**Python’da Koşullu İfadeler**

Hepinize tekrardan merhaba. Dergimizdeki Python eğitim serimize kaldığımız yerden devam ediyoruz. Önceki sayılarımızdaki eğitimlerimizi okuyup iyice anladıysanız ve alıştırmalarınızı da yaptıysanız Python öğrenmeye burdan devam edebilirsiniz.

Eğitim serimizde şimdiki konumuz ise “Koşullu İfadeler”. Programımızda belli şartlar sağlandığı zaman yapmak istediğimiz durumları koşullu ifadeler yardımı ile oluştururuz.

Bir koşul oluşturulurken bazı operatörler kullanılır. Bu operatörler:

“==” : eşit ise anlamına gelir.

“!=” : eşit değil ise anlamına gelir.

“<” : küçük ise anlamına gelir.

“<=” : küçük veya eşit ise anlamına gelir.

“>” : büyük ise anlamına gelir.

“>=” : büyük veya eşit ise anlamına gelir.

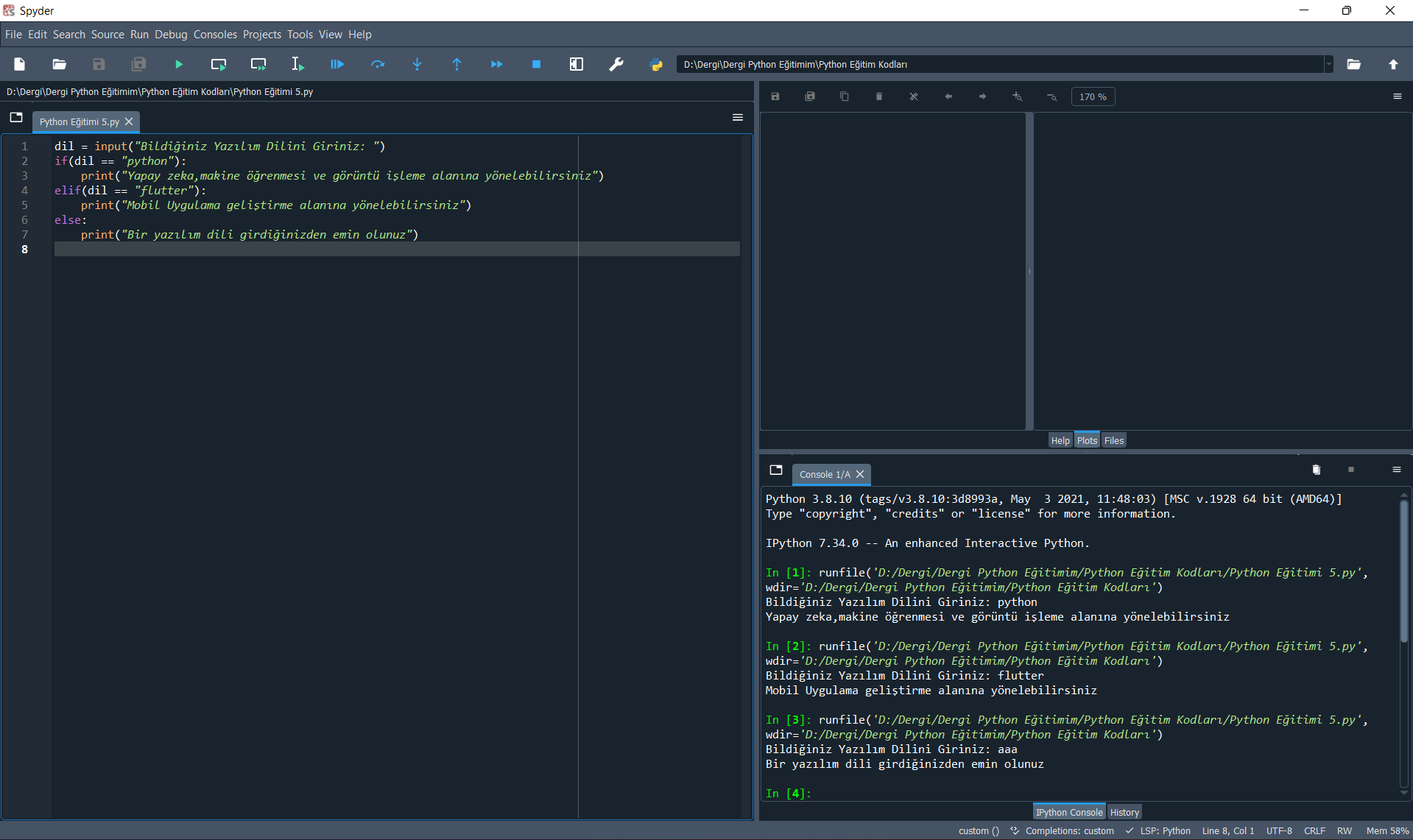
“and”: eğer iki koşulun aynı anda sağlandığında durumların gerçekleşmesini istiyorsak if yapısında parantezin içine iki koşul yazılır ve aralarına bu operatör konulur.

“or”: eğer iki koşuldan biri veya her ikisi sağlandığında durumların gerçekleşmesini istiyorsak if yapısında parantezin içine iki koşul yazılır ve aralarına bu operatör konulur.

Python’da koşullu ifadeleri oluştururken kullanıdığımız yapılar ise **“if,elif ve else”** yapılarıdır. if yapısı pythonda koşullu ifadelerde oluşturulması gereken ilk yapıdır. if yapısı “if(koşul):” şeklinde oluşturulur ve alt satıra geçildiğinde bilgisayarımızdaki “TAB” tuşu yardımı ile girinti oluşturarak altta yazacağımız kodların o koşul sağlandığında yapılması gereken kodlar olduğunu Python’a anlatmış oluruz.

Koşullu ifadeleri daha iyi anlamak için örnek olarak sizin bildiğiniz yazılım diline göre hangi alana yönelmeniz gerektiğini söyleyecek bir program yapmak istersek bu programda ilk başta önceki sayımızda öğrendiğimiz input() fonksiyonu yardımı ile kullanıcımızdan hangi dili bildiğini alacağız ve if yapısı ile yazılan dil ile koşulumuzdaki dilin eşleşmesi durumunda ekrana hangi alan ile ilgilenmesi gerektiğini yazacağız.

* dil = input("Bildiğiniz Programlama Dilini Giriniz: ”)
* if(dil == “python”):
  + print(“Yapay zeka, veri bilimi veya görüntü işleme alanına yönelebilirsiniz.”)

şeklinde koşul yapımızı oluşturabiliriz.

Peki biz bu koşul ifademize farklı koşullar da eklemek istiyorsak bunu nasıl yapmamız gerekiyor. Bunun için de bize “elif(koşul): ” yapısı yardımcı olur. Bu yapı if yapısından farklı olarak üstteki koşul gerçekleşmiyorsa bu koşul çalışır. Yani üstteki koşul gerçekleşiyorsa buradaki koşula hiç bakılmaz ve altındaki durumlar oradaki koşul sağlanıyor olsa bile gerçekleşmez.

* elif(dil == “flutter”):
  + print(“Mobil Uygulama geliştirme alanına yönelebilirsiniz.”)

Farklı bir durum olarak ise yukarıdaki koşulların hiçbirinin gerçekleşmediği durumlarda çalıştırmak istediğimiz kodlar olabilir. Bunun için de kullandığımız yapı “else: ” yapısıdır. Yukarıdaki koşulların hiçbirinin gerçekleşmemesi durumunda kodumuz bu yapının altına girer ve oradaki kodları çalıştırır.

* else:
  + print(“Lütfen bir kodlama dili girdiğinizden emin olun”)

Bu kısımda örnek olması için böyle bir uygulama yaptık. Tabiki iki tane yazılım dili çeşidi olmadığı için burada yazılım dili girsek bile kodumuz else yapısına girebilir. Siz kendiniz denemek için bu koşulları artırarak hem koşul yapılarını iyice kavrayabilirsiniz hem de bu alıştırmalarınız ile python öğrenme süreciniz hızlandırmış olursunuz.

**While Döngüsü**

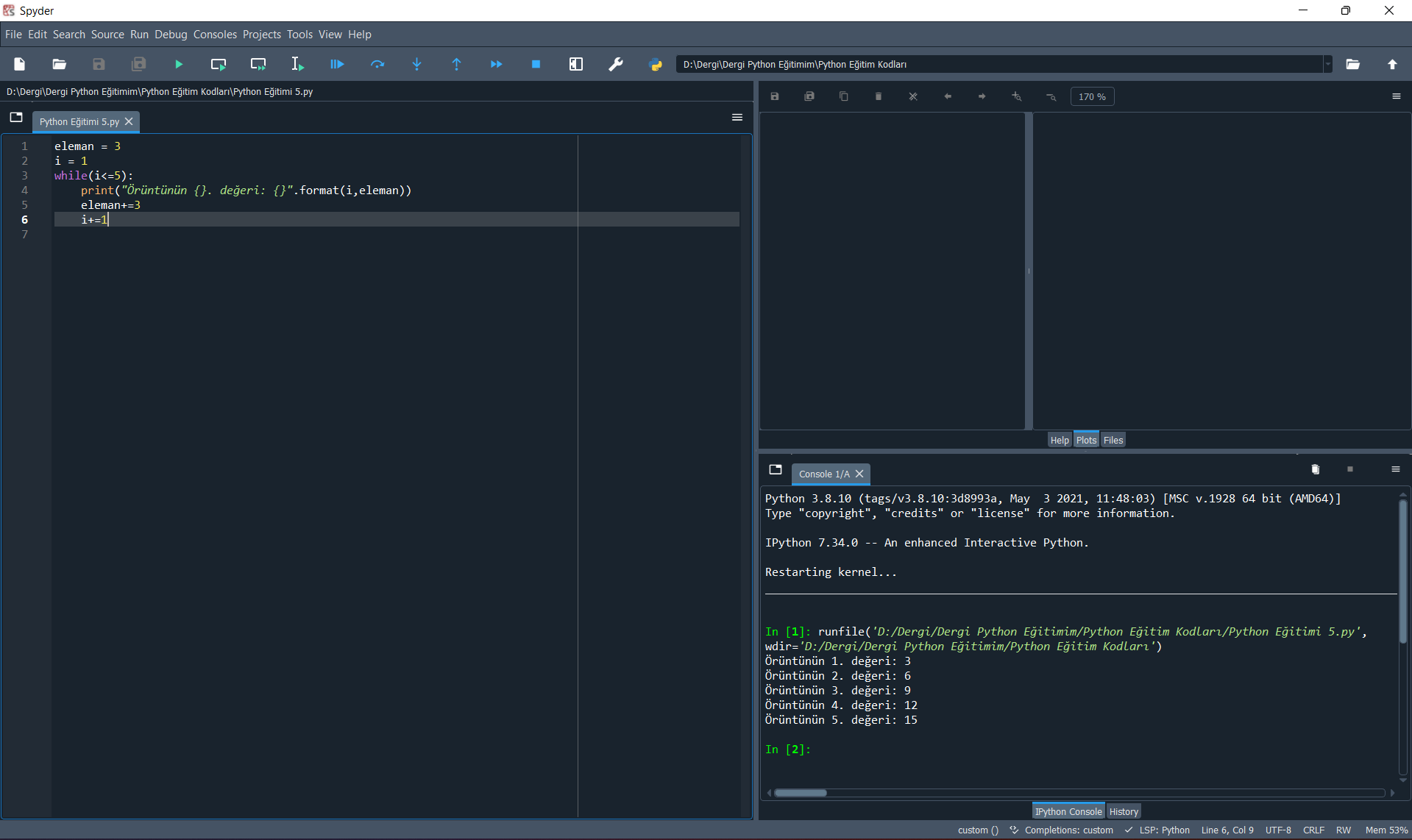
Döngüler Python’da en çok kullanılan ve en önemli konulardan biridir. Bir durumun kod başlatıldığında birden fazla kez gerçekleşmesini istediğimiz zamanlarda döngüler bize yardımcı olur. Döngüler kısmında ilk öğreneceğimiz döngü “while” döngüsü olacak. While döngüsünün yapısını inceleyecek olursak:

1- İlk başta bir döngü değişkeni tanımlanır. Bu tanımlamayı sonsuz bir döngü yapmak istediğimiz durumlarda yapmaya ihtiyacımız yoktur.

2- while(koşul): şeklinde while yapısı oluşturulur. Koşullu ifadelerdeki gibi while yapısında da altındaki kodların while yapısının içinde olduğunu belirtmek için girinti bırakırız. Ayrıca eğer sonsuz bir döngü oluşturmak istiyorsak koşul kısmına “True” kelimesini yazarak koşulun sürekli doğru olmasını sağlarız.

3- while yapısının altında bulunacak tüm kodlar yazıldıktan sonra da onların altına yine girintili bir şekilde döngü değişkenini artırma işlemi yapılır. Bu sayede döngünün kaç kere tekrarlanmasını istiyorsak ona göre bir ayarlama yapmış oluruz.

while döngüsünü de anlamak için bir örnek geliştirelim. Bu örneğimizde üçer üçer artan bir örüntünün ilk 5 elemanını oluşturmak isteyelim. Bunun için ilk başta örüntünün başlangıç elemanının bir değişken atıyoruz. Daha sonra ise döngünün kaç kere gerçekleşeceğini belirleyeceğimiz bir döngü değişkeni oluşturuyoruz. Daha sonra ise while yapısını kurarak örüntünün elemanlarını bastırıyoruz.

* eleman = 3
* i = 1 //döngü değişkeni
* while(i<=5):
  + print(“Örüntünün {}.elemanı: {}”.format(i,eleman))
  + eleman +=3
  + i+=1

Son iki satırda kullandığımız “+=” parametresi aslında bir kısaltma olarak kullanılmaktadır. Bu parametre “eleman = eleman + 3” ve “i = i + 1” işlemlerinin kısaltılması olarak göze daha hoş gelen bir kullanım olarak yazılmaktadır.

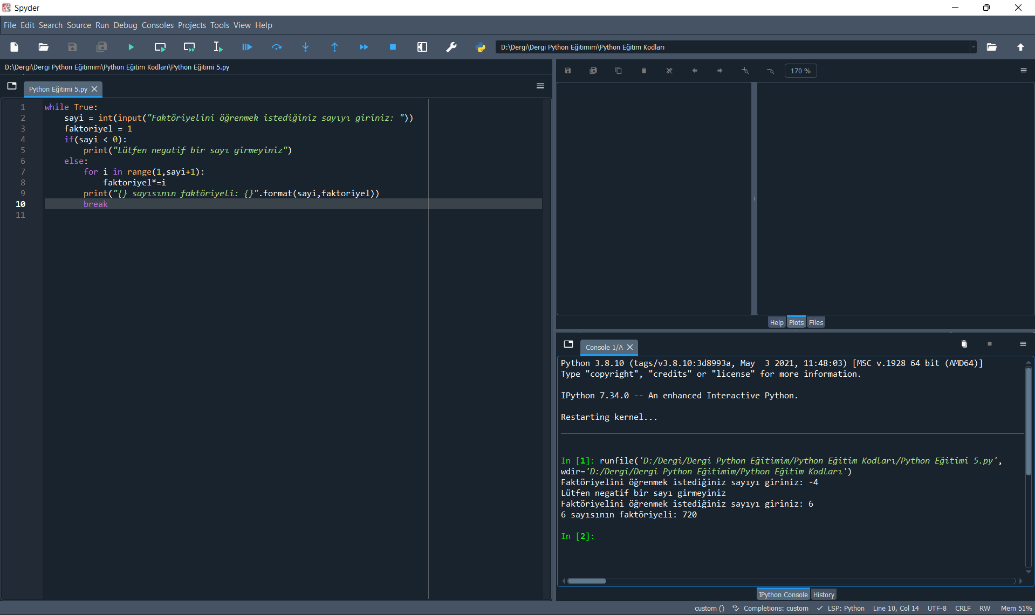
**For Döngüsü**

Python’daki diğer döngü çeşidimiz ise “for” döngüsüdür. Bu döngümüz belirli bir liste veya string’in her bir elemanını döngü değişkenine atayan bir döngüdür. Yani örneğin elimizde “FLOZOF” string değeri varsa for döngümüz 1. turunda tanımladığımız döngü değişkenine “F” değerini atar , 2. turunda bu değişkene “L” değerini atar ve bu şekilde sırayla atama işlemelerini gerçekleştirir. Aynı şekilde elimizde [0,1,2,3,4,5] şeklinde bir liste varsa da for döngüsü ilk önce 0 değerini daha sonra 1 değerini değişkene atar ve her atamada da döngünün altındaki işlemleri de yaparak devam eder. String veya listemizdeki eleman sayısı bitince bu döngü de sonlanır.

ÖNEMLİ NOT!!: Yukarıda örneğini verdiğimiz gibi ardışık sayılardan oluşan bir liste oluşturmak istiyorsak bunun için “range(ilk sayı,son sayı,artış miktarı)” fonksiyonunu kullanırız. Fakat son sayının listeye dahil olmayacağına dikkat edilmelidir. Bu fonksiyonda artış miktarını yazmazsak otomatik olarak 1 alınır. Ayrıca ilk sayıyı yazmazsak da otomatik olarak 0 alınır. Bu fonksiyon for döngüsü ile birlikte çok kullanılan bir fonksiyondur.

for döngüsü yapı olarak “for i in değişken: ” şeklinde kullanılır. Burada i şeklinde isimlendirdiğimiz döngü değişkenidir siz farklı bir şekilde de isimlendirebilirsiniz. Ayrıca bu döngünün altına yazılacak durumlar da bu döngünün altında olduğunun kodumuz tarafından anlaşılabilmesi için girintili şekilde yazılır.

ÖNEMLİ NOT!!: Döngülerde çok fazla kullanılan ve bize yardımcı olan iki tane kelime bulunuyor. Bunlar break ve continue kelimeleri. “break” kelimesi kodda yazıldığı yerde altta yapılacak şeyler olsa bile döngüyü direk orada bitirir ve döngü bir daha çalışmaz. “continue” kelimesi ise yazıldığı yere gelindiğinde altta yapılacak şeyler olsa bile döngünün başa dönmesini sağlar.

for döngüsünü de bir örnekle daha iyi anlayalım. Bu örneğimizde bir sayının faktöriyelini almak için for döngünü kullanacağız. Bu program için ilk başta kullanıcıdan faktöriyelini öğrenmek istediği sayıyı alacağız. Daha sonra ise bu sayının negatif bir sayı olup olmadığını kontrol ediyoruz. Eğer pozitif bir sayı ise de 1 den başlayarak verilen sayıya kadar döngü yardımıyla sayıyı değişkenimiz ile çarparak en sonda da sonucumuzu oluşturarak bastırabiliriz. Bu durumları da az önce öğrendiğimiz while döngüsü içinde yazarak negatif bir sayı yazıldığında kullanıcıdan tekrar bir sayı istenmesini sağlayacağız. Ayrıca break kelimesi yardımıyla faktöriyeli hesapladıktan sonra döngünün sonlanmasını sağlayacağız.

* while True:
  + sayi = int(input(“Faktöriyelini öğrenmek istediğiniz sayıyı giriniz: ”))
  + faktoriyel = 1
  + if(sayi < 0):
    - print(“Lütfen negatif bir sayı girmeyiniz”)
* else:
  + for i in range(1,sayi+1):
    - faktoriyel\*=i
* print("{} sayısının faktöriyeli: {}”.format(sayi,faktoriyel)”)
* break

**ÖDEV:** Siz de yukarıda öğrendiğiniz koşul yapılarını tam olarak özümseyebilmek için kendiniz kullanıcıdan kullanıcı adı ve şifre alıp bunu kendi seçtiğiniz değerlere eşit olup olmamasına göre giriş yapılabilecek bir uygulama geliştirebilirsiniz. Bu uygulama aynı zamanda giriş yapıldığında programı bitirecek ve yanlış giriş yapılırsa da başa dönecek bir şekilde döngüleri de programınıza entegre edebilirisiniz.

Bu sayımızdaki eğitimin sonuna geldik.Umuyorum kendinizi geliştirdiğiniz yararlı bir eğitim olmuştur. Kendiniz araştırarak ve yeni şeyler deneyerek kendinizi geliştirmeyi unutmayın. Ayrıca herhangi bir sorunuz olursa Python’la, anlattığımız konularla veya yazılımla ilgili bana [isaadahanunal@gmail.com](mailto:isaadahanunal@gmail.com) mail adresinden ulaşabilirsiniz. Bir sonraki sayımızdaki Python Eğitimimizde görüşmek üzere.

**İsa Adahan Ünal**