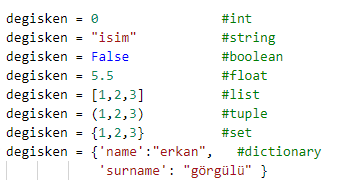
**PYTHON TEMELLERİ**

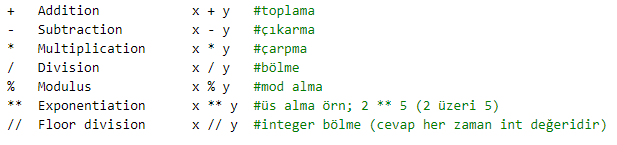
Bu dökümantasyon en az bir programlama dili bilen, Python ’ a hızlı bir giriş yapmak isteyen kişiler için hazırlanmıştır.

**Değişkenler**

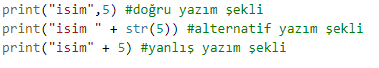
Python’ da değişken tanımlarken açıkça tipibelirtilmez fakat içine verilen default değeri değişkenin tipini belirtir.



Değişkenler ile kullanılabilecek aritmetik operatörler ise şunlardır;



**print()**

Ekrana parametre olarak gönderilen değeri ekrana yazar ve bir alt satıra geçer. Dikkat edilmesi gereken bir kural vardır. String ve int tipinde 2 değişkeni kendisi birleştiremez ve TypeError hatası verir.

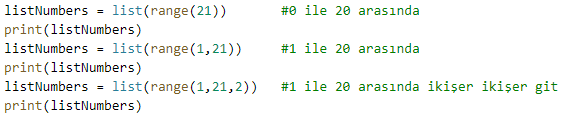
**input()**

İnput methodu dışarıdan bir değer almamızı sağlar ve içine parametre olarak string bir değer gönderebiliriz. Gönderdiğimiz değer veri alınmadan önce ekranda gösterilir.



**range()**

Range methodu bize belirtilen aralıkta sayı üretir. İçine 3 tane parametre alır. İlki, hangi değerden başlayacağı kendisi de dahil olmak üzere, ikinci parametre ise nerede biteceği fakat kendisi dahil değil. Son parametresi ise değer aralığıdır. Oluşturulan veri kümesi, eğer bir değişken içinde saklanacak ise dizi tiplerinden birine dönüştürülmelidir.(örn; list())

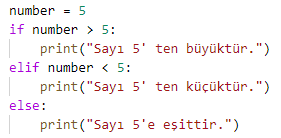
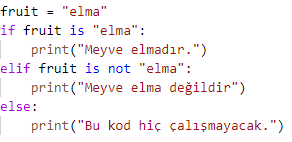


Üstteki kodun ekran çıktısı ise;



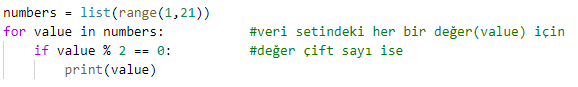
**if, elif, else**

Python’ da if koşulları için parantez kullanılmaz ve neredeyse düz İngilizce yazı şeklinde yazılır. Ama girinti kurallarına dikkat edilmelidir yoksa IndentationError verir. Python, if koşulunu girintiler sayesinde algılar.

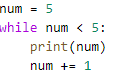
**For Döngüsü**

Pythondaki for döngüsünde döngü verilen aralık kadar değil içinde dönülen veri kümesinin uzunluğu kadar gider. Basitçe C#’ taki foreach döngüsüne benzer.



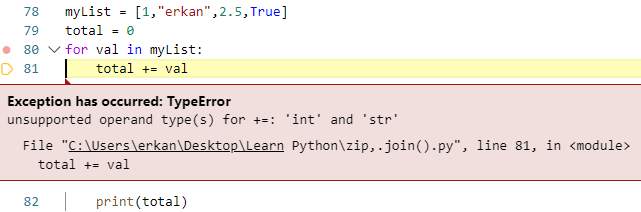
**While Döngüsü**

While döngüsü diğer dillerden farklı olarak sadece yazım şeklinde (syntax) değişiklik gösteriyor. Koşul sağlandıkça içinde dönmeye devam eder.



**Veri Kümeleri**

Python’ da veri kümeleri içinde her türden veri tipini aynı anda barındırabilir. İnt tipinde bir değişken ile string tipinde bir değişken aynı veri kümesi içinde bulunabilir. Fakat bu bazı riskleri beraberinde getirir. Örneğin içinde farklı tiplerde veriler barındıran bir seti dönerken aritmetik işlem yapmak TypeError almamıza sebep olabilir.



**List**

Listler içinde veri kümesi tutarlar. Diğer dillerdeki listler ile hemen hemen aynı özelliklere sahiptir. En basit haliyle bir list oluşturmak için ihtiyacımız olan tek şey köşeli parantezlerdir([ ]).



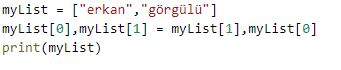
Listlerde indexteki değere erişmek için köşeli parantezler kullanılır.



Python’da aynı zamanda negatif indexleme vardır. Bu, değerlere tersten erişim sağlar örneğin listin son elemanına erişmek için;



Listlerde aynı zamanda 2 değerin karşılıklı olarak yerlerini değiştirebileceğimiz hazır bir syntax vardır.

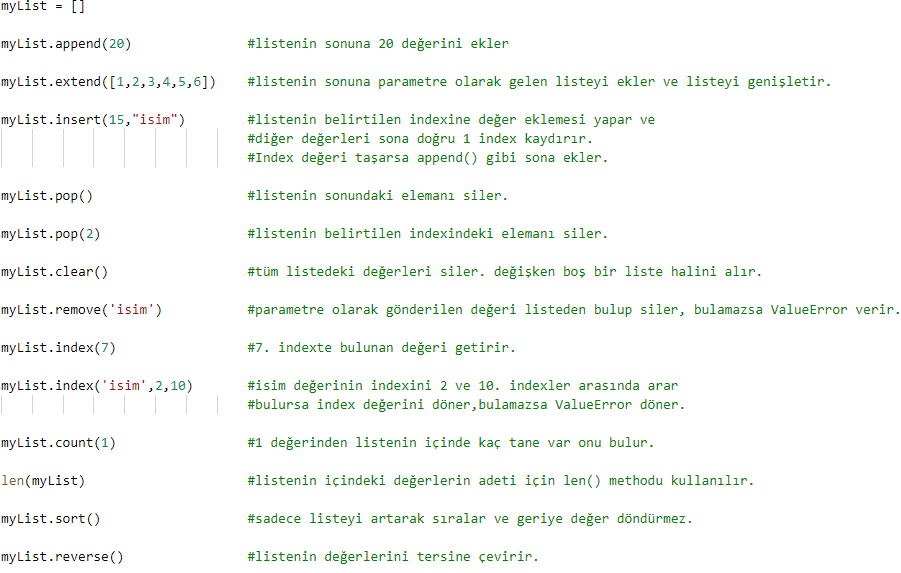


Üstteki kodun çıktısı aşağıdaki gibi olacaktır.



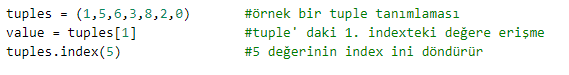
**List methodları**

Listler için Python’ da hazır gelen methodlar ise şu şekildedir;



**Tuple**

Tuple’ lar listlerden farklı olarak sadece bir kere tanımlanırlar ve değiştirilemezler. Tuple’ lara ekleme veya çıkarma uygulanamaz. Sadece indexlere erişim vardır. Tuple’ lar sabit oldukları için listlerden daha performanslıdır. Çoğu method geriye değer döndürdüğünde Tuple tipinde döndürür. Tanımlamak için normal parantezler kullanılır.



**Set**

Setlerde bir diğer veri kümesi tipidir. İçindeki her bir değer eşsizdir ve aynısından bir tane daha eklenemez. Eklemeye çalışıldığında ise Python değeri görmezden gelir ve eklemez. Sıralı değillerdir. Bu yüzden içindeki değerler hangi sıra ile karşımıza çıkacak bilinmez. Değerler indexlere sahip değildir. Haliyle indexlerle erişim sağlanamaz. Ama içinde döngü ile dönülebilir. Kıvırcık parantez ile tanımlanır.

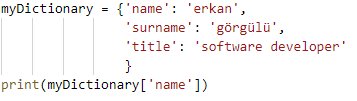


Python’ da aynı zamanda 1 değeri True değerine ve 0 değeri de False değerine karşılık gelir. Haliyle üstteki kodun çıktısı aşağıdaki gibi olur. Her zaman ilk yazılan değeri alır sonra gelen aynı değeri görmezden gelir. False değeri 0 ‘ dan önce geldiği için False setin içine girerken 0 giremez.



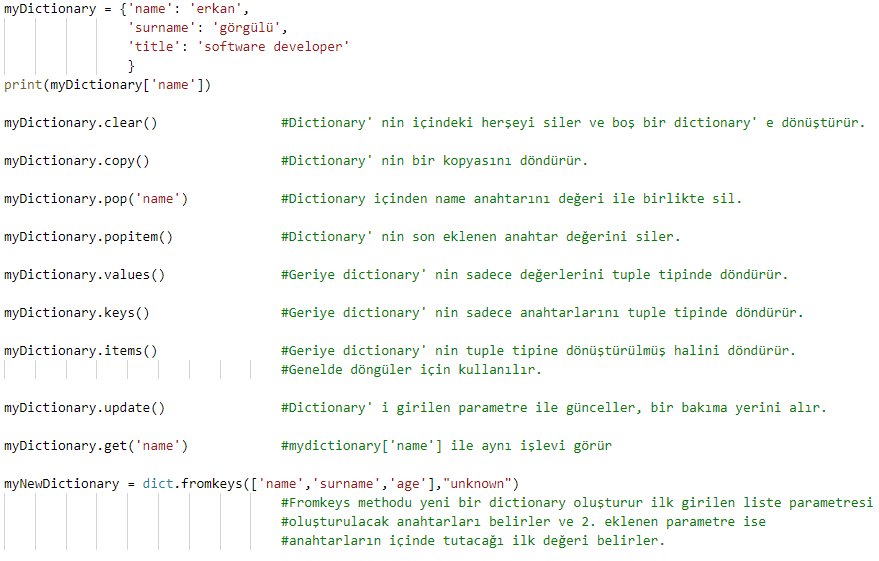
**Dictionary**

Dictionaryler diğer veri kümelerinden farklı olarak key, value (anahtar, değer) şeklinde çalışırlar. Her bir anahtarın kendisine ait bir değeri vardır. Index yerine anahtarlar üzerinden değere ulaşılır.



**Dictionary Methodları**

Dictionaryler için Python’ da hazır olarak gelen methodlar ise şu şekildedir;



**Comprehensions**

**List Comprehension**

List comprehensionlar Python’ da listeleri yaratmak için bir bakıma kısa yol diyebiliriz. Python köşeli parantezler içindeki döngüyü varsa koşula uygun şekilde dönerek bize bir liste döner.



Köşeli parantez içinde olan işlemde, 1’ den 20’ ye kadar olan sayılar içinde, her bir değer için koşul sağlanırsa, yani çift sayi ise, değerin küpünü al ve myList’ in içine at. Bu durumda karşımıza aşağıdaki gibi bir sonuç çıkıyor;



**Dictionary Comprehension**

List comprehension’ ların yanında aynı zamanda dictionary comprehensionlar da var. Bunlar ise key, value şeklinde döngü içinde dönülüyor ve geriye yine key, value olarak dönülüyor.

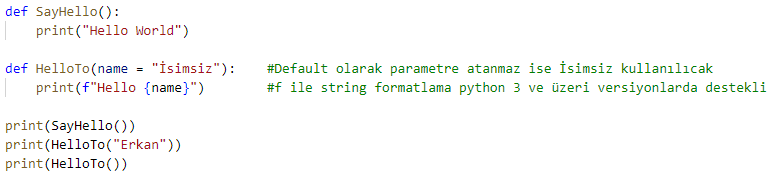


Yine aynı şekilde 1’ den 10’ a kadar olan sayılar içinde, eğer değer tek sayı ise, değerin kendisini anahtar olarak, üzeri 2 alınmış halini ise değer olarak myDictionary içine at. Bu işlemden sonra karşımızda şöyle bir sonuç çıkıyor;



**Functions**

Python’ da fonksiyonların dönüş tipi belirtilmez. Parametre alınırken parametrelerin tipi belirtilmez. Bu durum bazı zamanlarda riskler doğursa da çoğunlukla kolaylık sağlar. Fonksiyon tanımlanırken başına def yazılır.



Aynı zamanda fonksiyonun alacağı parametre sayısı belirsiz ise tek bir parametre adı verilip başına \* eklendiğinde girilen her değeri bir tuple kümesinde tutar.

