Python ile Çocuklar için Programlama

Mustafa Murat Coşkun

İçindekiler

Bölüm 1: Python ve Programlama 1

Programlama Dünyasına Giriş 2

Programlama Nedir? 3

Programlama Dili Nedir? 4

Python'ın Kurulumu 5

Bilgisayar Python Dilini Nasıl Anlar? 6

Etkileşimli Python Kabuğu ve Python Dosyaları 7

Neler Öğrendik? 10

Alıştırmalar 11

Çözümler 11

Bölüm 2: Değişkenler, İfadeler ve Deyimler 13

Python'daki Değerler ve Türleri 14

Değişken Nedir? ve Nasıl Tanımlanır? 15

Değişkenlere Değer Atama ve İşlem Sırası 16

Değişken İsimleri 17

Matematiksel İfadeler 19

Tür Dönüşümleri 22

Neler Öğrendik? 24

Alıştırmalar 25

Çözümler 25

Bölüm 3: Koşullu Durumlar ve İfadeler 27

Karşılaştırma Operatörleri 28

Mantıksal Operatörler 32

and Operatörü 32

or Operatörü 33

not Operatörü 34

Yorum Satırları 35
Python'da Girintiler 36
Koşullu İfadeler 37
if-else Kalıbı 40
if-elif-else Kalıbı 42
İç İçe İfadeler 45
Neler Öğrendik? 47
Alıştırmalar 48
Çözümler 50

BÖLÜM 4: FONKSİYONLAR 53

Fonksiyon Nedir? 54
Fonksiyonların Tanımlanması 55
Fonksiyon Çağrıları 56
Fonksiyon Girdileri 58
Fonksiyon Çıktıları 60
Kullanıcıdan Bilgi Almak : input() Fonksiyonu 63
Ekrana Değer Bastırmak: print() Fonksiyonu 67
Birbirini Çağıran Fonksiyonlar 68
Neler Öğrendik? 70
Alıştırmalar 71
Çözümler 73

BÖLÜM 5: DÖNGÜ MANTIĞI 77

Giriş 78
While Döngüsü 78
Break ve Contınue 82
Neler Öğrendik? 85
Alıştırmalar 86
Çözümler 88

BÖLÜM 6: LİSTELER 91

Giriş 92

Listelerin Elemanlarına Erişmek 93

Listelerin Elemanlarını Değiştirmek 94

Listeleri Birleştirmek 95

in Operatörü 96

Listelerde for Döngüsü 96

Listelerin Fonksiyonları 101

Listeleri Parçalamak 105

İç İçe Listeler 108

Neler Öğrendik? 109

Alıştırmalar 110

Çözümler 112

BÖLÜM 7: KARAKTER DİZİLERİ 115

String Nedir? 116

String'ler ve Listeler Arasındaki İlişki 117

String'lerdeki Özel Karakterler 119

String'lerin Fonksiyonları 120

Neler Öğrendik? 126

Alıştırmalar 127

Çözümler 128

BÖLÜM 8: DEMETLER VE SÖZLÜKLER 131

Demet Nedir? 132

Demetlerin Elemanlarına Erişim 133

Demetlerin Listelerden Farkı Nedir? 134

Sözlük Tanımlamak 134

Sözlük Elemanlarına Erişim 136

Sözlüklere Eleman Eklemek 138

Sözlüklerde Değişiklik Yapmak 138
Sözlüklerde Kullanılan Fonksiyonlar 139
Neler Öğrendik? 143
Alıştırmaları 144
Çözümler 145

BÖLÜM 9: MODÜLLER 149

Modül Nedir? 150
Modül Yazmak ve Çağırmak 151
Rastgele Sayı Üretmek: random ve time Modülü 156
Sayı Tahmin Oyunu 157
Neler Öğrendik? 159
Alıştırmalar 160
Çözümler 161

BÖLÜM 10: DOSYALAR 165

Dosya Oluşturmak 166
Dosyalara Yazmak 167
Dosyadan Veri Okumak 168
Dosyaları Otomatik Kapatma 171
Dosyaları İleri ve Geri Sarmak 172
Dosyalarda Değişiklik Yapmak 174
Dosyaların Sonunda Değişiklik Yapmak 174
Dosyaların Başında Değişiklik Yapmak 175
Dosyaların Ortasında Değişiklik Yapmak 176
Neler Öğrendik? 178
Alıştırmalar 179
Çözümler 180

SON SÖZ 179

Dizin 184

1

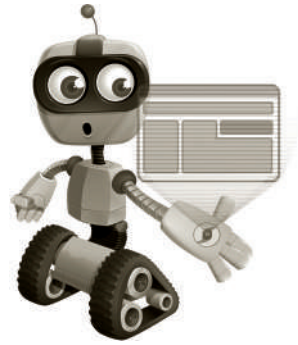
Python ve Programlama

Bu Bölümde

Programlama Dünyasına Giriş	2
Programlama Nedir?	3
Programlama Dili Nedir?	4
Python'ın Kurulumu	5
Bilgisayar Python Dilini Nasıl Anlar?	6
Etkileşimli Python Kabuğu ve Python Dosyaları	7
Neler Öğrendik?	10
Alıştırmalar	11

Hayranlık duyduğunuz oyunların, animasyon filmlerinin geliştirme süreçlerini hiç merak ettiniz mi?

Bu bölümde, merak ettiğiniz soruların cevabını arayacak ve programlama hakkında bilgi sahibi olacaksınız. Ayrıca, öğreneceğiniz bilgilerle beraber kitap boyunca gereken motivasyona sahip olacaksınız.



PROGRAMLAMA DÜNYASINA GİRİŞ

Eğer bu satırları evinizde okuyorsanız çevrenizdeki eşyaları birlikte inceleyelim. Buzdolabınıza, televizyonunuza, bilgisayarınıza, akıllı telefonunuza ve tüm elektronik aletlerinize dikkatlice bakın. Bir tuşa basıyorsunuz ve buzdolabınız sıcaklığı belli bir seviyeye kadar düşürüyor. Televizyonunuzun otomatik olarak kapanması için zaman ayarlaması yapıyorsunuz ve belirlediğiniz sürede kapanıyor. Akıllı telefonunuzla uzaktaki akrabanızla görüntülü konuşuyor ve onlara fotoğraflarınızı gönderiyorsunuz. Boş zamanınızda çok sevdiğiniz bir bilgisayar oyunuyla eğlenceli vakitler geçiriyorsunuz. Kısacası sabahtan akşama kadar bu teknolojilerle içli-dışlı oluyorsunuz. Şimdi kısaca bir durup düşünelim. Bu aletler bu işlevleri nasıl gerçekleştiriyor?

Hiç merak ettiniz mi? Nasıl oluyor da uzaktaki bir akrabanızın görüntüsünü anlık olarak görebiliyorsunuz? Bazılarınız, çevrenizden bu gibi teknolojilerin bilgisayarları programlama sonucu ortaya çıkmış olduğunu duymuş olabilir. Evet, gerçekten de bu teknolojilerin arkasında koskoca bir bilgisayar programlama kavramı yatmaktadır.

Bilgisayarları programlamak!

Ne kadar da sihirli bir cümle gibi geliyor insana! Bilgisayarları programlama, bilgisayarlarla harikalar yaratma ve dünyayı değiştirme fikri, herkesi etkilemiş olacak ki etrafımızda programlamayla uğraşan 7'den 70'e birçok insan bulunuyor. Hatta, bilgisayarları programlama fikri her geçen gün genç yaştaki insanları bile etkilemeye devam ediyor. Öyle ki, günümüzde 12 yaşındaki bir genci bile programlamayla uğraşırken görmek mümkün olabiliyor. Ülkemizde de, özellikle son yıllarda, sizler gibi genç arkadaşlarımız programlamaya ilgi duymaya başladı. Bu durum ülkemiz için gerçekten sevindirici bir gelişme.

Bazı insanlar bilgisayarları programlamanın ve teknolojik gelişmelerde bulunmanın sadece yetişkinler tarafından yapılacağını düşünüyor. Halbuki, dünyada programlamaya erken yaşta başlayıp teknolojik gelişmelere katkıda bulunan bir çok kişi bulunuyor. Aslında, bu insanların çoğunu tanıyorsunuz. Örnek vermek gerekirse, her gün ziyaret ettiğimiz bir site olan Facebook'un kurucusu **Mark Zuckerberg**, **Microsoft**'un kurucusu **Bill Gates** ve Apple'ın kurucusu **Steve Jobs** programlamaya çok erken yaşta başladıklarını söylüyorlar.

Bu insanların adlarını dünyaya duyurduklarını öğrenince, programlamaya erken yaşta başlamanın önemini daha iyi anlayabiliyoruz. Bu satırları okuyan siz değerli arkadaşlarım o kadar şanslısınız ki! Genç yaşta programlamaya başlayacak olmanız sizin de bu insanlardan biri olabilme şansınızı oldukça artırıyor. Eğer kararlı olursanız ve sıkı bir şekilde çalışırsanız, önünüzdeki tüm engelleri aşabilir ve adınızı tüm dünyaya duyurabilirsiniz.

İsterseniz biraz da programlamanın geleceği hakkında konuşalım.

10 sene önce bugünkü teknolojik gelişmeleri hayal bile edemezdik. Aynı şekilde, 10 sene sonra ortaya çıkacak teknolojik gelişmeleri hayal edebilmek de oldukça zor oluyor. Öyle ki, her gün kullandığımız telefon uygulamaları, bilgisayar oyunları vb. birçok program gün geçtikçe geliştirilmeye devam ediyor. Hal böyle olunca dünyada programcı ihtiyacı da her geçen gün hızla artmaya devam ediyor. Ülkemizde de bu durum öngörülmüş olacak ki, programcı ihtiyacını karşılamak, çocuklara ve gençlere yazılım öğretmek için gerçekten güzel adımlar atılıyor. Bu gelişmelerden faydalanarak, sizler de çiçeği burnunda programcılar olarak programcılığı ve yazılımı öğrenerek kendinizi geliştirebilir ve gelecekte güzel işler başarabilirsiniz. Buna tüm kalbimle inanıyorum.

Şimdi, gerekli motivasyona sahip olduğumuza göre koltuğumuza yaslanalım ve programcılığı kafamızda somutlaştırmaya çalışalım. Hazır mısınız?

Programlama Nedir?

Önceki konu başlığımızda programlama dünyasını keşfettik. Eğer programlamaya ilk adımı atmak için hazırsanız, kemerlerinizi sıkı bağlayın. Çünkü, çok uzun ve zevkli bir serüven için yola çıkıyoruz.

Şimdi kafamızda programlamayı biraz daha somutlaştırmaya çalışalım. Bilgisayarların ilk çıktığı zamanlardan beri insanlar komutlar vererek bilgisayarları programlamanın yollarını aradılar. Geçmişten günümüze programlama o kadar çok gelişti ki şu anda bilgisayarları istediğimiz gibi programlayıp her şeyi yaptırabiliyoruz. Peki, nedir bu programlama? Programlama veya yazılım en basit anlamıyla bilgisayarlara belirli komutlar vererek istediğimiz her şeyi bu komutlar sayesinde yaptırabilmektir. Bunu daha iyi anlamak için günlük hayattan basit bir benzerlik kurabiliriz. Örneğin; anneniz bir programcı, siz de bir bilgisayar olun. Anneniz sizi markete alışveriş için gönderdiğinde muhtemelen aşağıdaki şeyleri size söylüyor ve siz de harfiyen uyguluyorsunuz.

2

Değişkenler, İfadeler ve Deyimler

Bu Bölümde

Python'daki Değerler ve Türleri	14
Değişken Nedir? ve Nasıl Tanımlanır?	15
Değişkenlere Değer Atama ve İşlem Sırası	16
Değişken İsimleri	17
Matematiksel İfadeler	19
Tür Dönüşümleri	22
Neler Öğrendik?	24
Alıştırmalar	25

Evet, nihayet beklediğimiz an geldi.

Bu bölüm, Python'daki temel değerler, değişkenler, temel ifadeler vb. kavramları öğreneceğimiz bölüm olacak.

Artık Python'a başladığımız için bölüm sonundaki alıştırmaları anlayarak yaparsanız programlamaya iyice alışacak ve gelecekte programlama dünyasında adınızı duyurmak için gerekli temeli almış olacaksınız.

PYTHON'DAKİ DEĞERLER VE TÜRLERİ

Programlamanın artık ne olduğunu az da olsa öğrendiğimize göre, Python'daki değerleri ve türlerini öğrenmeye başlayabiliriz.

Birinci bölümde Python'daki `print()` fonksiyonunu kullanarak ekrana bir yazı yazdırmıştık. Bu yazı, gerçekte Python'da kullanılan bir veri türü olan string (karakter dizisi) veri türünden bir değerdir. Yani burada yazdırdığımız yazı, asıl adıyla karakter dizileri de denilebilir. Ancak Python'da kullanılan bir çok değer türü olduğu için bunları sırasıyla öğreneceğiz.

Python'da tamsayı ya da ondalık sayı kullanmak için bu türden değerleri kullanırız. Örnek vermek gerekirse 35, 3, 4 gibi değerler bizim için tamsayı (*integer*) değerlerimizdir. Aynı şekilde 3.1, 4.0 gibi değerler bizim için ondalık (*float*) değerlerimizdir. Python'daki bütün değerler ve türleri `print()` fonksiyonu kullanılarak ekrana yazdırılabilir. İsterseniz şimdi bu değerleri `print()` fonksiyonu yardımıyla etkileşimli kabuğumuzda görelim. İlk başta yazı yazmamızı sağlayan değerler olan karakter dizilerini (*string*) öğrenelim.

```
>>> print("Merhaba")
Merhaba
```

Karakter dizilerinin veya diğer adıyla yazıların, Python tarafından anlaşılması için tırnak (".....") içerisine alınması gerekir. Burada da biz tırnak işareti kullandık. İsterseniz bir fonksiyon kullanarak karakter dizilerinin (string) Python'daki türünü ekrana bastıralım. `type()` fonksiyonunun içine herhangi bir değer koyarsak, Python bu değerin bize hangi türden olduğunu söyleyecektir. Şimdi hep beraber görelim;

```
>>> type("Merhaba")
<class 'str'>
```

Yukarıdaki örnekte gördüğümüz gibi, Python bu değerin `'str'` yani karakter dizisi (string) olduğunu bize söyledi. Bundan sonra `type()` fonksiyonunu kullanarak değerlerimizin türlerini öğrenebiliriz. Şimdi hep beraber tamsayı (integer) değerlerini ve türünü ekrana bastıralım.

```
>>> print(23)
23
>>> type(23)
<class 'int'>
```

Burada gördük ki Python tamsayıların intege'rın kısaltması olan `int` türünde olduğunu söyledi. Örneklerimize devam edelim ve ondalıklı sayılara bakalım.

```
>>>print(3.5)
3.5
>>>type(3.5)
<class 'float'>
```

Python'daki ondalıklı sayılar aynı matematik derslerimizde gördüğümüz gibi ondalık kısmı nokta ile ayrılarak yazılıyor. Buradaki örnekte Python'daki ondalıklı sayıların türünün float (ondalıklı sayı) olduğunu öğrenmiş olduk.

Bu konuda Python'daki temel değerleri ve türlerini öğrendik. Aslında Python'da daha fazla türde değerlerimiz var ancak bu türleri derslerimizin ilerki kısımlarında öğreneceğiz.

Değişken Nedir? ve Nasıl Tanımlanır?

Bu konu başlığında da, Python'daki değişkenleri ve bu değişkenlerin nasıl tanımlandığını göreceğiz. Programlama dillerinde ve Python'da değişkenler, veri (değer) depolamamızı sağlayan birimlerdir. Yani, Python'da herhangi bir veri depolayan her şey değişken sayılabilir. Değişkenleri gerçek hayattan bir benzetme yaparak daha iyi anlayabiliriz. Örneğin, boş bir tabak düşünelim. İşte bu tabak bir değişkendir. Siz bu tabağın içine herhangi bir yemek koyduğunuz zaman bu tabak bir değişken, içindeki yemek de bir değer (veri) olmuş oluyor. Ayrıca, aynı tabağın içine başka bir yemek de koyabiliriz. Böylece değişkenimizin değeri değişmiş oluyor.

İşte Python'daki değişkenleri de bu şekilde düşünebiliriz. Değişkenimize bir isim veririz ve içine değer koyarız. Gerçek hayattaki örnekte olduğu gibi değişkenimizin değeri program boyunca değişebilir ve bunda hiç bir sıkıntı yoktur. Şimdi hep beraber değişkenler nasıl tanımlanır görmeye çalışalım. Etkileşimli kabuğumuzu açıyoruz ve başlıyoruz.

```
>>> isim = "Hasan Bayhan"
>>> pisayısı = 3.14
>>> mesafe = 60
```

Yukarıdaki örnekte görüldüğü gibi `isim`, `pisayısı`, `mesafe` adında 3 tane değişken tanımladık ve bu değişkenlere değerlerimizi verdik. Değişkenleri istediğimiz isimde tanımlayabilir ve = işaretiyle değerlerimizi atayabiliriz.

4

Fonksiyonlar

Bu Bölümde

Fonksiyon Nedir?	54
Fonksiyonların Tanımlanması	55
Fonksiyon Çağrıları	56
Fonksiyon Girdileri	58
Fonksiyon Çıktıları	60
Kullanıcıdan Bilgi Almak: input() Fonksiyonu	63
Ekrana Değer Bastırmak: print() Fonksiyonu	67
Birbirini Çağıran Fonksiyonlar	68
Neler Öğrendik?	70
Alıştırmalar	71

Şimdiye kadarki bölümleri iyi analiz edip kavrayabildiysek daha eğlenceli olan fonksiyonlar konusuna giriş yapabiliriz.

Bu bölümle birlikte artık fonksiyonları öğrenmeye başlayacağız.

FONKSİYON NEDİR?

Önceki bölümlerde farketmemiş olsak bile Python'daki bazı fonksiyonları kullanmıştık. Bunlara örnek olarak `print()` ve `type()` fonksiyonunu verebiliriz.

Bu fonksiyonlar Python'nun bize kullanmamız için önerdiği fonksiyonlardan sadece iki tanesi. Python'da bunun gibi yüzlerce hatta binlerce fonksiyon bulunmaktadır. Bu fonksiyonları zamanı geldikçe kullanacağız. Peki bu fonksiyonlar kavramı gerçek hayatta nerelerde karşımıza çıkıyor? İsterseniz bir giriş olarak gerçek hayattan örneklerle fonksiyonları anlamaya çalışalım.

Biliyorsunuz ki, günlük hayatta bir çok ev aleti, araç/gereç kullanıyoruz. Örneğin, katı meyve sıkacağı bunlardan yalnızca bir tanesidir. Katı meyve sıkacağına istediğimiz meyveleri veriyoruz ve bu alet bir takım işlemler yaparak sonunda meyve suyumuzu bize veriyor. Biz de bu meyve suyunu afiyetle içiyoruz. Bunun gibi bir çok örnek düşünebilirsiniz. Örneğin, mikrodalga fırına verdiğimiz yiyecekler mikrodalga fırın tarafından işlemden geçerek bize sıcak yemek olarak geri dönüyor. Bu örneklerde gördüğümüz şeyler aslında gerçek hayatta kullanılan fonksiyonlara birer örnektir. Katı meyve sıkacağının işlevi (fonksiyonu) aslında meyve suyu hazırlamaktır. Mikrodalga fırının da işlevi (fonksiyonu) yemekleri ısıtmaktır. Python'da kullanılan fonksiyonlar da aslında bu şekilde kullanılıyor. Örneğin, `print()` fonksiyonu ve katı meyve sıkacağı örneğini beraber düşünelim. Biliyorsunuz `print()` fonksiyonu şu şekilde kullanılabiliyordu.

```
>>> print("Ben bir programcıyım.")

Ben bir programcıyım.
```

Biz burada ne yaptık? `print()` fonksiyonuna bir string (karakter dizisi veya yazı) verdik ve `print()` fonksiyonu kendi içinde bir takım işler yaparak bize sonuç olarak ekrana yazı yazdırdı. Aynı şekilde katı meyve sıkacağına tıpkı `print()` fonksiyonuna string verdiğimiz gibi meyveler verdik. Katı meyve sıkacağı da `print()` fonksiyonunun ekrana yazı yazdırması gibi sonuç olarak bize meyve suyu verdi.

Daha önce söylediğimiz gibi Python'da direk olarak kullanabileceğimiz yüzlerce fonksiyon bulunmaktadır. Bu fonksiyonların hepsi içine verdiğimiz değerlere göre belli başlı işlemler gerçekleştirerek, yapmak istediğimiz şeyleri hızlı bir şekilde yapmaktadır. Örnek olarak `print()` fonksiyonuyla ekrana bir değer yazdırmaya çalışırken, değerimizi `print()` fonksiyonumuzun içine gönderiyoruz ve değerimiz ekranda görünüyor.

Kullanıcıdan bilgi ve girdi almak bu şekilde yapılmaktadır.Şimdi de print() fonksiyonunu biraz daha yakından inceleyelim.

Ekrana Değer Bastırmak: print() Fonksiyonu

Şimdi de `print` fonksiyonunu biraz daha yakından incelemeye çalışacağız. Bildiğiniz gibi `print()` fonksiyonu ekrana yazı, tamsayı, ondalıklı sayı gibi veri tiplerini bastırmamızı sağlıyordu. Ancak biz şimdiye kadar sadece `print` fonksiyonuna bir tane parametre gönderiyorduk. İsterseniz `print()` fonksiyonuna bir kaç tane parametre gönderelim ve sonucuna bakalım. Etkileşimli kabukta işlemlerimizi gerçekleştirebiliriz.

```
>>> print("Mustafa","Murat","Coşkun")
Mustafa Murat Coşkun
```

Burada biz `print()` fonksiyonumuza 3 tane yazı gönderdik ve `print` fonksiyonu bu yazıları ekrana bastırmış oldu. Buradan anlayacağımız gibi, `print()` fonksiyonunun içine istediğimiz kadar parametre gönderebiliyoruz. Başka bir örnek daha yapalım.

```
>>> print("Mustafa",23,"Yazılımcı",3.34)
Mustafa 23 Yazılımcı 3.34
```

Gördüğümüz gibi tam sayı, ondalıklı sayı ve yazıları da `print` fonksiyonumuza beraber gönderebiliyoruz. Ancak burada dikkat ettiyseniz, `print` fonksiyonu bastırdığı her parametreden sonra kendisi boşluk koyuyor. Eğer siz `print` fonksiyonunun boşluk koymasını istemiyorsanız veya parametreler arası başka karakterler yerleştirmek istiyorsanız bunu da şu şekilde yapabiliyorsunuz.

```
>>> print("Mustafa","Murat","Coşkun",sep = "") # boşluksuz
MustafaMuratCoşkun
>>> print("19","09","1993",sep = "/")
19/09/1993
```

Burada, `sep` isimli özel parametrenin ne işe yaradığını tahmin etmiş olmalısınız.Bu parametreye verilen değer parametreler arasına koyulmaktadır. İkinci `print()` fonksiyonu örneğinde, `sep` parametresi `"/"` değerine sahip oluyor ve parametreler arası `"/"` karakteri konuluyor.

5

Döngü Mantığı

Bu Bölümde

Giriş	78
While Döngüsü	78
Break ve Continue	82
Neler Öğrendik?	85
Alıştırmalar	86

Önceki bölümde, Python ve diğer programlama dillerinde de çok önemli bir kavram olan fonksiyonlar konusunu öğrendik.

Bu bölümde Python'daki döngü yapılarını görecek ve döngüleri anlamaya çalışacağız.

GİRİŞ

Önceki bölümlerde yazdığımız her program bir defa çalıştırıldıktan sonra sona eriyor ve programı tekrar çalıştırmamız gerekiyordu. Artık bu bölümde döngüleri öğrenecek ve bu dertten kurtulacağız. Şimdi biraz Python'da döngüler ne anlama geliyor hep beraber inceleyelim.

Python'daki döngüler yapacağımız bir işlemin belli koşul durumunda sürekli olarak tekrarlanmasını sağlayan yapılardır. Eğer belli işlemleri tekrar tekrar çalıştırmak istiyorsak, döngü yapılarını kullanmamız mantıklı olacaktır. Döngü yapılarıyla farkında olmasak bile gerçek hayatta kullanılan bilgisayar programlarında karşılaşıyoruz. Örneğin, bir web sitesinde kullanıcı girişi yaptığımız zaman eğer kullanıcı adımızı veya parolamızı yanlış girersek web sitesi bize doğru giriş yapana kadar kullanıcı adımızı ya da parolamızı tekrar girmemizi istiyor. Burada web sitesinde çalışan program, doğru giriş yapana kadar işlemleri tekrarlıyor ve aslında bir döngü yapısı kullanıyor. Bu döngünün bitmesi ancak doğru giriş yapılınca sona eriyor. Burada koşul durumu sonlandığı zaman döngümüz sona eriyor gibi düşünebilirsiniz.

Başka bir örnek vermek gerekirse, günlük yaşamda kullandığımız ATM'lerde aynı şekilde döngü yapılarını kullanmaktadır. Banka kartımızı ATM cihazına takıyoruz ve program çalışmaya başlıyor. Aynı döngünün içinde para yatırabiliyoruz, para çekebiliyoruz ve diğer benzer işlemleri gerçekleştirebiliyoruz. ATM programının sona ermesi ise **Kart İade** butonuna basınca olacaktır. Bunlar gibi bir çok örnek düşünebilirsiniz.

Söylediğimiz gibi yukarıdaki örnekleri Python ile yapabilmemiz için Python'daki döngüleri kullanmamız gerekmektedir. Python bizlere kullanmamız için 2 tane döngü yapısı sunmaktadır. Bu döngü yapıları, `while` ve `for` döngüleridir. Biz bu bölümümüzde sadece `while` döngülerini öğrenmeye çalışacağız. `for` döngülerini ise 6. bölümde anlatmamız daha doğru olacaktır. Çünkü, `for` döngüleri sonraki bölümlerde detaylıca öğreneceğimiz `liste`, `string`, `demet`, `sözlük` gibi veri türlerinde daha çok kullanılmaktadır. Bu yapıları öğrendiğimiz zaman `for` döngülerini daha rahat bir şekilde kavrayacaksınız.

WHILE DÖNGÜSÜ

Döngü mantığını biraz anladıysanız `while` döngülerini anlamanız zor olmayacaktır. Döngü mantığını anlatırken döngülerin sadece belirli koşullar sağlanınca çalıştığını söylemiştik. Bu mantığıyla döngüler aslında `if` yapılarına benzemektedir. Şimdi isterseniz `while` döngülerinin nasıl yazıldığına kısa bir giriş yapalım.

```
while (koşul cümlesi):
(Tab-Girinti) İşlem 1
(Tab-Girinti) İşlem 2
(Tab-Girinti) İşlem 3
(Tab-Girinti) İşlem 4
       //               //
```

while döngü yapısının yazımı yukarıdaki gibi olmaktadır. Peki, Python'daki döngü yapıları nasıl çalışıyor ve nasıl kullanılıyor? Eğer bir döngü yazacak isek, ilk olarak bir koşul cümlesi belirlememiz gerekmektedir. Döngümüz bu koşul cümlesi doğru (True) olduğu sürece çalışacaktır. Döngümüzün sona ermesi ise ancak ve ancak bu koşul cümlesi yanlış (False) olduğu durumda olacaktır. Şimdi döngümüzün çalışma prensiplerini bir örnekle açıklamaya çalışalım. Örneğin, sadece 4 işlem gerçekleştiren bir while döngümüz olsun.

```
while (koşul cümlesi):
        İşlem 1
        İşlem 2
        İşlem 3
        İşlem 4
```

Programımızın başında while döngüsü ilk başta içindeki koşul cümlesinin doğru olup olmadığını kontrol etmektedir. Eğer koşul cümlesi True ise, while döngüsü kendi bloğunu çalıştırmaktadır. İşlem1, İşlem2, İşlem3, İşlem4 işlemleri sırasıyla yapılıyor ve while döngüsü bloğunu bitiriyor. Ancak burada while döngüsü daha önce görmediğimiz bir işlem yapıyor ve tekrardan koşul cümlesini kontrol ediyor. Eğer, koşul cümlesi tekrar True ise while döngüsü bloğunu tekrar çalıştırıyor. İşte while döngüsünün ve döngülerin asıl mantığı böyle olmaktadır. while döngüsünün içinde koşul durumu True olmaya devam ettiği sürece sürekli olarak while bloğu çalıştırılmaktadır. Peki, bu döngü ne zaman sonra ermektedir? while döngüsünün sona ermesi ancak ve ancak koşul durumunun False olduğu durumda sona erecektir. Şimdi kolay bir while döngüsü yazalım. Buradaki mantığı iyice kavramaya çalışalım.

```
i = 0  # Bu değişken while döngüsünün koşul durumu için kullanılacak.
#Döngümüzü yazalım.
while (i < 4):
        print("i:",i)
        i = i +1
```

7

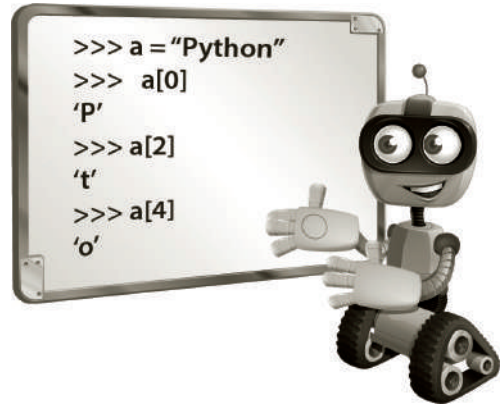
Karakter Dizileri

Bu Bölümde

String Nedir? 116
String'ler ve Listeler Arasındaki İlişki 117
String'lerdeki Özel Karakterler 119
String'lerin Fonksiyonları 120
Neler Öğrendik? 126
Alıştırmalar 127

Bu bölümde, Python'daki karakter dizilerini (yazı veya string) daha yakından incelemeye başlayacağız. Önceki bölümde belirttiğimiz gibi, karakter dizilerinin fonksiyonları ve üzerinde yapılan işlemler listelere çok benzemektedir.

Bu yüzden, bu bölümümüzde sıkıntı çekmeden karakter dizilerini anlayacağınızı düşünüyorum. Ayrıca, kitabın geri kalan kısmında karakter dizilerinden karakter dizilerinin İngilizce karşılığı olan string olarak bahsedeceğiz. Hemen başlayalım.



STRING NEDİR?

Kitabımızın 2. bölümünde veri tiplerinden bahsederken string (karakter dizileri) lerden ve tanımlamalarından bahsetmiştik. Bu bölümle birlikte, string'leri daha detaylı incelemeye başlayacağız.

String'ler ya da karakter dizileri neredeyse her programlama dilinde bulunan bir veri tipidir. Bu veri tipine karakter dizileri denilmesinin sebebi, karakterlerin bir araya gelmesinden oluşmasıdır. Yani, biz bir tane string tanımladığımız zaman, içine verdiğimiz değerin karakterlerden oluştuğunu söyleyebiliriz. Örneğin, bir tane string tanımlayalım.

```
>>> a = "Murat"
```

Buradaki a isimli string'imiz aslında sırasıyla `M`, `u`, `r`, `a`, `t` karakterlerinin bir araya gelmesinden oluşmaktadır. Buradaki durum aslında kafamızda şöyle bir benzetme yapmamıza yol açıyor: **Listelere ne kadar da benziyor**. Evet, gerçekten de Python'da string'ler birçok açıdan listelere çok benzemektedir. Bir sonraki başlığımızda bunu daha rahat göreceğiz. Ancak bu konu başlığını bitirmeden, string'lerin farklı şekillerde tanımlanmalarını öğrenmekte fayda var. String'leri şimdiye kadar sadece çift tırnak arasına karakterlerimizi yazarak tanımladık. Bunun gibi, string'lerimizi tek ve üç tırnak yardımıyla da tanımlayabiliyoruz. Aşağıdaki gösterimlerde, string'lerin farklı şekillerde tanımlanmalarını görebilirsiniz.

```
>>> a = "Murat"
>>> a = 'Murat'
>>> a = """Murat"""
>>> a = '''murat'''
```

Buradaki tanımlamalardan herhangi birisini kullanmakta özgürsünüz.Şimdi hep beraber, string'lerle listeler arasındaki ilişkiyi öğrenmeye çalışalım.

String'ler ve Listeler Arasındaki İlişki

Önceki bölümümüzde listeler konusunu işlerken listelerin indekslenmesini, parçalanmasını, listelerde `for` döngüsünü,`in` operatörünü öğrenmiştik. Aslında Python'da bu işlemler aynı şekilde string'ler üzerinde de geçerlidir. İsterseniz ilk önce Python'da string'lerin nasıl indekslendiğini görelim.

```
>>> a = "Python"
>>>   a[0]
'P'
>>> a[2]
't'
>>> a[4]
'o'
```

Hiçbir fark yok değil mi? Burada gördüğümüz gibi, string'lerin indekslenmesi de aynı listelerin indekslenmesi gibi olmaktadır. Şimdi de tanımladığımız `a` değişkeni üzerinde `len()` fonksiyonunu uygulayalım.

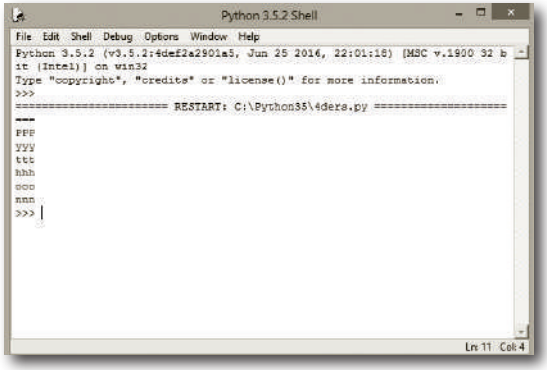
```
>>> a = "Python"
>>> len(a)
6
```

`len()` fonksiyonu stringlerin üzerinde uygulandığı zaman örnekteki gibi string'lerin uzunluğunu bize söylemektedir. Şimdi de bir tane dosya açalım ve `for` döngüyle string'ler üzerinde bir program yazalım.

```
# Bir tane string tanımlayalım
yazı = "Python"

for harf in yazı:
  print( 3  * harf)
```

Burada `for` döngüsünün `"Python"` yazısının üzerinde yaptığı işlem oldukça basittir. `for` döngüsü her bir harf üzerinde gezinerek ekrana her harfin 3 ile çarpımı ekrana yazdırmaktadır.



Şimdi de listelerde yaptığımız gibi string'lerimizi parçalayalım. Elimizde bir "Python Programlama" isminde bir stringimiz olsun ve bu string üzerinde parçalama işlemlerimizi gerçekleştirelim.

```
>>> yazı = "Python Programlama"
>>> yazı[5:]
'n Programlama'
>>> yazı[7:11]
'Prog'
>>> yazı[::-1]
'amalmargorP nohtyP'
>>> yazı[::2]
'Pto rgalm'
```

Buradaki örnekte görüldüğü gibi, string'lerin parçalama işlemlerinde mantık, listelerin parçalanmasıyla birebir olarak aynıdır.

Şimdi de son olarak `in` operatörünün stringler üzerinde uygulanmasını görelim.

```
>>> a = “Python”
>>> 'P' in a
True
>>> 'y' in a
True
>>> 's' in a
False
```