
buzdağı

İçindekiler

Programlama hakkında "gelişim odaklı bir zihniyete" sahip olmak önemlidir. Diğer bir deyişle, insanların programlama becerilerini pratik yaparak geliştirdiklerini anlamak önemlidir. Onlar programcı olarak doğmadılar ve programlamada şu anda beceriksiz olmak hiçbir zaman uzman olamayacağınızın anlamına gelmez.

Programlama Yaratıcı Bir Etkinliktir

Programlama resim yapmak, yazı yazmak, örgü örmek veya LEGO kaleleri inşa etmek gibi yaratıcı bir iştir. Boş bir tuvale resim yapmak gibi yazılımında pek çok kısıtlamaları fakat sonsuz imkânları mevcuttur.

Programlama ve diğer etkinlikler arasındaki fark, programlama yaparken bütün ham maddenin bilgisayarınızda mevcut olmasıdır. İlave olarak tuval, boya, film, iplik, LEGO tuğlaları veya elektronik bileşenler satın almak zorunda değilsiniz. On yıllık bir bilgisayar, program yazmak için yeterince güçlüdür. Programınız bir kere yazıldı mı sonsuz kere kusursuz bir şekilde kopyalanabilir. Örülmüş bir hırka bir seferde sadece bir kişi tarafından giyilebilir ancak kullanışlı bir program internet üzerinden bütün dünyayla kolaylıkla paylaşılabilir.

Bu Kitap Hakkında

Bu kitabın ilk kısmı temel Python programlama kavramlarını, ikinci kısmı ise bilgisayarınızın otomatikleştirmesini sağlayabileceğiniz çeşitli görevleri ele almaktadır. İkinci kısımdaki her bir bölüm, çalışmanız için proje programları içerir.

İşte her bir bölümde bulacaklarınızın kısa bir özeti.

Kısım I: Python ile Programlamanın Temelleri

Bölüm 1: Python'un Temelleri İfadeleri, en temel Python komutu türünü ve kodla deneme yapmak için Python etkileşimli kabuk yazılımının (ÇN: Python Interactive shell software) nasıl kullanılacağını ele almaktadır.

Bölüm 2: Akış Kontrolü Kodunuzun farklı koşullara akıllıca yanıt verebilmesi için programların hangi komutların yürütüleceğine nasıl karar vereceğini açıklamaktadır.

Bölüm 3: Fonksiyonlar Kodunuzu daha yönetilebilir parçalar hâlinde düzenleyebilmeniz için kendi fonksiyonlarınızı nasıl tanımlayacağınız konusunda size talimat vermektedir.

Bölüm 4: Listeler Liste veri tipini tanıtmakta ve verilerin nasıl düzenleneceğini açıklamaktadır.

Bölüm 5: Sözlükler ve Veriyi Yapılandırmak Sözlük veri tipini tanıtmakta ve size verileri düzenlemenin daha güçlü yollarını göstermektedir.

Bölüm 6: Dizgileri Manipüle Etmek Metin verisiyle (Python'da dizgiler olarak adlandırılır) çalışmayı ele almaktadır.

Kısım II: Görevleri Otomatikleştirmek

Bölüm 7: Düzenli İfadelerle Örüntü Eşleştirme Python'un dizgileri nasıl işleyebileceği ve düzenli ifadeler ile nasıl metin örüntüleri arayacağını ele almaktadır.

Bölüm 8: Girdi Doğrulama Kullanıcı verilerinin programın geri kalanında hatalara neden olmayacak bir biçimde ulaşmasını sağlayarak, programınızın, bir kullanıcının verdiği bilgileri nasıl doğrulayabileceğini açıklamaktadır.

Bölüm 9: Dosyaları Okumak ve Dosyalara Yazmak Programınızın sabit diskinizdeki metin dosyalarını nasıl okuyabileceğini ve bilgileri sabit diskinizdeki dosyalara nasıl kaydedebileceğini anlatmaktadır.

Bölüm 10: Dosyaları Düzenlemek Python'un çok sayıda dosyayı nasıl bir insandan çok daha hızlı bir biçimde kopyalayabileceğini, taşıyabileceğini, yeniden adlandırabileceğini ve silebileceğini göstermektedir. Aynı zamanda dosyaları sıkıştırmayı ve sıkıştırılmış dosyaları açmayı da anlatmaktadır.

Bölüm 11: Hata Ayıklama Python'un çeşitli hata bulma ve hata giderme araçlarının nasıl kullanılacağını göstermektedir.

Bölüm 12: Web Kazıma Web sayfalarını otomatik olarak indirebilen ve bilgi için ayrıştırabilen programların nasıl yazılacağını göstermektedir. Buna *web kazıma* denir.

Bölüm 13: Excel Elektronik Tablolarıyla Çalışmak Okumak zorunda kalmamanız için Excel elektronik tablolarını programlı olarak manipüle etmeyi kapsamaktadır. Bu, analiz etmeniz gereken belge sayısı yüzlerce veya binlerce olduğunda faydalıdır.

Bölüm 14: Google Tablolarıyla Çalışmak Popüler bir web tabanlı elektronik tablo uygulaması olan Google Sheets'in Python kullanılarak nasıl okunacağı ve güncelleneceğini ele almaktadır.

Bölüm 15: PDF ve Word Belgeleriyle Çalışmak Word ve PDF belgelerinin programlı olarak okunmasını ele almaktadır.

Bölüm 16: CSV Dosyaları ve JSON Verileriyle Çalışmak Belgelerin programlı olarak nasıl manipüle edilebileceğini açıklamaya devam etmektedir. Şimdi ise CSV ve JSON dosyalarından bahsetmektedir.

Bölüm 17: Zaman Tutma, Görevleri Zamanlamak ve Programları Başlatmak Python programlarının zaman ve tarihleri nasıl ele aldığını ve belirli zamanlarda görevleri yerine getirmesi için bilgisayarınızı nasıl zamanlayacağınızı anlatmaktadır. Ayrıca Python programlarınızın Python dışı programları nasıl başlatabileceğini göstermektedir.

Bölüm 18: E-posta ve Metin Mesajları Göndermek Sizin adınıza e-posta ve metin mesajları gönderebilen programların nasıl yazılacağını anlatmaktadır.

Bölüm 19. Görüntüleri Manipüle Etme JPEG veya PNG dosyaları gibi görüntülerin programlı olarak nasıl manipüle edilebileceğini açıklamaktadır.

Bölüm 20: GUI Otomasyonuyla Klavye ve Fareyi Kontrol Etmek Tıklamaları ve tuşa basmaları otomatikleştirmek için fare ve klavyenin programlı olarak nasıl kontrol edileceğini anlatmaktadır.

Ek A: Üçüncü Parti Modülleri Kurmak Kullanışlı ilave modüllerle Python'u nasıl genişleteceğinizi göstermektedir.

Ek B: Programları Çalıştırmak Python programlarınızı kod düzenleyicinin dışında Windows, macOS ve Linux üzerinde nasıl çalıştıracağınızı göstermektedir.

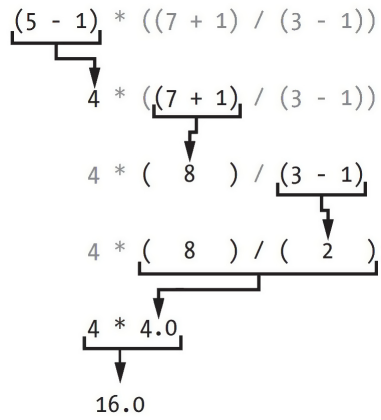
Ek C: Alıştırma Sorularının Yanıtları Her bölümün sonundaki alıştırma sorularına cevaplar ve bazı ek bağlamlar sağlamaktadır.

Python'u İndirmek ve Kurmak

https://python.org/downloads/ adresinden Python'u Windows, macOS ve Ubuntu için ücretsiz olarak indirebilirsiniz. Web sitesinin indirme sayfasından en son versiyonu indirirseniz, bu kitaptaki bütün programların çalışması gerekir.

Uyarı

Python 3 versiyonunu indirdiğinizden emin olunuz (örneğin 3.8.0). Bu kitaptaki programlar Python 3'te çalışacak şekilde yazılmıştır ve Python 2'de düzgün çalışmayabilir



Nasıl ki dil bilgisi kuralları iletişim kurmamıza yardımcı oluyorsa aynı şekilde ifadeler oluşturmak için operatörleri ve değerleri bir araya getiren bu kurallar da bir programlama dili olarak Python'un temel parçasıdır. İşte bir örnek:

Bu dil bilgisel olarak doğru bir Türkçe cümledir.

Bu dil bilgisel olarak cümledir değil Türkçe doğru bir.

İkinci satırı ayrıştırması zordur çünkü Türkçe kurallarına uymamaktadır. Benzer şekilde eğer kötü bir Python komutu girerseniz Python onu anlamayacak ve aşağıda gösterildiği gibi bir `SyntaxError` (ÇN: Sözdizim hatası) hata mesajı görüntüleyecektir:

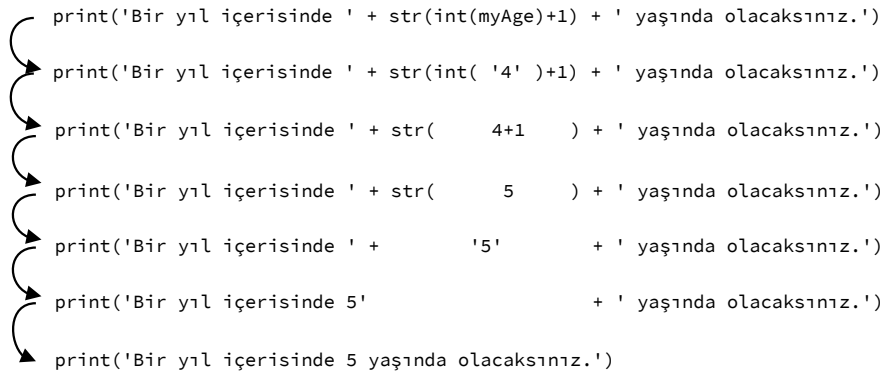
```
>>> 5 +
  File "<stdin>", line 1
    5 +
      ^
SyntaxError: invalid syntax
>>> 42 + 5 + * 2
  File "<stdin>", line 1
    42 + 5 + * 2
             ^
SyntaxError: invalid syntax
```

Etkileşimli kabuğa girmek suretiyle her zaman bir komutun çalışıp çalışmadığını test edebilirsiniz. Bilgisayarı bozarım diye endişelenmeyiniz: Olabileceklerin en kötüsü Python'un bir hata mesajı ile yanıt vermesidir. Profesyonel yazılım geliştiriciler kod yazarken sürekli hata mesajları alırlar.

`myAge` değişkeni `input()`'dan döndürülen değeri içerir. `input()` fonksiyonu her zaman bir dizgi döndürdüğünden (kullanıcı sayı yazmış olsa bile), `myAge`'deki dizginin tam sayı değerini döndürmek için `int(myAge)` kodunu kullanabilirsiniz. Bu tam sayı değeri daha sonra `int(myAge) + 1` ifadesinde `1`'e eklenir.

Bu toplama işleminin sonucu daha sonra `str()` fonksiyonuna gönderilir: `str(int(myAge) + 1)`. Döndürülen dizgi değeri sonrasında tek bir büyük dizgi değeri olarak değerlendirilmek üzere `'Bir yıl içerisinde '` ve `'yaşında olacaksınız.'` dizgileriyle birleştirilir.

Diyelim ki kullanıcı `myAge` için `'4'` dizgisini giriyor. `'4'` dizgisi bir tam sayıya dönüştürülür, böylece bu tam sayıya bir ekleyebilirsiniz. Sonuç `5`'tir. `str()` fonksiyonu sonucu bir dizgiye geri dönüştürür, böylece bu dizgiyi son mesajı oluşturmak için ikinci bir dizgi olan `'yaşında olacaksınız.'` ile birleştirebilirsiniz. Bu hesaplama adımları aşağıdakine benzer olur:



1.7 Özet

İfadeleri hesap makinesiyle hesaplayabilir veya kelime işlemcisinde dizgi birleştirmeleri girebilirsiniz.

Metin kopyalayıp yapıştırmak suretiyle bile kolayca dizgi çoğaltma yapabilirsiniz. Ancak ifadeler ve bunların bileşen değerleri –operatörler, değişkenler ve fonksiyon çağrıları– programları oluşturan temel yapı taşlarıdır. Bu elemanları nasıl ele alacağınızı öğrendikten sonra, Python'a sizin için büyük miktarda veri üzerinde çalışması talimatını verebileceksiniz.

Farklı operatör tiplerini (matematiksel işlemler için `+`, `-`, `*`, `/`, `//`, `%` ve `**`; dizgi işlemleri için `+` ve `*`) ve bu bölümde tanıtılan üç veri tipini (tam sayılar, kayan-noktalı sayılar ve dizgiler) hatırlamak iyidir.

2.7.6 `continue` Deyimleri

`break` deyimleri gibi `continue` (ÇN: devam et) deyimleri de döngülerin içerisinde kullanılır. Program çalışması bir `continue` deyimine ulaştığında, program çalışması hemen döngünün başına geri döner ve döngünün koşulunu yeniden değerlendirir. (Program çalışması döngünün sonuna ulaştığında da bu gerçekleşir.)

Kullanıcı ismi ve şifre isteyen bir program yazmak için continue deyimini kullanalım. Aşağıdaki kodu yeni bir dosya düzenleyici penceresine giriniz ve programı *swordfish.py* olarak kaydediniz.

SONSUZ DÖNGÜ TUZAĞINA MI DÜŞTÜNÜZ?

Sonsuz bir döngüde takılıp kalmasına neden olan bir hata içeren program çalıştırırsanız, `CTRL-C`'ye basınız veya IDLE'ın menüsünden Shell > Restart Shell'i (ÇN: Kabuk, Kabuğu Yeniden Başlat) seçiniz. Bu, programınıza bir `KeyboardInterrupt` (ÇN: Klavye kesintisi) hatası gönderir ve programın hemen durmasına sebebiyet verir. Dosya düzenleyicide basit bir sonsuz döngü oluşturarak programı durdurmayı deneyiniz ve programı *infiniteLoop.py* olarak kaydediniz.

```
while True:
  print('Merhaba, dünya!')
```

Bu programı çalıştırdığınızda ekrana sonsuza kadar Merhaba, dünya! yazdıracaktır çünkü `while` deyiminin koşulu her zaman `True`'dur. Ek olarak, sonsuz bir döngüde takılıp kalmasa bile sadece programınızı hemen sonlandırmak isterseniz `CTRL-C` kullanışlıdır.

```
while True:
  print('Kimsiniz?')
  name = input()
  if name != 'Joe': ❶
    continue  ❷
  print('Merhaba Joe. Şifre nedir? (bir balık)')
  password = input() ❸
  if password == 'swordfish':
    break ❹
print('Erişim verildi..')  ❺
```

Kullanıcı Joe dışında bir isim girerse ❶, `continue` deyimi ❷ programın çalışmasının döngünün başlangıcına geri dönmesine neden olur. Program koşulu yeniden değerlendirdiğinde, koşul yalnızca `True` değeri olduğundan, çalışma her zaman döngüye girer. Kullanıcı bu `if` deyimini geçtikten sonra bir şifre istenir ❸. Girilen şifre `swordfish` ise `break` deyimi çalıştırılır ❹ ve program

çalışması ekrana Erişim verildi yazmak için `while` döngüsünün dışına atlar ❺. Aksi takdirde, program çalışması `while` döngüsünün sonuna kadar devam eder ve daha sonra burada döngünün başına geri döner. Bu programın akış diyagramı için Şekil 2.12'ye bakınız.

"DOĞRUMSU" VE "YANLIŞIMSI" DEĞERLER

Koşullar, `True` ve `False`'a eş değer olan diğer veri türlerindeki bazı değerleri dikkate alacaktır. Koşullarda kullanıldığında `0`, `0.0` ve `''`(boş dizgi) `False` olarak kabul edilir. Diğer bütün değerler ise `True` olarak kabul edilir. Örneğin aşağıdaki programa bakınız:

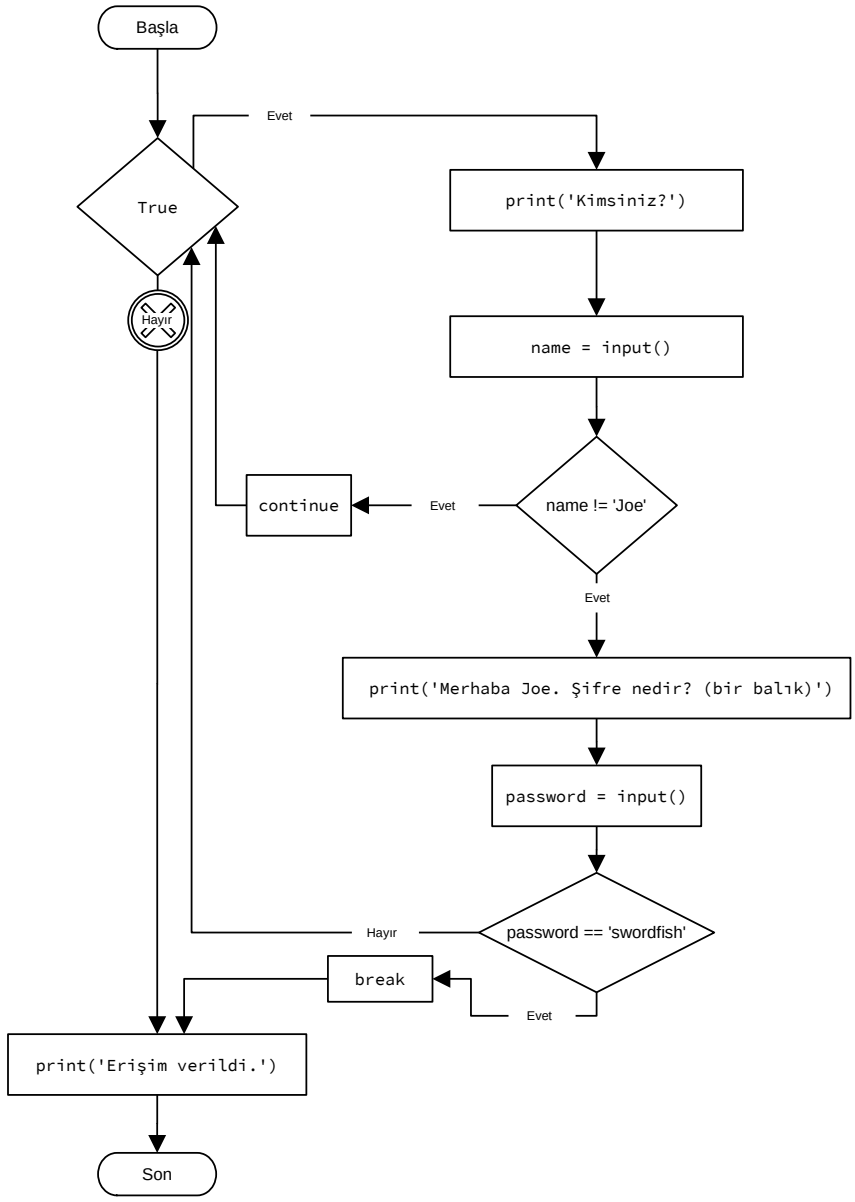
```
name = ''
while not name: ❶
  print('İsminizi giriniz:')
  name = input()
print('Kaç tane misafiriniz olacak?')
numOfGuests = int(input())
if numOfGuests:  ❷
  print('Bütün misafirleriniz için yeterince
         yeriniz olduğundan emin olunuz.') ❸
print('Tamamdır.')
```

Bu programın çalışmasını ***https://autbor.com/howmanyguests/*** adresinde görebilirsiniz. Kullanıcı `name` için boş bir dizgi girerse, `while` deyiminin koşulu `True` olur ❶ ve program isim sormaya devam eder. `numOfGuests`'in değeri `0` değilse ❷ koşul `True` olarak kabul edilir ve program kullanıcı için hatırlatıcı bir mesajı ekrana yazdırır ❸.

`not name` yerine, `not name != ''` ve `numOfGuests` yerine, `numOfGuests!=0` girebilirdiniz ancak doğrumsu ve yanlışımsı değerler kullanmak kodunuzu okunması daha kolay yapabilir.

Bu programı çalıştırınız ve programa birtakım girdiler veriniz. Joe olduğunuzu iddia edene dek program şifre sormamalıdır ve doğru şifreyi girdiğinizde program çıkış yapmalıdır.

```
Kimsiniz?
Ben iyiyim, teşekkürler. Kimsiniz?
Kimsiniz?
Joe
Merhaba Joe. Şifre nedir? (bir balık)
Mary
Kimsiniz?
Joe
Merhaba Joe. Şifre nedir? (bir balık)
swordfish
Erişim verildi.
```



Şekil 2.12: *swordfish.py için akış diyagramı. Döngü koşulu her zaman* `True` *olduğundan, X yolu mantıksal olarak asla gerçekleşmeyecektir*

Bu programın çalışmasını ***https://autbor.com/hellojoe/*** adresinde görebilirsiniz.

Bununla birlikte, yeni satır karakterini farklı bir dizgiyle değiştirmek için `end` anahtar kelime argümanını ayarlayabilirsiniz. Örneğin kod şu olsaydı:

```
print('Hello', end='')
print('World')
```

çıktı şu şekilde olurdu:

```
HelloWorld
```

Çıktı tek satıra yazılır çünkü 'Hello'dan sonra ekrana yazdırılan yeni bir satır yoktur. Bunun yerine, boş dizgi ekrana yazdırılır. Bu, her `print()` fonksiyon çağrısının sonuna eklenen yeni satırı devre dışı bırakmanız gerektiğinde kullanışlıdır.

Benzer şekilde `print()`'e birden çok dizgi değeri gönderdiğinizde fonksiyon bu değerleri otomatik olarak tek bir boşluk ile ayıracaktır. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> print('cats', 'dogs', 'mice')
cats dogs mice
```

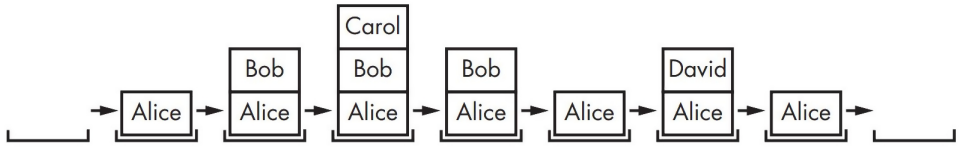
Ancak `sep` argümanına farklı bir dizgi göndererek varsayılan ayırıcı dizgiyi değiştirebilirsiniz. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> print('cats', 'dogs', 'mice', sep=',')
cats,dogs,mice
```

Yazdığınız fonksiyonlara da anahtar kelime argümanlar ekleyebilirsiniz. Ancak öncelikle sonraki iki bölümde liste ve sözlük veri tiplerini öğrenmeniz gerekmektedir. Şimdilik, bazı fonksiyonların, fonksiyon çağrıldığında belirtilebilecek isteğe bağlı anahtar sözcük argümanlarına sahip olduğunu biliniz.

3.5 Çağırma Yığını

Birileriyle konudan konuya atlayarak konuşma yaptığınızı düşünün. Arkadaşınız Alice hakkında konuşuyorsunuz, bu da size iş arkadaşınız Bob ile ilgili hikâyeyi hatırlatıyor ancak öncelikle kuzeniniz Carol ile ilgili bir şeyler açıklamanız gerekiyor. Carol ile ilgili hikâyeyi bitiriyorsunuz ve Bob ile ilgili olan konuşmanıza geri dönüyorsunuz ve Bob ile ilgili hikâyenizi bitirdiğinizde Alice ile ilgili olan konuşmanıza geri dönüyorsunuz. Ancak kardeşiniz David size hatırlatılıyor, dolayısıyla onunla ilgili bir hikâye anlatıyorsunuz ve Alice hakkındaki esas hikâyeyi bitirmeye geri dönüyorsunuz. Konuşmanız Şekil 3.1'deki gibi *yığın* benzeri bir yapıyı takip etmiştir. Konuşma yığın benzeridir çünkü mevcut konu her zaman yığının en üstündedir.



Şekil 3.1: *Konudan konuya atladığınız konuşmanızın yığını*

Konudan konuya atladığımız konuşmamıza benzer bir şekilde, bir fonksiyonu çağırmak, program çalışmasını fonksiyonun en üstüne doğru tek yönlü bir yolculuğa çıkarmaz. Python, fonksiyonu çağıran kod satırını hatırlayacaktır, böylece program çalışması bir `return` deyimine rastladığında oraya geri dönebilecektir. Eğer bu orijinal fonksiyon diğer fonksiyonları çağırdıysa program çalışması, orijinal fonksiyon çağrısından dönmeden önce bu fonksiyon çağrılarına geri dönecektir.

Bir dosya düzenleyici penceresi açınız ve aşağıdaki kodu giriniz ve bu kodu *abcdCallStack.py* olarak kaydediniz:

```
def a():
  print('a() başlar')
  b() ❶
  d() ❷
  print('a() döndürür')

def b():
  print('b() başlar')
  c() ❸
  print('b() döndürür')

def c():
  print('c() başlar') ❹
  print('c() döndürür')

def d():
  print('d() başlar')
  print('d() döndürür')

a() ❺
```

Bu programı çalıştırırsanız çıktısı aşağıdaki gibi olur:

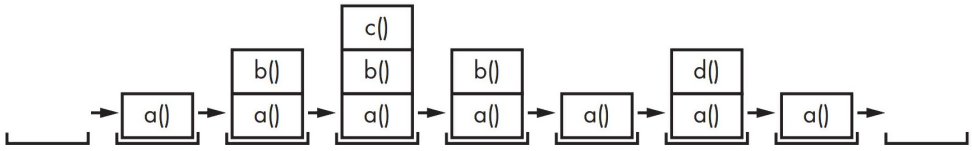
```
a() başlar
b() başlar
c() başlar
c() döndürür
```

```
b() döndürür
d() başlar
d() döndürür
a() döndürür
```

Bu programın çalışmasını ***https://autbor.com/abcdcallstack/*** adresinde görebilirsiniz. `a()` çağrıldığında ❺ `b()`'yi çağırır ❶, `b()` ise `c()`'yi çağırır ❸. `c()` fonksiyonu hiçbir şey çağırmaz; `c()` sadece `c() başlar`'ı ❹ görüntüler ve kendisini çağıran `b()`'deki satıra dönmeden önce döndürür ❸. Program çalışması `b()`'deki `c()`'yi çağıran koda geri döndükten sonra `a()`'daki `b()`'yi çağıran koda geri döner ❶. Program çalışması `b()` fonksiyonundaki bir sonraki satırla devam eder ❷. Bu satır `d()`'ye yapılan bir çağrıdır. `c()` fonksiyonu gibi `d()` fonksiyonu da hiçbir şey çağırmaz. `d()` fonksiyonu sadece `d() başlar`'ı görüntüler ve kendisini çağıran `b()`'deki satıra dönmeden önce döndürür. `b()` başka kod içermediği için programın çalışması `a()`'daki `b()`'yi çağıran satıra geri döner ❷. `a()`'daki son satır programın sonundaki asıl `a()` çağrısına dönmeden önce `a() döndürür`'ü görüntüler ❺.

Çağırma yığını, Python'un her bir fonksiyon çağrısından sonra program çalışmasını nereye döndüreceğini nasıl hatırladığını gösterir. Çağırma yığını programdaki bir değişkende saklanmaz. Daha ziyade, Python çağırma yığınını perde arkasında ele alır. Programınız bir fonksiyonu çağırdığında, Python, çağırma yığınının en üstünde bir *çerçeve nesnesi* oluşturur. Çerçeve nesneleri, Python'un nereye dönebileceğini hatırlayabilmesi için asıl fonksiyon çağrısının satır numarasını saklar. Bir başka fonksiyon çağrısı yapıldığında, Python, çağırma yığınındaki diğer çerçeve nesnesinin üzerine bir diğer çerçeve nesnesi koyar.

Bir fonksiyon çağrısı döndüğünde, Python yığının en üstünden bir çerçeve nesnesini siler ve program çalışmasını bu çerçeve nesnesinde saklanan satır numarasına götürür. Çerçeve nesnelerinin başka bir yerden değil her zaman yığının en üstünden eklenip silindiğine dikkat ediniz. Şekil 3.2, *abcdCallStack.py* içindeki çağırma yığınının her bir fonksiyon çağrıldığındaki ve geri döndüğündeki durumunu gösterir.



Şekil 3.2: *abcdCallStack.py çağırdıkça ve fonksiyonlardan döndükçe çağırma yığınının çerçeve nesneleri*

Çağırma yığınının en üstü program çalışmasının o an içerisinde olduğu fonksiyondur. Çağırma yığını boş olduğunda, program çalışması bütün fonksiyonların dışındaki bir satırdadır. Çağırma yığını, program yazmak için kesinlikle bilmeniz gerekmeyen teknik bir ayrıntıdır. Fonksiyon çağrılarının çağrıldıkları satır numarasına geri döndüğünü anlamak yeterlidir. Ancak çağırma yığınlarını anlamak, bir sonraki kısımda anlatılan yerel ve global kapsamları anlamayı kolaylaştırır.

3.6 Yerel ve Global Kapsam

Çağrılan bir fonksiyona atanan parametreler ve değişkenlerin, bu fonksiyonun *yerel kapsamında* var olduğu söylenir. Tüm fonksiyonların dışında atanan değişkenlerin *global kapsamda* var olduğu söylenir. Yerel kapsamda var olan bir değişkene *yerel değişken*, global kapsamda var olan bir değişkene ise *global değişken* denir. Bir değişken bu ikisinden biri olmalıdır, hem yerel hem global olamaz.

Kapsamı, değişkenler için bir konteyner gibi düşünün. Bir kapsam ortadan kaldırıldığında, kapsamın değişkenlerinde saklanan bütün değerler unutulur. Sadece bir tane global kapsam vardır ve bu kapsam program başladığında oluşturulur. Programınız sonlandığında, global kapsam ortadan kaldırılır ve bütün değişkenleri unutulur. Aksi takdirde, bir programı bir sonraki çalıştırışınızda, değişkenler, bu programı en son çalıştırdığınız andaki değerleriyle hatırlayacaktır.

Yerel kapsam bir fonksiyon çağrıldığında oluşturulur. Fonksiyonun içerisinde atanan her değişken fonksiyonun yerel kapsamında var olur. Fonksiyon döndüğünde, yerel kapsam ortadan kaldırılır ve bu değişkenler unutulur. Fonksiyonu bir sonraki çağırışınızda, yerel değişkenler, fonksiyonun en son çağrıldığı anda içlerinde depolanan değerlerini hatırlamayacaktır. Yerel değişkenler aynı zamanda çağırma yığınında çerçeve nesneleri içerisinde de saklanır.

Çeşitli nedenlerden ötürü kapsamlar önemlidir:

- Bütün fonksiyonların dışarısındaki global kapsamdaki kod hiçbir yerel değişkeni kullanamaz.
- Bununla birlikte, yerel kapsamdaki kod global değişkenlere erişebilir.
- Bir fonksiyonun yerel kapsamındaki kod, bir başka yerel kapsamdaki değişkenleri kullanamaz.
- Farklı kapsamlarda olmaları hâlinde, farklı değişkenler için aynı ismi kullanabilirsiniz. Yani `spam` isimli yerel bir değişken ve yine `spam` isminde global bir değişken olabilir.

4.7 Referanslar

Gördüğünüz üzere, değişkenler dizgileri ve tam sayı değerlerini "saklar." Ancak bu açıklama Python'un gerçekten ne yaptığının basitleştirilmiş bir hâlidir. Teknik olarak değişkenler, değerlerin saklandığı bilgisayar belleği konumlarına olan referansları saklar. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> spam = 42
>>> cheese = spam
>>> spam = 100
>>> spam
100
>>> cheese
42
```

`spam` değişkenine 42 atadığınızda, aslında bilgisayarın belleğinde 42 değerini oluşturuyor ve `spam` değişkeninde buna bir referans saklamış oluyorsunuz. `spam`'deki değeri kopyalayıp `cheese` (ÇN: peynir) değişkenine atadığınızda, aslında bu *referansı* kopyalıyorsunuz. Hem `spam` hem cheese değişkenleri bilgisayarın belleğindeki 42 değerine işaret etmektedir. Daha sonra `spam`'daki değeri 100 olarak değiştirdiğinizde, yeni bir 100 değeri oluşturmakta ve `spam`'da bu değere olan bir referansı saklamaktasınız. Bu, `cheese`'deki değeri etkilemez. Tam sayılar *değişmez* değerlerdir. `spam` değişkenini değiştirmek, esasında onu bellekte tamamen farklı bir değere işaret ettirmektedir.

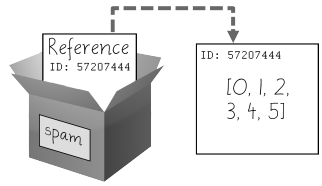
Ancak listeler bu şekilde çalışmaz çünkü liste değerleri değişebilir. Yani listeler *değişebilirdir*. İşte bu ayrımı anlaması kolay hâle getirecek olan bir kod. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> spam = [0, 1, 2, 3, 4, 5] ❶
>>> cheese = spam ❷ # Liste değil referans kopyalanmaktadır
>>> cheese[1] = 'Hello!' ❸ # Bu, liste değerini değiştirir
>>> spam
[0, 'Hello!', 2, 3, 4, 5]
>>> cheese # cheese değişkeni aynı listeye işaret eder
[0, 'Hello!', 2, 3, 4, 5]
```

Bu size garip gelebilir. Kod sadece `cheese` listesine dokunuyor ancak hem `cheese` hem de `spam` listeleri değişmiş gibi görünüyor.

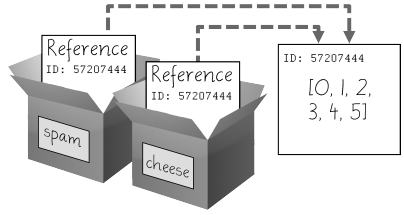
Listeyi oluşturduğunuzda ❶, ona `spam` değişkeninde bir referans atamaktasınız. Ancak bir sonraki satır ❷ liste değerinin kendisini değil, sadece `spam`'daki liste referansını `cheese`'e kopyalamaktadır. Bu ise `spam` ve `cheese`'de saklanan değerlerin her ikisinin de aynı listeye işaret ettiği anlamına gelir. Altta yatan tek bir liste vardır çünkü listenin kendisi aslında hiçbir zaman kopyalanmamıştır. Dolayısıyla `cheese`'in ilk elemanını değiştirdiğinizde ❸ `spam`'ın işaret ettiği aynı listeyi değiştirmiş olursunuz.

Değişkenlerin değer içeren kutular gibi olduğunu hatırlayınız. Bu bölümdeki daha önceki şekiller, kutulardaki listelerin tam olarak doğru olmadığını göstermektedir. Çünkü liste değişkenleri aslında liste içermezler, listelere olan *referanslar* içerirler. (Bu referanslar, Python'un dâhili olarak kullandığı ID numaralarına sahip olacaktır ancak bunları göz ardı edebilirsiniz.) Şekil 4.4 kutuları değişkenler için bir metafor olarak kullanarak, `spam` değişkenine bir liste atandığında ne olduğunu göstermektedir.



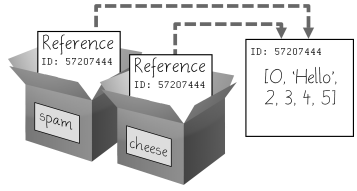
Şekil 4.4: *`spam = [0, 1, 2, 3, 4, 5]` gerçek listeyi değil, listeye olan bir referansı saklar*

Daha sonra, Şekil 4.5'te spam'daki referans `cheese`'e kopyalanmaktadır. `cheese`'de yeni bir liste değil, sadece yeni bir referans oluşturulup saklanmaktadır. Her iki referansın da nasıl aynı listeye işaret ettiğine dikkat ediniz.



Şekil 4.5: *`spam = cheese` listeyi değil referansı kopyalar*

`cheese`'in işaret ettiği listeyi değiştirdiğinizde, `spam`'ın işaret ettiği liste de değişir. Çünkü hem `cheese` hem `spam` aynı listeye işaret etmektedir. Bunu Şekil 4.6'da görebilmektesiniz.



Şekil 4.6: *`cheese[1] = 'Hello!'` her iki değişkenin de işaret ettiği listeyi değiştirir*

9.2 Dosya Okuma/Yazma Süreci

Klasörler ve göreceli yollarla rahatça çalışmaya başladıktan sonra, okunacak ve yazılacak dosyaların konumunu belirleyebileceksiniz. Sonraki birkaç bölümde ele alınan fonksiyonlar düz metin dosyalarına uygulanacaktır. *Düz metin dosyaları* yalnızca temel metin karakterlerini içerirler ve yazı tipi, boyut veya renk bilgilerini içermezler. *.txt* uzantılı metin dosyaları veya *.py* uzantılı Python betik dosyaları düz metin dosyalarına örnektir. Bu dosyalar Windows'un Notepad veya macOS'un TextEdit uygulamalarıyla açılabilir. Programlarınız düz metin dosyalarının içeriğini kolayca okuyabilir ve bunlara sıradan bir dizgi değeri gibi davranır.

İkili dosyalar, kelime işlem belgeleri, PDF'ler, resimler, elektronik tablolar ve yürütülebilir programlar gibi diğer tüm dosya türleridir. İkili bir dosyayı Notepad veya TextEdit'te açarsanız Şekil 9.6'daki gibi karmakarışık ve anlamsız görünür.

Şekil 9.6: *Windows calc.exe programının Notepad'de açılmış hâli*

Her farklı türdeki ikili dosyaların kendi yöntemiyle ele alınması gerektiğinden, bu kitap doğrudan ham ikili dosyaları okumaya ve yazmaya girmeyecektir. Neyse ki birçok modül ikili dosyalarla çalışmayı kolaylaştırır; bunlardan biri olan shelve modülünü bu bölümün ilerleyen kısımlarında keşfedeceksiniz. pathlib modülünün `read_text()` metodu bir metin dosyasının içeriğinin tümünü bir dizgi olarak döndürür. Bu modülün `write_text()` metodu kendisine gönderilen dizgiyle yeni bir metin dosyası oluşturur (veya var olanın üzerine yazar). Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> from pathlib import Path
>>> p = Path('spam.txt')
>>> p.write_text('Hello, world!')
13
>>> p.read_text()
'Hello, world!'
```

Bu metot çağrıları, içeriği `'Hello, world!'` olan *spam.txt* dosyasını oluşturur. `write_text()`'in döndürdüğü 13 rakamı dosyaya 13 karakterin yazıldığını

belirtir. (Bu bilgiyi sıklıkla görmezden gelebilirsiniz.) `read_text()` çağrısı yeni dosyamızın içeriğini bir dizgi olarak okur ve döndürür: `'Hello, world!'`

Bu `Path` nesne metotlarının yalnızca dosyalarla temel etkileşimler sağladığını unutmayın. Bir dosyaya yazmanın daha yaygın yolu, `open()` fonksiyonunu ve dosya nesnelerini kullanmayı içerir. Python'da dosya okumak veya yazmak için üç adım vardır:

1. Bir `File` nesnesi döndürmek için `open()` fonksiyonunu çağır.
2. `File` nesnesi üzerinde `read()` veya `write()` metodunu çağır.
3. `File` nesnesi üzerinde `close()` metodunu çağırarak dosyayı kapat.

Bu adımları aşağıdaki kısımlarda ele alacağız.

9.2.1 `open()` Fonksiyonuyla Dosyaları Açma

`open()` fonksiyonuyla bir dosyayı açmak için bu fonksiyona açmak istediğiniz dosyayı belirten bir dizgi yolu gönderirsiniz. Bu yol mutlak veya göreceli bir yol olabilir. `open()` fonksiyonu bir `File` nesnesi döndürür.

Notepad veya TextEdit kullanarak *hello.txt* isimli bir metin dosyası oluşturarak bunu deneyiniz. Bu metin dosyasının içeriği olarak **Hello, world!** yazınız ve dosyayı kullanıcı home klasöründe kaydediniz. Ardından aşağıdakini etkileşimli kabuğa giriniz:

```
>>> helloFile = open(Path.home() / 'hello.txt')
```

`open()` fonksiyonu dizgileri de kabul eder. Eğer Windows kullanıyorsanız, aşağıdakini etkileşimli kabuğa giriniz:

```
>>> helloFile = open('C:\\Users\\your_home_folder\\hello.txt')
```

macOS kullanıyorsanız, aşağıdakini etkileşimli kabuğa giriniz:

```
>>> helloFile = open('/Users/your_home_folder/hello.txt')
```

your_home_folder kısmını bilgisayarınızın kullanıcı ismiyle değiştirdiğinizden emin olunuz. Örneğin, benim ismim Al, dolayısıyla Windows'ta `'C:\\Users\\Al\\hello.txt'` girerim. `open()` fonksiyonunun Python 3.6 itibariyle sadece `Path` nesnelerini kabul ettiğine dikkat ediniz. Önceki versiyonlarda `open()`'a her zaman bir dizgi göndermeniz gerekir.

Her iki komut da dosyayı "düz metin okuyor" modunda veya kısaca okuma modunda açacaktır. Bir dosya okuma modunda açıldığında, Python sadece dosyadan verileri okumanıza izin verir; herhangi bir şekilde bu dosyaya yazamaz veya dosyayı değiştiremezsiniz. Okuma modu Python'da açtığınız dosyalar

için varsayılan moddur. Ancak Python'un varsayılanlarına güvenmek istemiyorsanız, `open()`'a ikinci argüman olarak `'r'` dizgi değerini göndererek modu açıkça belirtebilirsiniz. Dolayısıyla `open('/Users/Al/hello.txt', 'r')` ve `open('/Users/Al/hello.txt')` aynı şeyi yapar.

`open()` çağrısı bir `File` nesnesi döndürür. Bir `File` nesnesi bilgisayarınızdaki bir dosyayı temsil eder. Bu nesne hâlihazırda aşina olduğunuz liste ve sözlükler gibi Python'da bir başka değer tipidir. Bir önceki örnekte `File` nesnesini `helloFile` değişkeninde sakladınız. Artık ne zaman bir dosyayı okumak veya bir dosyaya yazmak isterseniz, `helloFile`'daki `File` nesnesi üzerinde metotlar çağırarak bunu yapabilirsiniz.

9.2.2 Dosyaların İçeriklerini Okuma

Artık bir `File` nesneniz olduğuna göre bu nesneden okumaya başlayabilirsiniz. Dosyanın içeriğinin tamamını bir dizgi olarak okumak isterseniz `File` nesnesinin `read()` metodunu kullanınız. `helloFile`'da depoladığınız *hello.txt* `File` nesnesiyle başlayalım. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> helloContent = helloFile.read()
>>> helloContent
'Hello, world!'
```

Dosyanın içeriğini tek bir büyük dizgi değeri olarak düşünürseniz, `read()` metodu dosyada depolanan dizgiyi döndürür.

Alternatif olarak, her metin satırı bir dizgi olacak şekilde dosyadan dizgi değerlerinin bir listesini almak için `readlines()` metodunu kullanabilirsiniz. Örneğin, *hello.txt* ile aynı dizinde s*onnet29.txt* isimli bir dosya oluşturunuz ve bu dosyanın içine aşağıdakileri yazınız:

```
When, in disgrace with fortune and men's eyes,
I all alone beweep my outcast state,
And trouble deaf heaven with my bootless cries,
And look upon myself and curse my fate,
```

Dört satırı da satır sonlarıyla ayırdığımızdan emin olunuz. Ardından aşağıdakini etkileşimli kabuğa giriniz:

```
>>> sonnetFile = open(Path.home() / 'sonnet29.txt')
>>> sonnetFile.readlines()
[When, in disgrace with fortune and men's eyes,\n',
' I all alone beweep my outcast state,\n',
And trouble deaf heaven with my bootless cries,\n',
And look upon myself and curse my fate,']
```

Dosyanın son satırı hariç dizgi değerlerinin her birinin `\n` yeni satır karakteriyle bittiğine dikkat ediniz. Dizgilerin bir listesiyle çalışmak, tek bir büyük dizgi değeriyle çalışmaktan genellikle daha kolaydır.

9.2.3 Dosyalara Yazma

Python, `print()` fonksiyonunun dizgileri ekrana "yazmasına" benzer şekilde bir dosyaya içerik yazmanıza olanak tanır. Yine de okuma modunda açtığınız bir dosyaya yazamazsınız. Bunun yerine, "düz metin yaz" modunda veya "sona düz metin ekle" modunda veya kısaca *yazma modunda* ve sona *ekleme modunda* açmanız gerekir.

Yazma modu, bir değişkenin değerinin üzerine yeni bir değer yazdığınızda olduğu gibi mevcut dosyanın üzerine yazar ve sıfırdan başlar. Dosyayı yazma modunda açmak için `open()`'a ikinci argüman olarak `'w'` gönderiniz. Öte yandan, sona ekleme modu mevcut dosyanın sonuna metin ekleyecektir. Bunu, değişkenin üzerine tamamen yazmak yerine, bir değişkendeki bir listenin sonuna ekleme olarak düşünebilirsiniz. Dosyayı sona ekleme modunda açmak için `open()`'a ikinci argüman olarak `'a'` gönderiniz.

`open()`'a gönderilen dosya adı yoksa hem yazma hem de sona ekleme modu yeni, boş bir dosya oluşturur. Bir dosyayı okuduktan veya yazdıktan sonra, dosyayı tekrar açmadan önce `close()` metodunu çağırınız.

Bu kavramları bir araya getirelim. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> baconFile = open('bacon.txt', 'w')
>>> baconFile.write('Hello, world!\n')
13
>>> baconFile.close()
>>> baconFile = open('bacon.txt', 'a')
>>> baconFile.write('Bacon is not a vegetable.')
>>> baconFile.close()
>>> baconFile = open('bacon.txt')
>>> content = baconFile.read()
>>> baconFile.close()
>>> print(content)
Hello, world!
Bacon is not a vegetable.
```

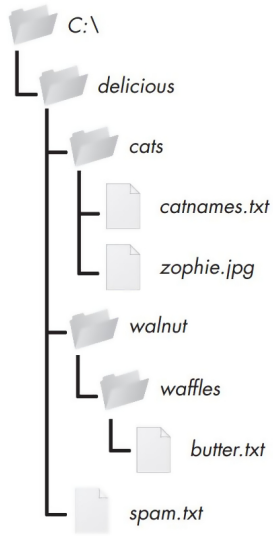
Öncelikle *bacon.txt*'yi yazma modunda açıyoruz. Henüz bir *bacon.txt* dosyası olmadığından Python bir tane oluşturur. Açılmış olan dosya üzerinde `write()`'ı çağırmak ve `write()`'a `'Hello, world! /n'` dizgi argümanını göndermek dizgiyi dosyaya yazar ve yeni satır da dâhil olmak üzere yazılmış olan karakterlerin sayısını döndürür. Daha sonra dosyayı kapatıyoruz.

Dizgiyi değiştirmek yerine, dosyanın mevcut içeriğine yeni yazdığımız metni

10.2 Dizin Ağacında Gezinme

Diyelim ki bir klasördeki her dosyayı ve ayrıca o klasörün her alt klasöründeki her dosyayı yeniden adlandırmak istiyorsunuz. Yani, ilerledikçe her dosyaya dokunarak dizin ağacında gezinmek istiyorsunuz. Bunu yapmak için bir program yazmak zor olabilir; neyse ki Python bu işlemi sizin için halledecek bir fonksiyon sağlar.

Şekil 10.1'deki dizin üzerinde `os.walk()` fonksiyonunu kullanan örnek bir program aşağıda yer almaktadır:



Şekil 10.1: *Üç klasör ve dört dosya içeren örnek bir klasör*

```
import os
for folder,subfolders,filenames in os.walk('C:\\delicious'):
  print('Mevcut klasör ' + folderName)

  for subfolder in subfolders:
    print( folder + ' ALT KLASÖRÜ: ' + subfolder)

  for filename in filenames:
    print( folder + ' İÇERİSİNDEKİ DOSYA: ' + filename)

  print('')
```

`os.walk` fonksiyonuna tek bir dizgi değeri gönderilmiştir: Bir klasörün yolu. Bir sayı aralığında gezinmek için nasıl `range()` fonksiyonunu kullanabiliyorsanız, bir dizin ağacında gezinmek için de `os.walk()`'u bir for döngüsü deyiminde

kullanabilirsiniz. `range()`'den farklı olarak `os.walk()` fonksiyonu döngüdeki her yinelemede üç değer döndürecektir:

- Mevcut klasörün isminin bir dizgisi
- Mevcut klasördeki klasörlerin dizgilerinin bir listesi
- Mevcut klasördeki dosyaların dizgilerinin bir listesi

(Mevcut klasör derken, for döngüsünün mevcut yinelemesi için olan klasörü kastetmekteyim. Programın geçerli çalışma dizini `os.walk()` tarafından değiştirilmemektedir.)

`for i in range(10):` kodunda i değişken ismini seçebildiğiniz gibi daha önce listelenen üç değer için de değişken isimleri seçebilirsiniz. Ben genellikle `foldername`, `subfolders` ve `filenames` (ÇN: klasör ismi, alt klasörler ve dosya isimleri) isimlerini kullanıyorum.

Bu programı çalıştırdığınızda aşağıdaki çıktıyı verecektir:

```
Mevcut klasör C:\delicious
C:\delicious ALT KLASÖRÜ: cats
C:\delicious ALT KLASÖRÜ: walnut
C:\delicious İÇERİSİNDEKİ DOSYA: spam.txt
1
Mevcut klasör C:\delicious\cats
C:\delicious İÇERİSİNDEKİ DOSYA: catnames.txt
C:\delicious İÇERİSİNDEKİ DOSYA: zophie.jpg

Mevcut klasör C:\delicious\walnut
C:\delicious ALT KLASÖRÜ: waffles

Mevcut klasör C:\delicious\walnut\waffles
C:\delicious İÇERİSİNDEKİ DOSYA: butter.txt
```

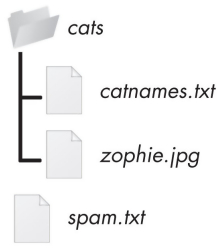
`os.walk()`, `subfolder` ve `filename` değişkenleri için dizgilerin bir listesini döndürdüğünden, bu listeleri kendi `for` döngülerinde kullanabilirsiniz. `print()` fonksiyon çağrılarını kendi kullanıcı tanımlı kodunuzla değiştiriniz (Ya da bir tanesine veya her ikisine birden ihtiyacınız yoksa `for` döngülerini siliniz.)

10.3 `zipfile` Modülüyle Dosyaları Sıkıştırma

Diğer birçok dosyanın sıkıştırılmış içeriğini tutabilen ZIP dosyalarına (*.zip* dosya uzantısına sahip) aşina olabilirsiniz. Bir dosyanın sıkıştırılması boyutunu küçültür, bu da dosyayı internet üzerinden aktarırken kullanışlıdır. Ayrıca bir ZIP dosyası birden fazla dosya ve alt klasör içerebileceğinden, birkaç dosyayı

tek bir dosyada paketlemenin kullanışlı bir yoludur. Arşiv dosyası olarak adlandırılan bu tek dosya daha sonra örneğin bir e-postaya eklenebilir.

Python programlarınız `zipfile` modülündeki fonksiyonları kullanarak ZIP dosyaları oluşturabilir ve açabilir (veya *ayıklayabilir*). Diyelim ki Şekil 10.2'de gösterilen içeriğe sahip *example.zip* adlı bir ZIP dosyanız var.



Şekil 10.2: *example.zip'in içeriği*

Bu ZIP dosyasını *https://nostarch.com/automatestuff2/* adresinden indirebilir veya hâlihazırda bilgisayarınızda mevcut bir ZIP dosyasını kullanarak devam edebilirsiniz.

10.3.1 ZIP Dosyalarını Okuma

Bir ZIP dosyasının içeriğini okuyabilmek için öncelikle bir `ZipFile` nesnesi (büyük Z ve F harflerine dikkat ediniz) oluşturmanız gerekir. `ZipFile` nesneleri kavramsal olarak bir önceki bölümde `open()` fonksiyonuyla döndürüldüğünü gördüğünüz File nesnelerine benzerdir: Bu nesneler, programın dosyayla etkileşime girdiği değerlerdir. `ZipFile` nesnesi oluşturmak için `zipfile.ZipFile()` fonksiyonunu çağırınız ve ona *.ZIP* dosyasının dosya ismini gönderiniz. `zipfile`'ın Python modülünün ismi, `ZipFile`'ın ise fonksiyonun ismi olduğuna dikkat ediniz.

Örneğin, aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import zipfile, os
>>> from pathlib import Path
>>> p = Path.home()
>>> exampleZip = zipfile.ZipFile(p / 'example.zip') örnekZip =
>>> exampleZip.namelist()
['spam.txt', 'cats/', 'cats/catnames.txt', 'cats/zophie.jpg']
>>> spamInfo = exampleZip.getinfo('spam.txt')
>>> spamInfo.file_size
13908
>>> spamInfo.compress_size
3828
```

```
>>> f'Compressed file is {round(spamInfo.file_size / spamInfo.compress_size,
    2)}x smaller!') ❶
'Compressed file is 3.63x smaller!'
#Sıkıştırılmış dosya 3.63 kat daha küçüktür!
>>> exampleZip.close()
```

Bir `ZipFile` nesnesinin, ZIP dosyasında yer alan tüm dosya ve klasörler için dizgilerin bir listesini döndüren `namelist()` (ÇN: İsim listesi) metodu vardır. Bu dizgiler, bu belirli dosya ile ilgili bir `ZipInfo` (ÇN: ZipBilgisi) nesnesi döndürmek için `getinfo()` `ZipFile` metoduna gönderilebilir. `ZipInfo` nesnelerinin bayt cinsinden `file_size` (ÇN: Dosya boyutu) ve `compress_size` (ÇN: Sıkıştırılmış boyut) gibi kendi nitelikleri mevcuttur. Bu nitelikler sırasıyla orijinal dosya boyutunun ve sıkıştırılmış dosya boyutunun tam sayı değerlerini barındırır. `ZipFile` nesnesi tüm bir arşiv dosyasını temsil ederken, `ZipInfo` nesnesi arşivdeki tek bir dosya hakkında faydalı bilgiler barındırır.

❶'deki komut orijinal dosya boyutunu sıkıştırılmış dosya boyutuna bölerek example.zip'in ne kadar verimli sıkıştırıldığını hesaplar ve bu bilgiyi ekrana yazdırır.

10.3.2 ZIP Dosyalarından Çıkarma

ZipFile nesnelerinin `extractall()` metodu, bir ZIP dosyasındaki bütün dosya ve klasörleri çıkarır ve geçerli çalışma dizinine yerleştirir.

```
>>> import zipfile, os
>>> from pathlib import Path
>>> p = Path.home()
>>> exampleZip = zipfile.ZipFile(p / 'example.zip')
>>> exampleZip.extractall() ❶
>>> exampleZip.close()
```

Bu kodu çalıştırdıktan sonra, *example.zip*'in içeriği *C:* dizinine çıkarılacaktır. İsteğe bağlı olarak, dosyaları geçerli çalışma dizini dışında bir klasöre çıkarmasını sağlamak için `extractall()`'a bir klasör ismi gönderebilirsiniz. `extractall()` metoduna gönderilen klasör yoksa oluşturulacaktır. Örneğin, ❶'deki çağrıyı `exampleZip.extractall('C:\\delicious')` ile değiştirirseniz, kod, dosyaları *example.zip*'ten çıkartıp, yeni bir klasör olan *C:\delicious*'a yerleştirecektir.

`ZipFile` nesneleri için `extract()` metodu, ZIP dosyasından tek bir dosya çıkaracaktır. Etkileşimli kabuk örneğine devam edin:

```
>>> exampleZip.extract('spam.txt')
'C:\\spam.txt'
>>> exampleZip.extract('spam.txt', 'C:\\some\\new\\folders')
'C:\\some\\new\\folders\\spam.txt'
>>> exampleZip.close()
```

Şekil 12.4: *Chrome tarayıcısında Geliştirici Araçları*

HTML'İ AYRIŞTIRMAK İÇİN DÜZENLİ İFADELER KULLANMAYINIZ

Bir dizgideki belirli bir HTML parçasının konumunu belirlemek düzenli ifadeler için mükemmel bir durum gibi görünüyor. Ancak size bunu yapmamanızı tavsiye ederim HTML'in biçimlendirilmesinin ve yine de geçerli HTML olarak kabul edilmesinin birçok farklı yolu vardır. Ancak tüm bu olası varyasyonları bir düzenli ifade ile yakalamaya çalışmak sıkıcı ve hataya açık olabilir. bs4 gibi HTML'i ayrıştırmak için özel olarak geliştirilmiş bir modülün hatalarla sonuçlanması daha az olasıdır.

HTML'i neden düzenli ifadelerle ayrıştırmamanız gerektiği üzerine geniş bir tartışmayı *https://stackoverflow.com/a/1732454/1893164/* adresinde bulabilirsiniz.

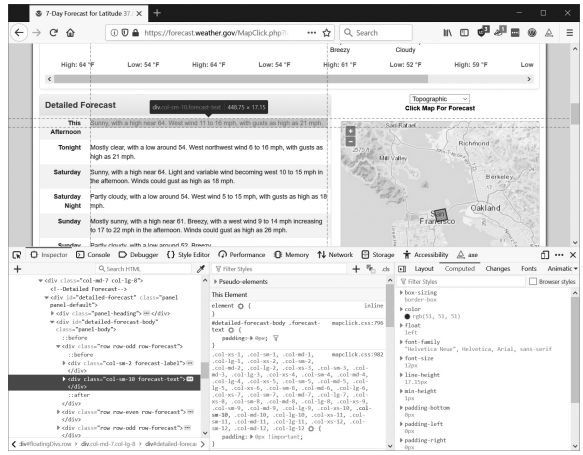
12.4.5 HTML Elementlerini Bulmak için Geliştirici Araçlarını Kullanma

Programınız `requests` modülünü kullanarak bir web sayfasını indirdikten sonra sayfanın HTML içeriğine tek bir dizgi değeri olarak sahip olursunuz. Şimdi, HTML'in hangi kısmının ilgilendiğiniz web sayfasındaki bilgilere karşılık geldiğini bulmanız gerekmektedir.

Tarayıcının geliştirici araçlarının yardımcı olabileceği yer burasıdır. Diyelim ki ***https://weather.gov/*** adresinden hava tahmin verisi almak için bir program yazmak istiyorsunuz. Herhangi bir kod yazmadan önce küçük bir araştırma

yapınız. Siteyi ziyaret eder ve 94105 posta kodunu aratırsanız site sizi bu bölge için hava tahminini gösteren bir sayfaya götürecektir.

Peki, bu posta kodu için hava tahmin bilgisini kazımak isterseniz ne olacak? Bu bilgi sayfada her neredeyse oraya sağ tıklayınız (veya macOS'ta `CONTROL-tıkla`) ve açılan bağlam menüsünden Inspect Element'i seçiniz. Bu, Geliştirici Araçları penceresini açacaktır. Bu pencere size web sayfasının bu özel kısmını üreten HTML'i gösterir. Şekil 12.5 geliştirici araçlarının en yakın hava tahmininin HTML için açılmış hâlini göstermektedir. ***https://weather.gov/*** sitesi web sayfalarının tasarımını değiştirdiği takdirde yeni elementleri incelemek için bu süreci tekrarlamanız gerektiğine dikkat ediniz.



Şekil 12.5: *Geliştirici araçları vasıtasıyla hava tahmini metnini taşıyan elementi incelemek*

Geliştirici araçlarından web sayfasının hava tahmininden sorumlu HTML'inin
`<div class = 'col-sm-10 forecast-text'>Sunny, with a high near 64. West wind 11 to 16 mph, with gusts as high as 21 mph.</div>`
(ÇN: En yüksek sıcaklık 64 olmak üzere güneşli, saatte 21 mile kadar ani rüzgârlarla birlikte saatte 11 milden 16 mile kadar batılı rüzgârlar) olduğunu görebilirsiniz. Tam olarak aradığınız şey budur! Görünen odur ki hava tahmin bilgileri `forecast-text` CSS (ÇN: Cascaded Style Sheet) sınıfıyla birlikte bir elementinin içerisindedir. Tarayıcının geliştirici konsolunda bu elemente sağ tıklayınız ve açılan bağlam menüsünden Kopyala » CSS Seçici'yi seçiniz. Bu, `'div.row-odd:nth-child(1) > div:nth-child(2)'` gibi bir dizgiyi panoya kopyalayacaktır. Bu dizgiyi, bu bölümün ilerleyen kısımlarında açıklanacağı gibi Beautiful Soup'un `select()` ve Selenium'un `find_element_by_css_selector()` metotlarında kullanabilirsiniz. Artık ne aradığınızı bildiğinize göre, Beautiful Soup modülü onu dizgide bulmanıza yardımcı olacaktır.

12.5 bs4 Modülü ile HTML'i Ayrıştırma

Beautiful Soup bir HTML sayfasından bilgi çıkarmak için kullanılan bir modüldür (ve bu amaç doğrultusunda düzenli ifadelerden çok daha iyidir). Beautiful Soup modülünün ismi `bs4`'tür (Beautiful Soup, versiyon 4 anlamına gelir). Bu modülü kurmak için komut satırından `pip install --user beautifulsoup4` komutunu çalıştırmanız gerekir. (Üçüncü-parti modülleri kurmakla ilgili talimatlar için Ek A'ya bakınız.) `beautifulsoup4` kurulum için kullanılan isim olmakla birlikte Beautiful Soup'u içe aktarmak için `import bs4`'u kullanırsınız.

Bu bölüm için Beautiful Soup örnekleri, sabit sürücüdeki bir HTML dosyasını ayrıştıracak (yani, parçalarını analiz edecek ve tanımlayacaktır). Mu'da yeni bir dosya düzenleyici sekmesi açınız, aşağıdakini giriniz ve *example.html* olarak kaydediniz. Alternatif olarak bu dosyayı ***https://nostarch.com/automatestuff2/*** adresinden de indirebilirsiniz.

```
<!-- This is the example.html example file. -->

<html><head><title>The Website Title</title></head>
<body>
<p>Download my <strong>Python</strong> book from <a href="https
    ://inventwithpython.com">my website</a>.</p>
<p class="slogan">Learn Python the easy way!</p>
<p>By <span id="author">Al Sweigart</span></p>
</body></html>
```

Gördüğünüz üzere basit bir HTML sayfası bile farklı birçok etiket ve nitelik içermektedir ve karmaşık web siteleriyle birlikte işler hızla kafa karıştırıcı bir hâl alır. Neyse ki Beautiful Soup, HTML ile çalışmayı çok daha kolaylaştırmaktadır.

12.5.1 HTML'den BeautifulSoup Nesnesi Oluşturma

`bs4.BeautifulSoup()` fonksiyonunun ayrıştıracağı HTML'i içeren bir dizgi ile çağrılması gerekir. `bs4.BeautifulSoup()` fonksiyonu `BeautifulSoup` nesnesi döndürür. Bilgisayarınız internete bağlıyken aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import requests, bs4
>>> res = requests.get('https://nostarch.com')
>>> res.raise_for_status()
>>> noStarchSoup = bs4.BeautifulSoup(res.text, 'html.parser')
>>> type(noStarchSoup)
<class 'bs4.BeautifulSoup'>
```

Bu kod, No Starch Press web sitesinden ana sayfayı indirmek için `requests.get()`'i kullanır ve daha sonra yanıtın text niteliğini `bs4.BeautifulSoup()`'a

13.7 Hücrelerin Yazı Tipi Stilini Belirleme

Belirli hücrelere, satırlara veya sütunlara stil vermek elektronik tablonuzdaki önemli alanları vurgulamanıza yardımcı olabilir. Örneğin, ürün elektronik tablosunda programınız patates, sarımsak ve yaban havucu satırlarına kalın yazı uygulayabilir. Veya belki de pound başına düşen maliyetin 5 dolardan daha büyük olduğu her satırı italik yapmak istiyorsunuz. Büyük bir elektronik tablonun kısımlarına elle stil vermek sıkıcı bir iştir ama programlarınız bu işi anında yapabilir.

Önemli hücrelerdeki yazı tipi stilini kullanıcı tanımlı hâle getirmek için `openpyxl.styles` modülünden `Font()` fonksiyonunu içe aktarınız.

```
from openpyxl.styles import Font
```

Bu `openpyxl.styles.Font()` yazmak yerine `Font()` yazmanıza olanak sağlar. (`import` deyiminin bu stilini gözden geçirmek için 51. sayfada "Modülleri İçe Aktarma" kısmına bakınız.)

Aşağıda yeni bir çalışma sayfası oluşturan ve A1 hücresini 24 punto italik yazı tipine ayarlayan bir örnek bulunmaktadır. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import openpyxl
>>> from openpyxl.styles import Font
>>> wb = openpyxl.Workbook()
>>> sheet = wb['Sheet']
# Bir yazı tipi oluştur.
>>> italic24Font = Font(size=24, italic=True) ❶
# Yazı tipini A1'e uygula.
>>> sheet['A1'].font = italic24Font ❷
>>> sheet['A1'] = 'Hello, world!'
>>> wb.save('styles.xlsx')
```

Bu örnekte `Font(size=24, italic=True)` bir `Font` nesnesi döndürür ve bu nesne `italic24Font`'ta depolanır ❶. `Font()`'un anahtar kelime argümanları olan `size` ve `italic` (ÇN: boyut ve italik), Font nesnesinin stil verme bilgisini yapılandırır. Ve `sheet['A1'].font italic24Font` nesnesine atandığında ❷, bütün bu yazı tipine stil verme bilgisi A1 hücresine uygulanır.

13.8 Yazı Tipi Nesneleri

`font` niteliklerini ayarlamak için `Font()`'a anahtar kelime argümanları gönderirsiniz. Tablo 13.2 `Font()` fonksiyonu için olası anahtar kelime argümanlarını göstermektedir.

Tablo 13.2: `Font` Nesneleri için Anahtar Kelime Argümanları

Anahtar kelime argüman	Veri tipi	Açıklama
`name` (ÇN: isim)	Dizgi	'Calibri've 'Times New Roman' gibi bir yazı tipi ismi
`size` (ÇN: boyut)	Tam sayı	Punto boyutu
`bold` (ÇN: kalın)	Boole	Kalın yazı tipi için True
`italic` (ÇN: italik)	Boole	İtalik yazı tipi için True

Font nesnesi oluşturmak için `Font()`'u çağırabilir ve bu `Font` nesnesini bir değişkende saklayabilirsiniz. Daha sonra bu değişkeni bir `Cell` nesnesinin font niteliğine atarsınız. Örneğin, aşağıdaki kod çeşitli yazı tipi stilleri oluşturur:

```
>>> import openpyxl
>>> from openpyxl.styles import Font
>>> wb = openpyxl.Workbook()
>>> sheet = wb['Sheet']

>>> fontObj1 = Font(name='Times New Roman', bold=True)
>>> sheet['A1'].font = fontObj1
>>> sheet['A1'] = 'Bold Times New Roman'

>>> fontObj2 = Font(size=24, italic=True)
>>> sheet['B3'].font = fontObj2
>>> sheet['B3'] = '24 pt Italic'

>>> wb.save('styles.xlsx')
```

Burada `fontObj1`'de bir `Font` nesnesi saklıyoruz ve ardından A1 `Cell` nesnesinin font niteliğini `fontObj1` olarak belirliyoruz. İkinci bir hücrenin yazı tipini belirlemek için süreci bir başka `Font` nesnesiyle tekrarlıyoruz. Bu kodu çalıştırdıktan sonra elektronik tablodaki A1 ve B3 hücrelerinin stilleri Şekil 13.4'teki gösterildiği gibi kullanıcı tanımlı yazı tipleri olarak belirlenecektir.

Şekil 13.4: *Kullanıcı tanımlı yazı tipi stilleriyle birlikte bir elektronik tablo*

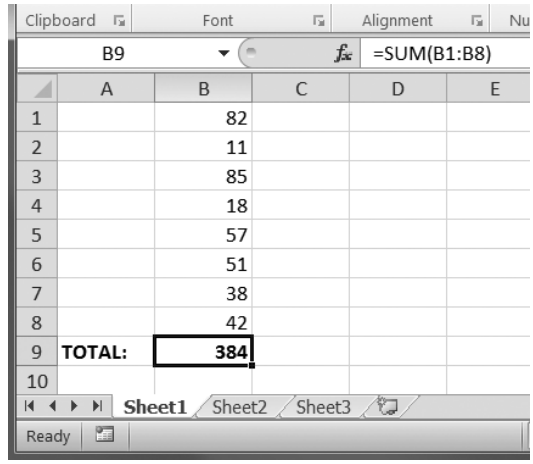
A1 hücresi için yazı tipi ismini `'Times New Roman'` ve `bold`'u `True` yapıyoruz, böylece metnimiz kalın Times New Roman olarak görünecektir. Bir boyut belirtmedik, dolayısıyla `openpyxl` varsayılanı olan 11 kullanılmaktadır. B3 hücresinde, metnimiz italiktir ve boyutu 24'tür. Yazı tipi ismi belirtmedik, dolaysıyla `openpyxl` varsayılanı Calibri kullanıldı.

13.9 Formüller

Eşittir işaretiyle başlayan Excel formülleri, hücreleri diğer hücrelerden hesaplanan değerleri içerecek şekilde yapılandırabilir. Bu kısımda, herhangi bir normal değer gibi hücrelere programlı olarak formüller eklemek için `openpyxl` modülünü kullanacaksınız. Örneğin:

```
>>> sheet['B9'] = '=SUM(B1:B8)'
```

Bu, *=SUM(B1:B8)*'i B9 hücresinde bir değer olarak saklayacaktır. Bu, B9 hücresini B1 hücresinden B8 hücresine kadar olan hücrelerdeki değerlerin toplamını hesaplayan bir formüle ayarlar. Bunu Şekil 13-5'te eylem hâlinde görebilirsiniz.



Şekil 13.5: *B9 hücresi B1'den B8'e kadar olan hücreleri toplayan =SUM(B1:B8) formülünü içermektedir*

Bir Excel formülü bir hücredeki herhangi bir metin değeri gibi belirlenir. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.active
```

```
>>> sheet['A1'] = 200
>>> sheet['A2'] = 300
# Formülü belirle.
>>> sheet['A3'] = '=SUM(A1:A2)'
>>> wb.save('writeFormula.xlsx')
```

A1 ve A2'deki hücreler sırasıyla 200 ve 300 yapılmıştır. A3 hücresindeki değer A1 ve A2'deki değerleri toplayan bir formüle ayarlanmıştır. Elektronik tablo Excel'de açıldığında A3 değerini 500 olarak görüntüleyecektir.

Excel formülleri elektronik tablolar için bir programlama düzeyi sunmaktadır ancak karmaşık görevlerde çabucak yönetilemez hâle gelebilirler. Örneğin, Excel formüllerine derinlemesine aşina olsanız bile *=IFERROR(TRIM(IF(LEN (VLOOKUP(F7, Sheet2!$A$1:$B$10000, 2, FALSE))>0,SUBSTITUTE(VLOOKUP(F7, Sheet2!$A$1:$B$10000, 2, FALSE), "", ""),"")), "")* formülünün ne yaptığını çözmeye çalışmak bir baş ağrısıdır. Python kodu çok daha okunaklıdır.

13.10 Satırları ve Sütunları Ayarlama

Excel'de, satırların ve sütunların boyutlarını ayarlamak, bir satırın veya sütunun başlığının kenarlarına tıklamak ve sürüklemek kadar basittir. Ancak hücrelerinin içeriğine göre bir satırın veya sütunun boyutunu ayarlamanız veya çok sayıda elektronik tabloda boyutları ayarlamanız gerekirse, bunu yapmak için Python kodu yazmak çok daha hızlı olacaktır.

Satırlar ve sütunlar da tamamen görünümden gizlenebilir. Veya her zaman ekranda görünür olmaları ve elektronik tablo yazdırıldığında her sayfada görünmeleri için yerinde "dondurulabilirler" (ki bu durum başlıklar için kullanışlıdır).

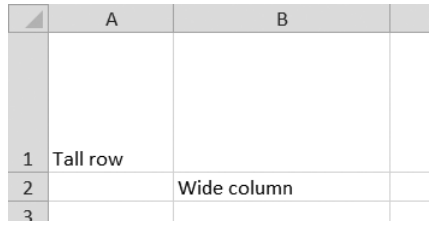
13.10.1 Satır Yüksekliği ve Sütun Genişliğini Belirleme

`Worksheet` nesnelerinin satır yüksekliğini ve sütun genişliğini kontrol eden `row_dimensions` (ÇN: Satır boyutları) ve `column_dimensions` (ÇN: Sütun boyutları) nitelikleri mevcuttur. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.active
>>> sheet['A1'] = 'Tall row'
>>> sheet['B2'] = 'Wide column'
# Yüksekliği ve genişliği belirle
>>> sheet.row_dimensions[1].height = 70
>>> sheet.column_dimensions['B'].width = 20
>>> wb.save('dimensions.xlsx')
```

Bir sayfanın `row_dimensions` ve `column_dimensions` nitelikleri sözlük benzeri değerlerdir. `row_dimensions`, `RowDimension` nesnelerini ve `column_dimensions` ise `ColumnDimension` nesnelerini içerir. `row_dimensions`'ta nesnelerden birine satırın numarasını kullanarak erişebilirsiniz (bu durumda 1 veya 2). `column_dimensions`'ta nesnelerden birine sütunun harfini kullanarak erişebilirsiniz (bu durumda A veya B).

dimensions.xlsx elektronik tablosu Şekil 13.6'daki gibidir.



Şekil 13.6: *Satır 1 ve Sütun B büyük yükseklik ve genişliklere ayarlanmıştır*

`RowDimension` nesnesine sahip olduktan sonra yüksekliğini belirleyebilirsiniz. `ColumnDimension` nesnesine sahip olduktan sonra genişliğini belirleyebilirsiniz. Satır yüksekliği, `0` ile `409` arasında bir tam sayıya veya kayan noktalı sayı değerine ayarlanabilir. Bu değer, *punto* ile ölçülen yüksekliği temsil eder. Bir punto 1/72 inç'e eşittir. Varsayılan satır yüksekliği 12,75 inçtir. Sütun genişliği `0` ile `255` arasında bir tam sayıya veya kayan noktalı sayı değere ayarlanabilir. Bu değer, bir hücrede görüntülenebilecek varsayılan yazı tipi boyutunda (11 punto) karakter sayısını temsil eder. Varsayılan sütun genişliği 8.43 karakterdir. 0 genişliğindeki sütunlar ve 0 yüksekliğindeki satırlar kullanıcıdan gizlenir.

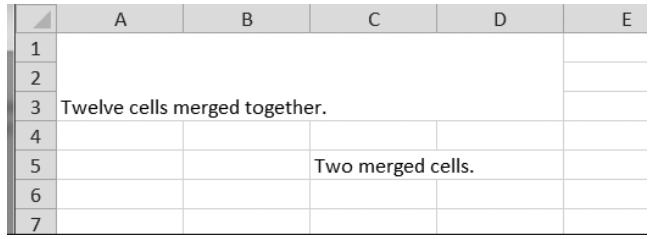
13.10.2 Hücreleri Birleştirme ve Ayırma

Dikdörtgen bir hücre alanı, `merge_cells()` sayfa yöntemiyle tek bir hücrede birleştirilebilir. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.active
# Bütün bu hücreleri birleştir.
>>> sheet.merge_cells('A1:D3')
>>> sheet['A1'] = 'Twelve cells merged together.'
# Bu iki hücreyi birleştir.
>>> sheet.merge_cells('C5:D5')
>>> sheet['C5'] = 'Two merged cells.'
>>> wb.save('merged.xlsx')
```

`merge_cells()`'in argümanı, birleştirilecek olan dikdörtgen şeklindeki alanın sol üst ve sağ alt hücrelerinin tek bir dizgisidir: `'A1:D3'` 12 hücreyi tek bir hücre olacak şekilde birleştirir. Bu birleştirilmiş hücrelerin değerini belirlemek için birleştirilmiş grubun sol üst hücresinin değerini belirleyeniz.

Bu kodu çalıştırdığınızda *merged.xlsx* Şekil 13.7'deki gibi olacaktır.



Şekil 13.7: *Bir elektronik tablodaki birleştirilmiş hücreler*

Hücreleri ayırmak (ÇN: Unmerge) için `unmerge_cells()` sayfa metodunu çağırınız. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('merged.xlsx')
>>> sheet = wb.active
# Bu hücreleri böl.
>>> sheet.unmerge_cells('A1:D3')
>>> sheet.unmerge_cells('C5:D5')
>>> wb.save('merged.xlsx')
```

Değişikliklerinizi kaydeder ve ardından elektronik tabloya bir göz atarsanız, birleştirilmiş hücrelerin ayrı ayrı hücrelere dönüştüğünü görürsünüz.

13.10.3 Bölmeleri Dondurma

Tek seferde görüntülenemeyecek kadar büyük elektronik tablolar için ekranın en üstteki satırlarından veya en soldaki sütunlardan birkaçını "dondurmak" yararlı olur. Örneğin, dondurulmuş sütun veya satır başlıkları, elektronik tabloda gezinirken bile kullanıcıya her zaman görünürdür. Bunlar bölmeleri dondur (ÇN: Freze panes) olarak bilinirler. OpenPyXL'de, her `Worksheet` nesnesinin bir `Cell` nesnesine veya bir hücrenin koordinat dizgisine ayarlanabilen bir `freeze_panes` niteliği vardır. Bu hücrenin üstündeki tüm satırların ve solundaki tüm sütunların dondurulacağını ancak hücrenin satır ve sütununun dondurulmayacağını unutmayınız.

Dondurulmuş bütün bölmeleri çözmek için `freeze_panes`'i None veya `'A1'` yapınız. Tablo 13.3 bazı örnek `freeze_panes` ayarları için hangi satır ve sütunların dondurulacağını göstermektedir.

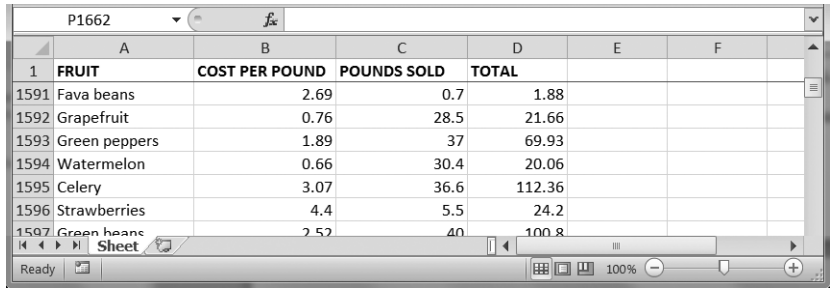
Tablo 13.3: Dondurulmuş Bölme Örnekleri

`freeze_panes` ayarları	**Dondurulan satırlar ve sütunlar**
`sheet.freeze_panes = 'A2'`	Satır 1
`sheet.freeze_panes = 'B1'`	Sütun A
`sheet.freeze_panes = 'C1'`	Sütun A ve B
`sheet.freeze_panes = 'C2'`	Satır 1 ve Sütun A ve B
`sheet.freeze_panes = 'A1'`	Dondurulmuş bölme yok
veya `sheet.freeze_panes = None`	

https://nostarch.com/automatestuff2/ adresindeki ürün satışları elektronik tablosuna sahip olduğunuzdan emin olunuz. Daha sonra aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('produceSales.xlsx')
>>> sheet = wb.active
# A2'nin üstündeki satırları dondur.
>>> sheet.freeze_panes = 'A2'
>>> wb.save('freezeExample.xlsx')
```

`freeze_panes` niteliğini `'A2'` yaparsanız, kullanıcı elektronik tablonun neresinde gezinirse gezinsin, satır 1 her zaman görünür olur. Bu durumu Şekil 13.8'de görebilirsiniz.



Şekil 13.8: *`freeze_panes 'A2'` yapıldığında, kullanıcı aşağı bile kaysa, satır 1 her zaman görünürdür*

13.11 Grafikler

OpenPyXL, bir sayfanın hücrelerindeki verileri kullanarak çubuk, çizgi, dağılım ve pasta grafikleri oluşturmayı destekler. Bir grafik oluşturmak için aşağıdakileri yapmanız gerekir:

1. Hücrelerin dikdörtgen şeklindeki bir seçiminden bir `Reference` nesnesi oluşturunuz.

2. `Reference` nesnesini göndermek suretiyle bir `Series` nesnesi oluşturunuz.

3. Bir `Chart` nesnesi oluşturunuz.

4. Series nesnesini `Chart` nesnesinin sonuna ekleyiniz.

5. Hangi hücrenin grafiğin sol üst köşesi olacağını isteğe bağlı olarak belirterek `Chart` nesnesini `Worksheet` nesnesine ekleyiniz.

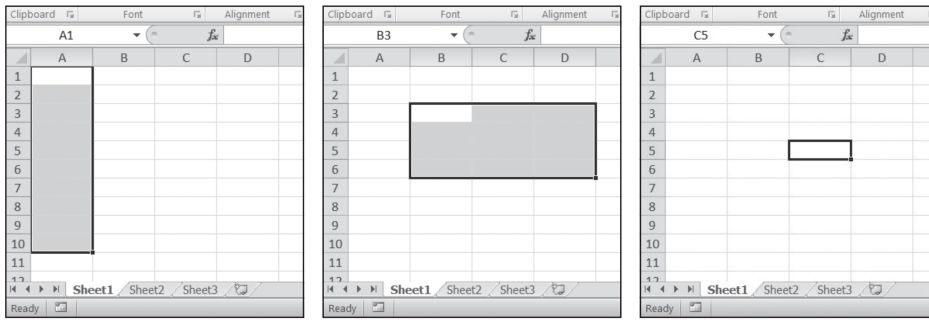
Reference nesnesinden biraz bahsetmek gerekir. Reference nesnelerini `open pyxl.chart.Reference()` fonksiyonunu çağırıp, üç argüman göndererek oluştursunuz:

1. Grafik verilerinizi içeren Worksheet nesnesi.

2. Grafik verilerinizi içeren dikdörtgen hücre seçiminin sol-üst hücresini temsil eden iki tam sayının bir demeti: Demetteki ilk tam sayı satır, ikincisi ise sütundur. 0'ın değil, 1'in ilk satır olduğuna dikkat ediniz.

3. Grafik verilerinizi içeren dikdörtgen hücre seçiminin sağ-alt hücresini temsil eden iki tam sayının bir demeti: Demetteki ilk tam sayı satır, ikincisi ise sütundur.

Şekil 13.9 bazı örnek koordinat argümanlarını göstermektedir.



Şekil 13.9: *Soldan sağa: (1, 1), (10, 1); (3, 2), (6, 4); (5, 3), (5, 3)*

Bir çubuk grafik oluşturmak ve bunu elektronik tabloya eklemek için bu etkileşimli kabuk örneğini giriniz:

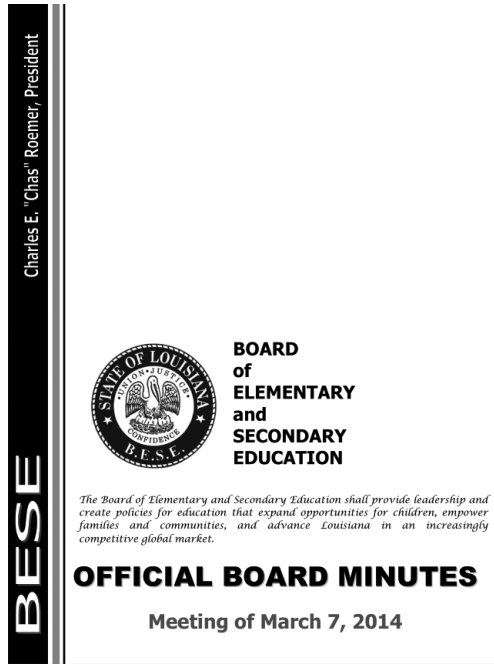
olduğundan emin olunuz. (Üçüncü-parti modülleri kurmakla ilgili tüm ayrıntılar için Ek A'ya bakınız.) Modül doğru bir şekilde kurulmuşsa etkileşimli kabukta `import PyPDF2` çalıştırıldığında herhangi bir hata görüntülenmemelidir.

PROBLEMLİ PDF FORMATI

PDF dosyaları, insanlar açısından yazdırması ve okuması kolay bir şekilde metin düzenlemek için harika olsa da yazılımın bu tür belgeleri düz metne ayrıştırması kolay değildir. Sonuç olarak, PyPDF2 bir PDF'ten metin çıkarırken hatalar yapabilir ve hatta bazı PDF'leri hiç açamayabilir. Bu konuda yapabileceğiniz pek bir şey yok maalesef. PyPDF2, belirli PDF dosyalarınızdan bazılarıyla çalışamayabilir. Bununla birlikte, şu ana kadar PyPDF2 ile açılamayan herhangi bir PDF dosyası bulamadım.

15.1.1 PDF'lerden Metin Çıkarma

PyPDF2'nin PDF belgelerinden görüntüleri, grafikleri veya diğer medya türlerini çıkarmasının bir yolu yoktur ancak metni çıkarabilir ve bir Python dizgisi olarak döndürebilir. PyPDF2'nin nasıl çalıştığını öğrenmeye başlamak için onu Şekil 15-1'de gösterilen örnek PDF'te kullanacağız..



Şekil 15.1: *Metin çıkaracağımız PDF sayfası*

Bu PDF'i ***https://nostarch.com/automatestuff2/*** adresinden indiriniz ve aşağıdakini etkileşimli kabuğa giriniz:

```
>>> import PyPDF2
>>> pdfFileObj = open('meetingminutes.pdf', 'rb')
>>> pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
>>> pdfReader.numPages ❶
19
>>> pageObj = pdfReader.getPage(0) ❷
>>> pageObj.extractText() ❸
'OOFFFFIICCIIAALL BBOOAARRDD MMIINNUUTTEESS Meeting of
   March 7, 2015 \n The Board of Elementary and
   Secondary Education shall provide leadership and
   create policies for education that expand
   opportunities for children, empower families and
   communities, and advance Louisiana in an increasingly
    competitive global market. BOARD of ELEMENTARY and
   SECONDARY EDUCATION '
>>> pdfFileObj.close()
```

İlk olarak PyPDF2 modülünü içe aktarınız. Ardından *meetingminutes.pdf* dosyasını ikili okuma modunda açınız ve dosyayı `pdfFileObj`'de saklayınız. Bu PDF'i temsil eden bir `PdfFileReader` nesnesi elde etmek için `PyPDF2.PdfFileReader()`'i çağırınız ve ona `pdfFileObj`'i gönderiniz. Bu `PdfFileReader`'ı `pdfReader`'da depolayınız.

Belgedeki toplam sayfa sayısı bir `PdfFileReader` nesnesinin `NumPages` niteliğinde saklıdır ❶. Örnek PDF'in 19 sayfası vardır ancak metni yalnızca ilk sayfadan çıkaralım.

Bir sayfadan metin çıkarmak için bir `PdfFileReader` nesnesinden bir `Page` nesnesi elde etmeniz gerekir. Bu nesne bir PDF'in tek bir sayfasını temsil eder. Bir `page` nesnesini bir `PdfFileReader` nesnesi üzerinde `getPage()` metodunu çağırarak ❷ ve bu metoda ilgilendiğiniz sayfa numarasını göndererek elde edebilirsiniz. Bizim durumumuzda sayfa numarası 0'dır.

PyPDF2, sayfaları elde etmek için sıfır tabanlı indeks kullanmaktadır: İlk sayfa 0'ıncı sayfa, ikinci sayfa 1'inci sayfadır ve bu böyle devam eder. Belge içinde sayfalar farklı şekilde numaralandırılsa bile bu her zaman bu şekildedir. Örneğin, PDF'iniz daha uzun bir rapordan üç sayfalık bir alıntıdır ve sayfaları 42, 43 ve 44 olarak numaralandırılmıştır. Bu belgenin ilk sayfasını almak için `getPage(42)` veya `getPage(1)` değil, `pdfReader.getPage(0)`'ı çağırmalısınız.

Page nesnesine sahip olduktan sonra sayfanın metninin dizgisini döndürmek için `Page` nesnesinin `extractText()` metodunu çağırınız ❸. Metin çekme işi mükemmel değildir: PDF'teki Charles E. "Chas" Roemer, President (ÇN: Charles E. "Chas" Roemer, Başkan) metni `extractText()` tarafından döndü-

rülen dizgide eksiktir ve boşluklar da bazen uygunsuzdur. Yine de PDF metin içeriğinin bu tahmini, programınız için yeterince iyi olabilir.

15.1.2 PDF'lerin Şifresini Çözme

Bazı PDF belgelerinin, belgeyi açan kişi bir şifre girene kadar okunmalarını engelleyen bir şifreleme özelliği vardır. İndirdiğiniz ve *rosebud* şifresiyle şifrelenmiş PDF ile etkileşimli kabuğa aşağıdakini giriniz:

```
>>> import PyPDF2
>>> pdfReader = PyPDF2.PdfFileReader(open('encrypted.pdf', 'rb'))
>>> pdfReader.isEncrypted ❶
True
>>> pdfReader.getPage(0)
Traceback (most recent call last): ❷
  File "<pyshell#173>", line 1, in <module>
    pdfReader.getPage()
--kesinti--
  File "C:\Python34\lib\site-packages\PyPDF2\pdf.py",
     line 1173, in getObject
    raise utils.PdfReadError("file has not been
       decrypted")
PyPDF2.utils.PdfReadError: file has not been decrypted
>>> pdfReader = PyPDF2.PdfFileReader(open('encrypted.pdf', 'rb'))
>>> pdfReader.decrypt('rosebud') ❸
1
>>> pageObj = pdfReader.getPage(0)
```

Bütün `PdfFileReader` nesneleri PDF şifreli olduğunda `True` ve şifreli olmadığında ise `False` olan bir `isEncrypted` niteliğine sahiptir ❶. Doğru şifre ile şifresi çözülmeden önce dosyayı okuyan bir fonksiyonu çağırmaya yönelik herhangi bir girişim bir hatayla sonuçlanacaktır ❷.

Not

PyPDF2 versiyon 1.26.0'daki bir hata nedeniyle, şifreli bir PDF üzerinde `decrypt()` çağrılmadan önce `getPage()`'in çağrılması, gelecekteki `getPage()` çağrılarının aşağıdaki hatayla başarısız olmasına neden olur: `IndexError: list index out of range` (ÇN: İndeksHatası: Liste indeksi aralık dışı). Bu nedenle örneğimiz dosyayı yeni bir `PdfFileReader` nesnesiyle yeniden açtı.

Şifrelenmiş bir PDF'i okumak için `decrypt()` fonksiyonunu çağırınız ve şifreyi bir dizgi olarak gönderiniz ❸. `decrypt()`'i doğru şifreyle çağırdıktan sonra `getPage()`'in artık hataya neden olmadığını göreceksiniz. Yanlış şifre

verilirse `decrypt()` fonksiyonu 0 döndürecek ve `getPage()` başarısız olmaya devam edecektir. `decrypt()` metodunun gerçek PDF dosyasının değil, sadece `PdfFileReader` nesnesinin şifresini çözdüğüne dikkat ediniz. Programınız sona erdikten sonra, sabit sürücünüzdeki dosya şifreli kalır. Programınız bir sonraki çalıştırılışında `decrypt()`'i tekrar çağırmak zorunda kalacaktır.

15.1.3 PDF'ler Yaratma

PyPDF2'nin `PdfFileReader`'ın karşılığı, yeni PDF dosyaları oluşturabilen `PdfFileWriter`'dır. Ancak PyPDF2, Python'un düz metin dosyalarıyla yapabileceği gibi bir PDF'e rastgele metin yazamaz. Bunun yerine, PyPDF2'nin PDF yazma yetenekleri, diğer PDF'lerden sayfa kopyalama, sayfaları döndürme, sayfaları üst üste yerleştirme ve dosyaları şifreleme ile sınırlıdır.

PyPDF2, bir PDF'i doğrudan düzenlemenize izin vermez. Bunun yerine, yeni bir PDF oluşturmanız ve ardından mevcut bir belgeden içerik kopyalamanız gerekir. Bu bölümdeki örnekler bu genel yaklaşımı izleyecektir:

1. Var olan bir veya daha fazla PDF'i (kaynak PDF'leri) `PdfFileReader` nesnesine açınız.
2. Yeni bir `PdfFileWriter` nesnesi oluşturunuz.
3. Sayfaları `PdfFileReader` nesnesinden `PdfFileWriter` nesnesine kopyalayınız.
4. Son olarak, çıktı PDF'ine yazmak için `PdfFileWriter` nesnesini kullanınız.

Bir `PdfFileWriter` nesnesi oluşturmak, yalnızca Python'da bir PDF belgesini temsil eden bir değer yaratır. Gerçek PDF dosyası oluşturmaz. Bunun için PdfFileWriter'ın `write()` metodunu çağırmalısınız.

`write()` metodu, *ikili yazma* (ÇN: write-binary) modunda açılmış normal bir `File` nesnesi alır. Böyle bir `File` nesnesini Python'un `open()` fonksiyonunu iki argümanla çağırarak elde edebilirsiniz: PDF'in dosya isminin ne olmasını istediğinizin dizgisi ve dosyanın ikili yazma modunda açılması gerektiğini belirten `'wb'` (ÇN: write-binary'nin baş harfleri).

Bu biraz kafa karıştırıcı geliyorsa endişelenmeyin; bunun nasıl çalıştığını aşağıdaki kod örneklerinde göreceksiniz.

Sayfaları Kopyalama

Sayfaları bir PDF belgesinden diğerine kopyalamak için PyPDF2'yi kullanabilirsiniz. Bu, birden çok PDF dosyasını birleştirmenize, istemediğiniz sayfaları kesmenize veya sayfaları yeniden sıralamanıza olanak tanır.

16.2.4 Benzer Programlar için Fikirler

CSV dosyaları için yazabileceğiniz programlar, her ikisi de elektronik tablo dosyaları olduğundan, Excel dosyaları için yazabileceğiniz türlere benzer. Aşağıdakileri yapmak için programlar yazabilirsiniz:

- Bir CSV dosyasındaki farklı satırlar arasındaki veya birden çok CSV dosyası arasındaki verileri karşılaştırmak.
- Bir CSV dosyasındaki belirli verileri bir Excel dosyasına kopyalamak veya tam tersi.
- CSV dosyalarındaki geçersiz verileri veya formatlama hatalarını kontrol etmek ve kullanıcıya bu hataları bildirmek.
- Bir CSV dosyasındaki verileri Python programlarınızın girdisi olarak okumak.

16.3 JSON ve API'ler

JavaScript Object Notation (ÇN: JavaScript Nesne Gösterimi); verileri, insan tarafından okunabilir tek bir dizgi olarak formatlamanın popüler bir yoludur. JSON, JavaScript programlarının veri yapılarını yazdığı özgün yoldur ve genellikle Python'un `pprint()` fonksiyonunun üreteceklerine benzemektedir. JSON formatlı verilerle çalışmak için JavaScript bilmenize gerek yoktur.

İşte JSON olarak formatlanmış bir veri örneği:

```
{"name": "Zophie",
"isCat": true,"miceCaught": 0,
"napsTaken": 37.5,
"felineIQ": null}
```

JSON'u bilmek faydalıdır çünkü birçok web sitesi, programların web sitesiyle etkileşime girmesi için bir yol olarak JSON içeriği sunar. Bu, *application programming interface* (API) (ÇN: uygulama programlama arayüzü) sağlama olarak bilinir. Bir API'ye erişmek, bir URL vasıtasıyla herhangi bir web sayfasına erişmekle aynıdır. Aradaki fark, bir API tarafından döndürülen verilerin makineler için biçimlendirilmesidir (örneğin JSON ile); API'leri insanların okuması kolay değildir.

Birçok web sitesi verilerini JSON formatında kullanıma sunmaktadır. Facebook, Twitter, Yahoo, Google, Tumblr, Wikipedia, Flickr, Data.gov, Reddit, IMDb, Rotten Tomatoes, LinkedIn ve diğer birçok popüler site, programların kullanması için API'ler sunar. Bu sitelerden bazıları kayıt olmayı gerektirmektedir ve bu işlem hemen hemen her zaman ücretsizdir. İstediğiniz verileri

almak için programınızın istemesi gereken URL'lerin yanı sıra döndürülen JSON veri yapılarının genel formatına ilişkin gerekli dokümantasyonu bulmanız gerekecektir. Bu dokümantasyonlar, API'yi hangi site sağlıyorsa o site tarafından sunulmalıdır. Eğer "Developers" (ÇN: Geliştirici) sayfası mevcut ise dokümantasyonu orada arayınız.

API'leri kullanarak aşağıdakileri yerine getiren programlar yazabilirsiniz:

- Web sitelerinden ham veri kazımak. (API'lere erişmek, web sayfalarını indirmek ve Beautiful Soup modülü ile HTML'i ayrıştırmaktan genellikle daha rahattır.)

- Yeni gönderileri sosyal ağ hesaplarınızın birinden otomatik olarak indirmek ve bu gönderileri başka bir hesaba postalamak. Örneğin, Tumblr gönderilerinizi alıp Facebook'ta yayınlayabilirsiniz.

- IMDb, Rotten Tomatoes ve Wikipedia'dan veriler çekip, bu verileri bilgisayarınızda tek bir metin dosyasına koyarak kişisel film koleksiyonunuz için bir "film ansiklopedisi" oluşturmak.

Örnek bazı JSON API'lerini ***https://nostarch.com/automatestuff2/*** adresindeki kaynaklarda görebilirsiniz.

JSON veriyi insan tarafından okunabilir bir dizgi olarak formatlamanın tek yolu değildir. XML (eXtensible Markup Language), TOML (Tom's Obvious, Minimal Language), YML (Yet another Markup Language), INI (Initialization), ve hatta modası geçmiş ASN.1 (Abstract Syntax Notation One) formatları da dâhil olmak üzere birçok format mevcuttur. Bu formatların hepsi de veriyi insan tarafından okunabilir bir metin olarak temsil etmek için bir yapı sağlar. Bu kitap diğer formatlardan bahsetmeyecektir çünkü JSON hızla en yaygın kullanılan alternatif bir format hâline gelmiştir. Ancak diğer formatları kolayca ele alan üçüncü-parti Python modülleri vardır.

16.4 `json` Modülü

Python'un `json` modülü, `json.loads()` ve `json.dumps()` fonksiyonları için JSON verileri içeren bir dizgi ile Python değerleri arasında çeviri yapmanın tüm ayrıntılarını ele alır. JSON, her tür Python değerini depolayamaz. Yalnızca şu veri tiplerinin değerlerini içerebilir: Dizgiler, tam sayılar, kayan değerler, Boole'lar, listeler, sözlükler ve `NoneType`. `JSON`, `File` nesneleri, CSV `reader` veya `writer` nesneleri, `Regex` nesneleri veya Selenium `WebElement` nesneleri gibi Python'a özgü nesneleri temsil edemez.

16.4.1 `loads()` Fonksiyonuyla JSON'u Okuma

JSON verisi içeren bir dizgiyi bir Python değerine çevirmek için o dizgiyi `json.loads()` fonksiyonuna gönderiniz. (Fonksiyonun ismi "loads" (ÇN: yükler) değil "load string" (ÇN: dizgiyi yükle)) anlamına gelmektedir.) Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> stringOfJsonData = '{"name": "Zophie", "isCat": true, "miceCaught":
    0,"felineIQ": null}'
>>> import json
>>> jsonDataAsPythonValue = json.loads(stringOfJsonData)
>>> jsonDataAsPythonValue
{'isCat': True, 'miceCaught': 0, 'name': 'Zophie', 'felineIQ':
    None}
```

`json` modülünü içe aktardıktan sonra, `loads()`'u çağırabilir ve ona JSON verisinin bir dizgisini gönderebilirsiniz. JSON dizgilerinin her zaman çift tırnak işareti kullandığına dikkat ediniz. `loads()` fonksiyonu bu veriyi bir Python sözlüğü olarak döndürür. Python sözlükleri sıralı değildir, dolayısıyla `jsonDataAsPythonValue` değişkenini ekrana yazdırdığınızda anahtar-değer çiftleri farklı bir sıralamada görünebilir.

16.4.2 `dumps()` Fonksiyonuyla JSON'u Yazma

`json.dumps()` fonksiyonu ("dumps" (ÇN: döker) değil, "dump string" (ÇN: dizgiyi dök) anlamına gelir.) bir Python değerini JSON formatlı bir veri dizgisine çevirir. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> pythonValue = {'isCat': True, 'miceCaught': 0, 'name':
    'Zophie','felineIQ': None}
>>> import json
>>> stringOfJsonData = json.dumps(pythonValue)
>>> stringOfJsonData
'{"isCat": true, "felineIQ": null, "miceCaught": 0, "name": "
    Zophie" }'
```

Değer yalnızca şu temel Python veri tiplerinden biri olabilir: Sözlük, liste, tam sayı, kayan değer, dizgi, Boole veya None.

16.5 Proje: Mevcut Hava Durumu Verilerini Alıp Getirme

Hava durumunu kontrol etmek oldukça sıradan bir iş gibi görünmektedir: Web tarayıcısını aç, adres çubuğuna tıkla, hava durumu web sitesinin URL'sini yaz (veya bir web sitesini ara ve bağlantıya tıkla), sayfanın yüklenmesini bekle, bütün reklamları görmezden gel, vesaire.

Uyarı

E-posta gönderen veya alan tüm komut dosyaları için ayrı bir e-posta hesabı oluşturmanızı şiddetle tavsiye ederim. Bu, programlarınızdaki hataların kişisel e-posta hesabınızı etkilemesini önleyecektir (örneğin, e-postaları silerek veya kişilerinize yanlışlıkla spam göndererek). E-postaları gerçekten gönderen veya silen kodu açıklamaya dönüştürerek ve bu kodu geçici bir `print()` *çağrısı ile değiştirerek ilk önce bir deneme çalıştırması yapmak iyi bir fikirdir. Bu şekilde, programınızı gerçek anlamda çalıştırmadan önce test edebilirsiniz.*

18.1 Gmail API'siyle E-posta Gönderme ve Alma

Gmail, e-posta istemcisi pazar payının üçte birine sahiptir ve kuvvetle muhtemel sizlerin en az bir Gmail e-posta adresi vardır. Ek güvenlik ve istenmeyen posta önleme önlemleri nedeniyle, bir Gmail hesabını EZGmail modülü aracılığıyla kontrol etmek, bu bölümün ilerleyen kısımlarında ele alınacak olan `smtplib` ve `imapclient` aracılığıyla kontrol etmekten daha kolaydır. EZGmail resmi Gmail API'sinin üzerinde çalışan, benim yazdığım bir modüldür ve Gmail'i Python'dan kullanmayı kolaylaştıran fonksiyonlar sağlar. EZGmail ile ilgili tüm ayrıntıları ***https://github.com/asweigart/ezgmail/*** adresinde bulabilirsiniz. EZGmail Google tarafından üretilmemiştir ve Google ile bağlantısı yoktur. Resmi Gmail API dokümantasyonunu ***https://developers.google.com/gmail/api/v1/reference/*** adresinde bulabilirsiniz.

EZGmail'i kurmak için Windows'ta `pip install --user --upgrade ezgmail` komutunu çalıştırınız (veya macOS ve Linux'ta pip3 kullanınız). `--upgrade` seçeneği, Gmail API gibi sürekli değişen bir çevrim içi hizmetle etkileşim kurmak için gerekli olan paketin en son versiyonunu yüklemenizi sağlayacaktır.

18.1.1 Gmail API'sini Etkinleştirme

Kod yazmadan önce ***https://gmail.com/*** adresinde bir Gmail e-posta hesabına kaydolmalısınız. Ardından ***https://developers.google.com/gmail/api/quickstart/python/*** adresine gidiniz ve bu sayfadaki Gmail API'sini Etkinleştir düğmesine tıklayınız ve açılan formu doldurunuz.

Bu formu doldurduktan sonra sayfa *credentials.json* dosyasına bir bağlantı verecektir. Bu dosyayı indirmeniz ve *.py* dosyanızla aynı klasöre yerleştirmeniz gerekmektedir. *credentials.json* dosyası Client ID'yi (ÇN: İstemci ID'si) ve Client Secret (ÇN: İstemci Parolası) bilgisini içerir. Client Secret'e Gmail şifreniz gibi muamele etmelisiniz ve kimseyle paylaşmamalısınız.

Daha sonra etkileşimli kabukta aşağıdaki kodu giriniz:

```
>>> import ezgmail, os
>>> os.chdir(r'C:\path\to\credentials_json_file')
>>> ezgmail.init()
```

Geçerli çalışma dizininizi *credentials.json*'un bulunduğu klasör ile aynı klasör olacak şekilde ayarladığınızdan ve internete bağlı olduğunuzdan emin olunuz. `ezgmail.init()` fonksiyonu tarayıcınızı Google oturum açma sayfasıyla açacaktır. Gmail adresinizi ve şifrenizi giriniz. Sayfa sizi "This app isn't verified" (ÇN: Bu uygulama doğrulanmamıştır) şeklinde uyarabilir fakat bu normaldir. Advanced'e (ÇN: Gelişmiş) tıklayınız ve ardından Go to Quickstart (unsafe)'e (ÇN: Hızlıbaşlat'a git (güvenilir değil)) tıklayınız. (Diğer şahıslar için Python betikleri yazıyorsanız ve bu uyarının onlar için görünmemesini istiyorsanız Google'ın uygulama doğrulama sürecini öğrenmeniz gerekmektedir. Bu ise bu kitabın kapsamı dışındadır.) Bir sonraki sayfa sizden "Quickstart wants to access your Google Account" (ÇN: Hızlıbaşlat Google hesabınıza erişmek istiyor) şeklinde bir istemde bulunduğunda Allow'a tıklayınız ve ardından tarayıcıyı kapatınız.

Python betiklerinizin girdiğiniz Gmail hesabına erişmesini sağlamak için bir *token.json* dosyası oluşturulacaktır. Tarayıcı var olan bir *token.json* dosyası bulamazsa sadece oturum açma sayfasını açacaktır. *credentials.json* ve *token.json* ile birlikte Python betikleriniz Gmail şifrenizi kaynak kodunuza eklemenize gerek kalmadan, Gmail hesabınızdan e-posta gönderebilir ve okuyabilir.

18.1.2 Bir Gmail Hesabından E-posta Gönderme

token.json dosyasına sahip olduktan sonra EZGmail modülü tek bir fonksiyon çağrısıyla e-posta gönderebilmelidir:

```
>>> import ezgmail
>>> ezgmail.send('recipient@example.com', 'Subject line', 'Body of
    the email')
```

E-postanıza dosya eklemek isterseniz `send()` (ÇN: Gönder) fonksiyonuna ekstra bir liste argümanı sağlayabilirsiniz:

```
>>> ezgmail.send('recipient@example.com', 'Subject line', 'Body of
    the email',['attachment1.jpg', 'attachment2.mp3'])
```

Güvenlik ve istenmeyen posta önleme özelliklerinin bir parçası olarak, Gmail'in tamamen aynı metne sahip e-postaları (bu e-postalar muhtemelen spam olacağından) veya *.exe* veya *.zip* dosya ekleri içeren e-postaları (bu dosyalar muhtemelen virüs olacağından) göndermeyebileceğine dikkat ediniz.

Karbon kopyaları veya görünmeyen karbon kopyaları göndermek için isteğe bağlı cc (ÇN: carbon copy, karbon kopya) veya bcc (ÇN: blind carbon copy, görünmeyen karbon kopya) anahtar kelime argümanlarını da sağlayabilirsiniz:

```
>>> import ezgmail
>>> ezgmail.send('recipient@example.com', 'Subject line', 'Body of
    the email',cc='friend@example.com',
    bcc='otherfriend@example.com,someoneelse@example.com')
```

token.json dosyasının hangi Gmail adresi için yapılandırıldığını hatırlamanız gerekirse `ezgmail.EMAIL_ADDRESS`'i inceleyebilirsiniz. Bu değişkenin sadece `ezgmail.init()`'ten sonra veya herhangi bir EZGmail fonksiyonu çağrıldıktan sonra doldurulacağına dikkat ediniz:

```
>>> import ezgmail
>>> ezgmail.init()
>>> ezgmail.EMAIL_ADDRESS
'example@gmail.com
```

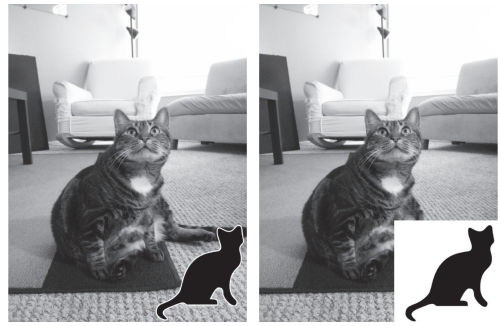
token.json dosyasına şifrenizle muamele ettiğiniz gibi muamele ettiğinizden emin olunuz. Bir başkası bu dosyayı elde ederse Gmail hesabınıza erişebilir (ancak Gmail şifrenizi değiştiremez). Daha önceden verilen *token.json* dosyalarının izinlerini iptal etmek için ***https://security.google.com/settings/security/permissions?pli=1/*** adresine gidiniz ve Quickstart uygulamasına olan erişim iznini iptal ediniz. Yeni bir *token.json* dosyası elde etmek için `ezgmail.init()`'i çalıştırmanız ve oturum açma sürecini tekrarlamanız gerekecek.

18.1.3 Bir Gmail Hesabından E-posta Okuma

Gmail, birbirine yanıt olan e-postaları konuşma dizileri (ÇN: threads) hâlinde düzenler. Web tarayıcınızda veya bir uygulama aracılığıyla Gmail'de oturum açtığınızda, bireysel e-postalar yerine gerçekten e-posta dizilerine bakıyorsunuz (dizide yalnızca bir e-posta olsa bile).

EZGmail, sırasıyla konuşma dizilerini ve bireysel e-postaları temsil eden `GmailThread` ve `GmailMessage` nesnelerine sahiptir. `GmailThread` nesnesi `GmailMessage` nesnelerinin listesini tutan bir `messages` niteliğine sahiptir. `unread()` fonksiyonu, tüm okunmamış e-postalar için `GmailThread` nesnelerinin bir listesini döndürür ve bu listedeki konuşma dizilerinin bir özetini yazdırmak için `ezgmail.summary()`'e gönderebilir:

```
>>> import ezgmail
>>> unreadThreads = ezgmail.unread()
# GmailThread nesnelerinin listesi
>>> ezgmail.summary(unreadThreads)
Al, Jon - Do you want to watch RoboCop this weekend? -
    Dec 09
Jon - Thanks for stopping me from buying Bitcoin. - Dec
    09
```



Şekil 19.13: *zophie.png görüntüsünün yeniden boyutlandırılmış ve logo eklenmiş hâli (solda). Üçüncü argümanı unutursanız logodaki saydam pikseller katı beyaz pikseller olarak kopyalanacaktır (sağda)*

19.4 Görüntülerin Üzerine Çizme

Eğer bir görüntünün üzerine çizgiler, dikdörtgenler, çemberler veya diğer basit şekiller çizmek isterseniz Pillow'un `ImageDraw` modülünü kullanınız. Aşağıdakini etkileşimli kabuğa giriniz:

```
>>> from PIL import Image, ImageDraw
>>> im = Image.new('RGBA', (200, 200), 'white')
>>> draw = ImageDraw.Draw(im)
```

Öncelikle `Image` ve `ImageDraw`'u içe aktarıyoruz. Ardından yeni bir görüntü oluşturuyoruz ki bu durumda 200x200'lük beyaz bir görüntüdür bu ve `Image` nesnesini `im`'de depoluyoruz. Bir `ImageDraw` nesnesi alabilmek için `Image` nesnesini `ImageDraw.Draw()` fonksiyonuna gönderiyoruz. Bu nesne bir `Image` nesnesi üzerine metin ve şekiller çizmek için çeşitli metotlara sahiptir. Takip eden örnekte kolayca kullanabilmek için `ImageDraw` nesnesini `draw` gibi bir değişkende depolayınız.

19.4.1 Şekiller Çizmek

Aşağıdaki `ImageDraw` metotları görüntü üzerinde çeşitli şekiller çizer. Bu metotların `fill` ve `outline` parametreleri isteğe bağlıdır ve belirtilmezlerse varsayılan olarak beyaz olurlar.

Noktalar

`point(xy, fill)` metodu tek tek pikseller çizer. `xy` argümanı çizmek istediğiniz noktaların bir listesini temsil eder. Liste, `[(x, y), (x, y), ...]` gibi x ve y

koordinat demetlerinin bir listesi olabilir veya `[x1, y1, x2, y2, ...]` gibi x ve y koordinatlarının demetsiz bir listesi olabilir. `fill` argümanı noktaların rengidir ve ya bir RGBA demetidir ya da `'red'` gibi bir renk isminin bir dizgisidir. `fill` argümanı isteğe bağlıdır.

Çizgiler

`line(`*`xy`*`, `*`fill`*`, `*`width`*`)` bir çizgi veya çizgi serisi çizer. xy, ya `[(x, y), (x, y), ...]` gibi demetlerin bir listesi ya da `[x1, y1, x2, y2, ...]` gibi tam sayıların bir listesidir. Her nokta, çizdiğiniz çizgiler üzerindeki bağlantı noktalarından biridir. İsteğe bağlı fill argümanı, bir RGBA demeti veya renk ismi olarak çizgilerin rengidir. İsteğe bağlı `width` argümanı, çizgilerin genişliğidir ve belirtilmemişse varsayılan olarak 1'dir.

Dikdörtgenler

`rectangle(`*`xy`*`, `*`fill`*`, `*`outline`*`)` bir dikdörtgen çizer. `xy` argümanı, `(`*`left`*`, `*`top`*`, `*`right`*`, `*`bottom`*`)` biçiminde bir kutu demetidir. left ve top değerleri dikdörtgenin sol üst köşesinin x ve y koordinatlarını belirler. `right` ve `bottom` ise dikdörtgenin sağ alt köşesini belirler. İsteğe bağlı `fill` argümanı dikdörtgenin içerisini dolduracak olan renktir. İsteğe bağlı `outline` argümanı dikdörtgenin ana hatlarının rengidir.

Elipsler

`ellipse(`*`xy`*`, `*`fill`*`, `*`outline`*`)` metodu bir elips çizer. Elipsin genişliği ve yüksekliği aynıysa, bu yöntem bir daire çizecektir. xy argümanı, tam olarak elipsi içeren bir kutuyu temsil eden `(`*`left`*`, `*`top`*`, `*`right`*`, `*`bottom`*`)` şeklinde bir kutu demetidir. İsteğe bağlı `fill` argümanı elipsin içinin rengi, isteğe bağlı `outline` argümanı ise elipsin ana hatlarının rengidir.

Poligonlar

`polygon(`*`xy`*`, `*`fill`*`, `*`outline`*`)` metodu keyfi bir poligon çizer. `xy` argümanı, poligonun yanlarının bağlantı noktalarını temsil eden `[(x, y), (x, y), ...]` gibi demetlerin bir listesi veya `[x1, y1, x2, y2, ...]` gibi tam sayıların bir listesidir. Son koordinat çifti, otomatik olarak ilk koordinat çiftine bağlanacaktır. İsteğe bağlı `fill` argümanı poligonun içinin rengidir, isteğe bağlı `outline` argümanı ise poligonun ana hatlarının rengidir.

Çizim Örneği

Aşağıdakini etkileşimli kabuğa giriniz:

PyGetWindow modülünün dokümantasyonu ***https://pygetwindow.readthedocs.io/*** adresindedir.

20.10 Klavyeyi Kontrol Etme

PyAutoGUI ayrıca, formları doldurmanızı veya uygulamalara metin girmenizi sağlayan, bilgisayarınıza sanal tuş basmaları gönderme fonksiyonlarına da sahiptir.

20.10.1 Klavyeden Bir Dizgi Gönderme

`pyautogui.write()` fonksiyonu, bilgisayarınıza sanal tuş basışları gönderir. Bu tuşa basmaların ne yaptığı, hangi pencerenin etkin olduğuna ve hangi metin alanının odaklandığına bağlıdır. Odaklandığından emin olmak için önce istediğiniz metin alanına bir fare tıklaması göndermek isteyebilirsiniz.

Basit bir örnek olması açısından bir dosya düzenleyici penceresine otomatik olarak `Hello, world!` kelimelerini yazmak için Python'u kullanalım. İlk olarak, yeni bir dosya düzenleyici penceresi açınız ve onu ekranınızın sol üst köşesine konumlandırınız, böylece PyAutoGUI onu odaklamak için doğru yere tıklayacaktır. Daha sonra aşağıdakini etkileşimli kabuğa giriniz:

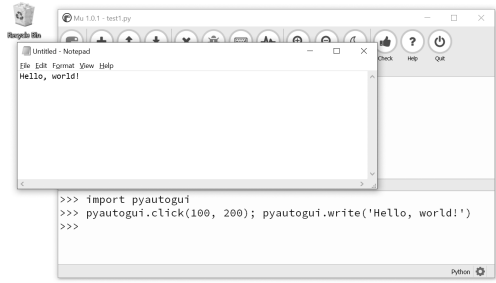
```
>>> pyautogui.click(100, 200); pyautogui.write('Hello, world!')
```

Noktalı virgülle ayırarak aynı satıra iki komutu yerleştirmenin, etkileşimli kabuğun bu iki talimatı çalıştırma arasında sizden girdi istemesini nasıl engellediğine dikkat ediniz. Bu, `click()` ile `write()` çağrıları arasında kazara yeni bir pencereyi odağa getirmenize engel olur. Böyle bir pencerenin odağa gelmesi bu örneği berbat ederdi.

Python öncelikle (100, 200) koordinatlarına sanal bir fare tıklaması gönderecektir. Bunun dosya düzenleyici penceresine tıklaması ve onu odağa getirmesi gerekir. `write()` çağrısı, `Hello, world!` metnini pencereye gönderecektir ve pencerenin Şekil 20.6'daki gibi görünmesini sağlayacaktır. Artık sizin için yazı yazabilecek bir kodunuz var!

Varsayılan olarak `write()` fonksiyonu dizginin tamamını anında yazacaktır. Ancak her bir karakter arasına kısa bir duraklama eklemek için isteğe bağlı ikinci bir argüman gönderebilirsiniz. Bu ikinci argüman, duraklatılacak saniye sayısının bir tam sayı veya kayan nokta değeridir. Örneğin, `pyautogui.write('Hello, world!', 0.25)`, H'yi yazdıktan sonra çeyrek saniye bekleyecek, e'yi yazdıktan sonra yine çeyrek saniye bekleyecek ve bu böyle devam edecektir. Bu kademeli daktilo efekti, tuş vuruşlarını PyAutoGUI'ye ayak uyduracak kadar hızlı işleyemeyen daha yavaş uygulamalar için faydalı olabilir.

A veya *!* gibi karakterler için PyAutoGUI otomatik olarak `SHIFT` tuşunu basılı tutmayı da simüle edecektir.



Şekil 20.6: *Dosya düzenleyici penceresine tıklamak ve bu pencereye Hello, world! yazmak için PyAutoGUI fonksiyonunu kullanmak*

20.10.2 Tuş İsimleri

Tüm tuşların tek metin karakterleriyle gösterilmesi kolay değildir. Örneğin, `SHIFT` veya sol ok tuşunu tek karakter olarak nasıl temsil edersiniz? PyAutoGUI'de bu klavye tuşları, bunun yerine kısa dizgi değerleriyle temsil edilirler: `ESC` tuşu için `'esc'` veya `ENTER` tuşu için `'enter'`.

Tek bir dizgi argümanı yerine, `write()`'a bu klavye tuş dizgilerinin bir listesi gönderilebilir. Örneğin, aşağıdaki çağrı önce A tuşuna, sonra B tuşuna, ardından iki kere sol ok tuşuna ve son olarak X ve Y tuşlarına basar:

```
>>> pyautogui.write(['a', 'b', 'left', 'left', 'X', 'Y'])
```

Sol ok tuşuna basmak klavye imlecini hareket ettireceğinden bu, XYab şeklinde bir çıktı verecektir. Tablo 20.1 herhangi bir tuş kombinasyonunu simüle etmek için `write()`'a gönderebileceğiniz PyAutoGUI klavye tuş dizgilerini listelemektedir.

PyAutoGUI'ın kabul edeceği mümkün olan bütün klavye tuş dizgilerini görmek için `pyautogui.KEYBOARD_KEYS` listesini de inceleyebilirsiniz. `'shift'` dizgisi sol `SHIFT` tuşuna işaret eder ve `'shiftleft'` ile aynıdır. Aynı şey `'ctrl'`, `'alt'` ve `'win'` dizgileri için de geçerlidir; hepsi de sol taraftaki tuşlara atıfta bulunur.

20.10.3 Klavyeye Basma ve Klavyeyi Serbest Bırakma

`mouseDown()` ve `mouseUp()` fonksiyonlarına çok benzer bir şekilde, `pyautogui.keyDown()` ve `pyautogui.keyUp()` fonksiyonları da bilgisayarınıza sanal tuş basışları ve serbest bırakışları gönderir. Argüman olarak bu fonksiyonlara

bir klavye tuş dizgisi gönderilir (bakınız Tablo 20.1). Kolaylık sağlamak için PyAutoGUI tam bir tuşa basmayı simüle etmek için bu fonksiyonların her ikisini de çağıran `pyautogui.press()` fonksiyonunu sağlar.

Tablo 20.1: `PyKeyboard` Nitelikleri

Klavye tuş dizgisi	**Anlamı**
`'a'`, `'b'`, `'c'`, `'A'`, `'B'`, `'C'`, `'1'`, `'2'`, `'3'`,`'!'`, `'@'`, `'#'` ve diğerleri	Tek karakterler için olan tuşlar
`'enter'` (veya `'return'` veya `'\n'`)	`ENTER` tuşu
`'esc'`	`ESC` tuşu
`'shiftleft'`, `'shiftright'`	Sol ve sağ `SHIFT` tuşları
`'altleft'`, `'altright'`	Sol ve sağ `ALT` tuşları
`'ctrlleft'`, `'ctrlright'`	Sol ve sağ `CTRL` tuşları
`'tab'` (veya `'\t'`)	`TAB` tuşu
`'backspace'`, `'delete'`	`BACKSPACE` ve `DELETE` tuşları
`'pageup'`, `'pagedown'`	`PAGE UP` ve `PAGE DOWN` tuşları
`'home'`, `'end'`	`HOME` ve `END` tuşları
`'up'`, `'down'`, `'left'`, `'right'`	Yukarı, aşağı, sol ve sağ ok tuşları
`'f1'`, `'f2'`, `'f3'` ve diğerleri	`F1`'den `F12`'ye kadar olan tuşlar
`'volumemute'`, `'volumedown'`, `'volumeup'`	mute, volume down ve volume up tuşları
`'capslock'`, `'numlock'`, `'scrolllock'`	`CAPS LOCK`, `NUM LOCK` VE `SCROLL LOCK` tuşları
`'insert'`	`INS` veya `INSERT` tuşu
`'printscreen'`	`PRTSC` veya `PRINT SCREEN` tuşu
`'winleft'`, `'winright'`	Sol veya sağ `WIN` tuşları (Windows'ta)
`'command'`	Komut (⌘) tuşu (macOS'ta)
`'option'`	`OPTION` tuşu (macOS'ta)

Dolar işareti karakteri (`SHIFT` tuşuna basılı tutup 4'e basmakla elde edilir) yazan aşağıdaki kodu çalıştırınız:

```
>>> pyautogui.keyDown('shift'); pyautogui.press('4');
    pyautogui.keyUp('shift')
```

Bu satır `SHIFT`'e basar, 4'e basar (ve serbest bırakır) ve ardından `SHIFT`'i serbest bırakır. Eğer bir metin alanına bir dizgi yazmanız gerekiyorsa `write()` fonksiyonu daha uygundur. Ancak tek tuşluk komutlar alan uygulamalar için `press()` fonksiyonu daha basit bir yaklaşımıdır.

20.10.4 Kısayol Tuş Kombinasyonları

Kısayol tuşu (ÇN: hotkey) veya *kısayol* (ÇN: shortcut), bazı uygulama fonksiyonlarını çağırmak için bir tuşa basma kombinasyonudur. Seçilmiş bir şeyi kopyalamak için olan ve sık kullanılan kısayol (Windows'ta) `CTRL-C` veya (macOS'ta) ⌘-C'dir. Kullanıcı `CTRL` tuşunu basılı tutar, daha sonra `C` tuşuna basar ve ardından `C` ve `CTRL` tuşlarını serbest bırakır. Bunu PyAutoGUI'ın `keyDown()` ve `keyUp()` fonksiyonlarıyla yapmak için aşağıdakini girmeniz gerekir.

Dizin