arasında büyüklük ilişkisini göstermek için kullanılır.
- "<" küçüktür, ">" büyüktür, "=<" küçük eşittir, "=>" büyük eşittir.

Değer Aralığı Örneği

```
A = float(input("1 ila 10 arasında değer girin : ")
if A < 5:
     print("5'den küçük değer girdiniz.")
elif A > 5:
     print("5'den büyük değer girdiniz.")

elif A > 10:

     print("Lütfen 10'dan küçük değer girin.")

elif A < 1:

     print("Lütfen 1 den büyük değer girin.")
else:
     print("5 rakamını girdiniz.")
```

Yukarıdaki örnekte kullanıcıdan float bir değer istedik ve girdiği değeri 5'i baz alarak aralığını hesapladık. Kullanıcının istenen değerler arasında değer girmesine karşılık elif'lerimizi oluşturduk. Aksi taktirde programın açığı olacağı için kullanıcı 11 girse de çalışacaktı ancak biz başta 10'dan küçük değer istemiştik.

And ve Or

- And İngilizce'de olduğu gibi ve anlamındadır. Or ise veya anlamındadır.
- And operatörü her iki koşulunda sağlanmasını ister. Karpuz ve elma dersen ikisini de alması gerekir, sadece biri sağlandığı takdirde çalışmaz.
- Or operatöründe koşullardan birisinin sağlanması yeterlidir. Karpuz veya elma al dediğimiz zaman tek karpuzda alınsa veya tek elma da alınsa kabul edilir. Her ikisi de alınabilir.

Bir örnek yapalım. A ve B olmak üzere iki değişkenimiz olsun. Bu değişkenleri kullanıcıdan alalım. A 50'den büyük ve B 50'den küçükse ekrana adamsın çıktısını versin.

```
a = float(input("Bir sayı girin : ")
b = float(input("Bir sayı girin : ")
if a > 50 and b < 50:
     print("adamsın!")
```

Bir de "or" ile örnek yapalım. Bu sefer şartlardan biri sağlansa yeterli olacaktır.

```
a = float(input("Bir sayı girin : ")
b = float(input("Bir sayı girin : ")
if a > 50 or b < 50:
     print("adamsın!")
```

In

- In operatörü içermek anlamındadır.
- Kullanıcıdan aldığımız verinin içerisinde istediklerimiz var mı diye kontrol etmekte kullanılır.

Basit bir şifre örneği yapalım.

```
giris = input("Parola belirleyiniz : ")

if "abc" in giris:
    print("abc harfleri vardır! ve yanyanadır.")
else:
    print("abc harfleri kullanın.")
```

Yukarıda ki kodda giriş adlı değişkenimizi kullanıcıdan aldığımız parolaya tanımladık. Alt satırında içerisinde abc varsa ve yan yana ise ekrana "`abc harfleri vardır! ve yanyanadır.`" çıktısını çıkarmasını istedik. Eğer yoksa veya yan yana değilse ekrana "`abc harfleri kullanın.`" çıktısını çıkartmasını istedik.

`Örneği biraz daha geliştirelim. a,b,c harflerine ayrı ayrı baksın.`

```
giris = input("Parola belirleyiniz : ")
if "a" in giris:
    print("a harfi vardır!")
else:
    print("a harfini kullanın")
if "b" in giris:
    print("b harfi vardır!")
else:
    print("b harfini kullanın")
if "c" in giris:
    print("c harfi vardır!")
else:
    print("c harfini kullanın")
```

Yukarıdaki kodda diğer örnekten farklı olarak her bir harfi ayrı ayrı incelemesini istedik.

Not in

- In operatörünün tersidir. İçermesine değil içermemesine bakar.

Basit bir örnek yapalım.

```
giris = input("Parola belirleyiniz : ")

if "?" not  in giris:
    print("? işareti kullanın!!")
else:
    print("Parola başarıyla oluşturulmuştur!")
```

Yukarıda ki kodda in operatörünün tersine, ? işaretinin yokluğuna baktı.

Is ve Id

- Is operatörü eşittir demektir ancak “=” den farklıdır.
- Is operatörü görünüşe değil kimliğe bakar.
- Her bir ifadenin onluk sistemde farklı bir anlamı vardır.
- Her ifadenin onluk tabanda gösterimi bir kimliktir.
- Kimliği aynı olup görünüşü farklı olan ifadeler vardır. Bunlar için is operatörü kullanılır.
- Id operatörü ifadenin onluk tabandaki gösterimini gösterir.

Bunları örneklendirelim.

```
a = 5
b = 5

print(id(a))
print(id(b))

if a == b:
    print("Eşit")
if a is b:
    print("kimlik eşit")
```

Döngüler

- Bir şeyi sürekli olarak başa sarma işlemine döngü denir.
- Neden sürekli başa saracağız diye sorarsanız, biz hesap makinesi yaptığımızda her işlemden sonra program kapanıyordu. Bunun olmaması için döngüleri kullanacağız. Döngüler sayesinde program biz kapatana kadar çalışır.
- Kapanmasını istersek otonomda yapabiliriz.

For

- Koşul belirtiyoruz ve o koşul sağlandığı süre boyunca çalışır.
- Koşullandırmada kullanılan değişkenden yararlanarak geçici değişkenler oluşturur.

Örnek

```
sayilar = "123456789"
for sayi in sayilar:
    print(sayi)
```

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

Yukarıdaki örnekte sayilar adlı bir değişken oluşturduk. Bu değişkeni sadece koşullandırma için kullanacağız. 2.satırda for sayesinde sayi adında yeni bir değişken oluşturduk ancak bu değişken geçici bir değişkendir. Sayi değişkenini değerlerini sayilar değişkenindeki değerleri seçerek oluşturur. 3.satırda da sayi değişkenimizi çıktı aldık.

Örneği geliştirelim.

```
sayilar = "123456789yddhjfgjhhhhgjg"
a = 0
for sayi in sayilar:
    a+=1
    print(sayi)
```

```
⇨ 1
⇨ 2
⇨ 3
⇨
```

Bu örnekte diğerinden farklı olarak sayilar değişkeninin karakter değeri kadar 1 ekleyerek devam ediyor. Sayilar değişkeninin karakter değeri 24.

```
sayilar = "123456789yddhjfgjhhhhgjg"
a = 2
for sayi in sayilar:
    a*=5
    print(sayi)
```

```
⇨ 10
⇨ 50
⇨ 250
⇨
```

Bu sefer a'yı başta 2'ye eşitledik ve 24 kere (sayilar değişkeninin karakter değeri) çıkan sonucu 5 ile çarptı.

Range () Fonksiyonu

- ❖ for ile birlikte kullanılır.
- ❖ Range fonksiyonun içine girdiğimiz değeri 0 dan başlayarak o değere kadar forun oluşturduğu değişkenin içine atar.

Örnek

```
for a in range(100):
    print(a)
```

⇨ 0

⇨ 1

⇨ 2

⇨

Yukarıdaki kod 0'dan başlayarak 100'e kadar (range atadığımız değer) sayar.

- ❖ Eğer 0'dan değil 10'dan başlayarak 100'e kadar gitmesini istersek 100 değerinden önce 10 değerini gireriz.

Örnek

```
for a in range(10,100):
    print(a)
```

⇨ 10

⇨ 11

⇨ 12

⇨

- ❖ Bir bir değil de 10'ar 10'ar artmasını istersek 100'den sonra 10 yazarız.

```
for a in range(10,100,10):
    print(a)
```

⇨ 10

⇨ 20

⇨ 30

- ❖ Yalnızca sayı çıktısı almak zorunda değiliz.

Örnek

```
for a in range(10,100,10):
    print(“Seni Seviyorum”)
```

⇨ Seni seviyorum

⇨ Seni seviyorum

⇨ Seni seviyorum

⇨

Örnekte ekrana 9 kere seni seviyorum yazacak çünkü 10’dan başlayıp 10’ar 10’ar 100’e kadar gitmesini söyledik. 100’ü saymadığı için 9 tane yazacak.

While

- ❖ While döngüsü “ken” anlamına gelir ve sürekli bir dönüş yapar.
- ❖ Sürekli olarak çalışmasını istersek “while True” olarak kullanırız.
- ❖ Çıkış fonksiyonunu çalıştırana kadar program çalışmayı durdurmaz.
- ❖ Bir fonksiyonun sürekli çalışmasını istersek while döngüsünün içinde olması gerekir. Yani tam boşluk olması gerekir.

Örnek

```
while True:
     print(“SAIC”)
```

Bu örnek çalışır ve ekrana programı kapatana kadar SAIC yazdırır.

```
while True:
print(“SAIC”)
```

Bu örnekte while döngüsü çalışmaz yalnızca 1 kere SAIC yazar.

Örnek

```
a = 10
while a == 10:
    print("SAIC")
```

⇨ SAIC

⇨ SAIC

⇨

Bu örnekte a 10 iken ekrana sürekli SAIC yazdırır.

Örnek

```
a = 0
while a <= 10:
    a += 1
    print("SAIC")
```

⇨ SAIC

⇨ SAIC

⇨ SAIC

⇨

Bu örnekte her SAIC yazdığında a'ya bir ekler ve a 10'dan küçükken ve eşitken çalışır. Sonrasında programı kapatır. Buna büyük eşittir falan filan koyarak da yapabiliriz.

Fonksiyonlar

- Daha önce işlediğimiz print ve range bir fonksiyondur.
- Fonksiyon aslında binlerce satır koddan oluşur. Mesela print binlerce satırdan oluşmuş bir koddur ancak bunlar print'in içinde olduğu için biz göremeyiz ve sadece print yazmamız gerekir.
- Değişkenleri nasıl adlandırdığımızda çağırmak için adını yazmamız yeterli ise fonksiyonlarda da öyledir adlandırdığımız zaman binlerce kodu yazmamız gerekmez sadece adını yazmamız yeterlidir.

Def

- Def komutu bizim fonksiyon oluşturmamızı sağlar.

Örnek

```
def toplama():
    toplama = 2 + 8
    print(toplama)

toplama()
```

⇨ 10

Yukarıdaki örnekte def ile toplama adlı fonksiyon oluşturduk. 2. satırda fonksiyonun işlevini tanımladık. 3.satırda toplama fonksiyonun çalıştırdığımızda ekrana çıktı çıkarmasını sağladık. Print yazmazsak fonksiyon çalışmaz. Fonksiyonumuzu oluşturduktan sonra toplama kelimesini farklı renkte yazmaya başladığını gördük. Ekrana toplama yazdığımızda aslında def in içinde tanımladığımız yeri yazmış oluyoruz. Bu sayede kod yazmamız daha pratik bir hale gelmiş oluyor.

- Fonksiyonları tabii ki bu şekilde kullanmıyoruz. Biz bir algoritma kuruyoruz ve bu algoritmayı her yerde kullanabilmek için fonksiyon oluşturuyoruz.

Return ()

- Return fonksiyonu def ile oluşturduğumuz fonksiyonun çalışmasını sağlar.
- Önceki örnekte print kullanmıştık ancak bu sadece 2 3 satırlık fonksiyonlar içindir. Eğer fonksiyon algoritmamız uzunsa return fonksiyonunu kullanırız.
- Print'ten farklı olarak fonksiyonumuzun çıktısını almak için print ile kullanılır.

Örnek

```
def toplama(ilksayi,ikincisayi):
    toplama = ilksayi + ikincisayi
    return(toplama)

print(toplama(5,8))
```

⇨ 13

Yukarıda ki kodda def fonksiyonunda bu sefer ilksayi ve ikincisayi adında iki tane geçici değişken atadık. 2. satırda değişkenleri toplamasını söyledik. 3. satırda fonksiyonumuzun sürekli olarak çalışması ve ekrana çıktı çıkarması için return fonksiyonumuzu kullandık. 5. satırda toplama fonksiyonunun ekrana çıktısını aldık ve geçici iki değişkenimize değer atadık.

- ❖ Def komutunda kaç geçici değişken atamak istersek o kadar değişken adlandırmamız yeterlidir.

Örnek

```
def toplama (ilksayi,ikincisayi,ucuncusayi):
     toplama = ilksayi + ikincisayi + ucuncusayi
     return (toplama)
print(toplama(5,8,2))
```

⇨ 15

Yukarıdaki örnekte 3 değişken adlandırdık ve 3 tane geçici değişkenimiz oldu.

"Dik Üçgen mi?" Örneği

Bu örneğimizde 3 tane değişken alalım ve bunlar üçgenin kenar uzunlukları olsun. Bu kenar uzunluklarını kullanarak hipotenüs teoremine göre üçgenin dik olup olmadığını söylesin.

```
def hipo(a,b,c):
   if a**2 + b**2 == c**2:
      return "Bu bir dik üçgendir!"
   else:
      return "Bu üçgenin bir iç açısı  90 derece değildir!"

print(hipo(3,4,5))
```

⇨ Bu bir dik üçgendir!

```
def hipo(a,b,c):
   if a**2 + b**2 == c**2:
      return "Bu bir dik üçgendir!"
   else:
      return "Bu üçgenin bir açısı 90 derece değildir!"

print(hipo(4,5,6))
```

⇨ Bu üçgenin bir açısı 90 derece değildir.

Kristal Şekil Örneği

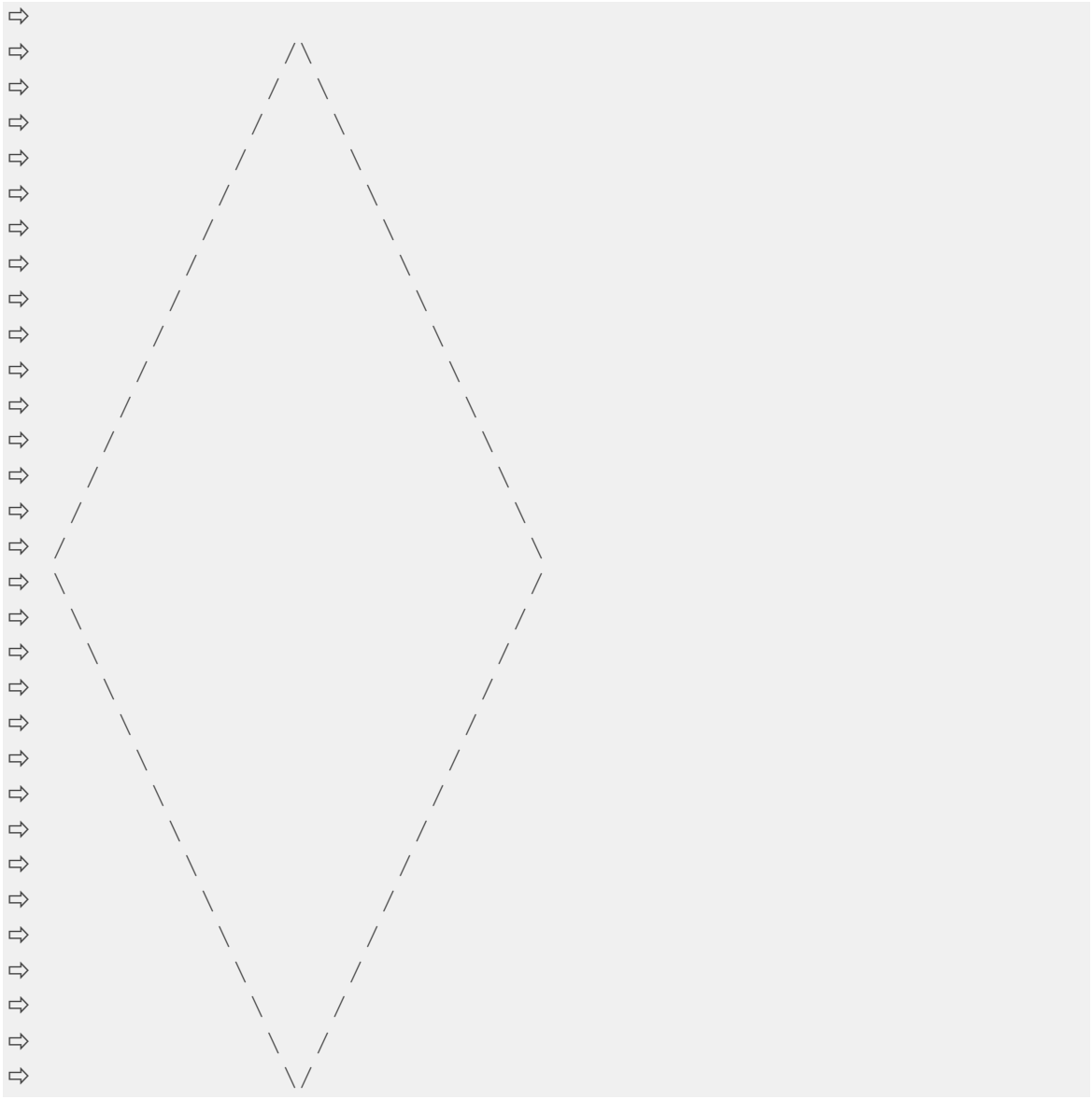
Bu örnek biraz zor bir örnek ancak şimdiye kadar öğrendiklerimizin hepsini kavrayacak bir örnek. Bu örnekte kristal şekli oluşturacağız ve bunu slash(/) ve ters slash(\) işaretleri kullanarak yapacağız. Şeklimizin büyüklüğünü kullanıcı belirleyecek. Algoritma mantığının oturması için güzel bir örnek.

```
def sag(adet):
    for a in range(int(adet)):
        print("/",end="")
#sağ tarafı oluşturması için sağ fonksiyonu oluşturduk. Range
#fonksiyonu ile a değişkenine geçici değer verdik ve kaç adet /
#koyması gerektiğini belirttik. End ile alt satıra geçmesini
sağladık.
def sol(adet):
    for a in range(int(adet)):
        print("\\",end="")
def bosluk(adet):
    for a in range(int(adet)):
        print(" ",end="")

def atla(adet):
    for a in range(int(adet)):
        print()
#slashların başına boşluk koymak için
def üst(cap):
    baslangıc = cap/2-1
    for a in range(int(cap/2)):
        bosluk(baslangıc-a)
        sag(1)
        bosluk(a*2)
        sol(1)
        atla(1)
#şeklin üst kısmını oluşturacak fonksiyon. Çap/2 dememizin sebebi
yarı çap almaya başlarken de yarıçapın bir eksiğinden başlayacak
def alt(cap):
    baslangıc = cap-2
    for a in range(int(cap/2)):
        bosluk(a)
        sol(1)
        bosluk(baslangıc-a*2)
        sag(1)
        atla(1)
def paralel(cap):
    üst(cap)
    alt(cap)

#şeklin kenarlarının paralel olması için
while True:
    boyut = int(input("Çapı giriniz : "))
    print()
    paralel(boyut)

#kodu döngüye sokarak her seferinde çalıştırmak zorunda kalmadık.
```



Kullanıcı 30 değerini girerse böyle bir çıktı alacak.