**Veri Tipleri ve Veri Girişi**

**input() Fonksiyonu**

Kullanıcıdan veri almamızı sağlar. Alınan veriler her zaman **string** tipindedir. Sayısal olarak kullanacağımız bir veri alıyor ve aritmetik işlemler yapacaksak **int()** ‘a dönüştürmeliyiz.

**Veri Tiplerine Dönüştürmek**

**int() :**  Uygun olan verileri sayısal , integer veri tipine dönüştürmek için kullanılır.

**str() :** Verileri string, yani metin veri tipine dönüştürmek için kullanılır.

**float() :** Tamsayı, noktalı sayısal değerleri olacak şekilde dönüştürmek için kullanılır.

**complex() :** Karmaşık sayılara ait veri tpine dönüştürmemizi sağlar.

Her sayı bir karakter dizisine dönüştürülebilir fakat, karakterler sayısal değerlere dönüştürülemez.

**len()**, String türündeki verilerin uzunluğu, karakter sayısını verir.

String değerleri **split()** metodu ile bölebiliriz. Bölüm sonrası çıktı **list()** türünde olacaktır.

>> kardiz = "İstanbul Büyükşehir Belediyesi"

>> print(kardiz.split())

['İstanbul', 'Büyükşehir', 'Belediyesi']

**print() Fonksiyonu**

Ekrana değer yazdırmak için kullanılır.

**format() Metodu**

Kullanıcıya yansıtılacak değişkenlerin daha nizami bir şekilde kod içerisine yazılmasına olanak sağlar. Örneğin doldurulması gereken yerleri olan bir dilekçe taslağında ilgili yerlere **{}** işareti koyarak, sırasıyla girilen verileri, yine sırasıyla ilgili yerlere ekler.

>> print(" {} ve {} iyi bir ikilidir".format("Python", "Django"))

>> metin = "{} ve {} iyi bir ikilidir"

>> metin.format("Python", "Django")

Python ve Django iyi bir ikilidir

**eval() ve exec() Fonksiyonları**

**eval()** : Kullanıcının girdiği her veriyi bir Python komutu olarak algılar ve bu veriyi işleme sokar. Kullanılacağı durumlarda sıkı bir kontrol mekanizmasının olması gerekmektedir.

**exec()** : eval() ile yapamadığımız bazı şeyleri yapamızı sağlar. Bu fonksiyon yardımıyla karakter dizileri içerisinde çok kapsamlı Python kodları çalıştırabiliriz. Örneğin eval() içerisinde değişken tanımlaması yapamazken, exec() içerisinde bunu yapabiliriz.

**Liste ve Demetler**

**list(),** içinde birden fazla veri barındıran fonksiyonlardır. Barındırdığı verilere **index** üzerinden, **for** ya da **map** fonksiyonları ile erişebiliriz. Map’ ın çıktısı objectir. İç içe liste eklemesi yapabiliriz.

>> liste = ["öğe1", "öğe2", "öğe3"]

Listenin elemanlarına erişmek için **for** döngüsünü kullanırız.

>> meyveler = ["elma","armut","karpuz"]

>> for i in range(len(meyveler)):

>>    print("{}. {}".format(i,meyveler[i]))

1. elma
2. armut
3. karpuz

Öğe değişikliği yapmak istersek, liste içerisinden ilgili index değerine atama yaparız.

>> meyveler = ["elma","armut","karpuz"]

>> meyveler[0] = "çilek"

>> print(meyveler)

['çilek', 'armut', 'karpuz']

**len()** fonksiyonu liste içi veri tiplerinin sayısını söyler.

>> diller = ["İngilizce", "Fransızca", "Türkçe", "İtalyanca"]

>> print(len(diller))

Listeden öğe silmek için **del** ifadesini kullanırız.

>> meyveler = ["elma","armut","karpuz"]

>> del meyveler[-1]

**Liste Üreteçleri (****List Comprehensions)**, kodları satır satır yazmak yerine tek satırda daha kısa halletmemizi sağlar. İç içe ne kadar çok kod yazarsak, işlev ve okunaklığını kaybeder.

>> liste = [i for i in range(1000) if i % 2 == 0]

Normalde yazmak isteseydik şöyle olacaktı.

>> liste = []

>> for i in range(1000):

>>    if i % 2 == 0:

>>        liste += [i]

**append()**, listeye öğe eklemek için kullanırız. Her seferinde tek öğe eklenebildiği için çoklu eklemelerde **for** döngüsü kullanmak gerekir.