Python Nedir?

Python 1990 yılında Guido van Rossum tarafından Amsterdam'da geliştirilmeye başlanmış nesne yönelimli, yorumlanabilen, modüler ve etkileşimli bir programlama dilidir.
Pek çok insan Python isminin piton yılanından geldiğini düşünse de aslında durum böyle değildir. Python geliştiricisi Guido van Rossum, programlama diline adını The Monty Python isimli bir İngiliz komedi grubunun Monty Python's Flying Circus adlı gösterisinden esinlenerek vermiştir. Her ne kadar olayın aslı böyle olsa da Python programlama dilinin yılan figürü ile temsil edilmesi bir gelenek haline gelmiştir.

Neden Python?

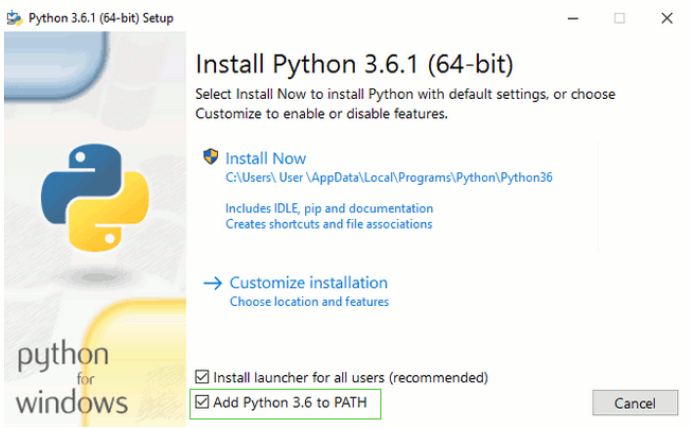
- Python dili C, C++ gibi dillerin aksine Interpreter, yani yorumlayıcı bir dildir. Dolayısıyla derlemeye gerek kalmadan çalıştırabilir ve bu sayede çok hızlı bir biçimde uygulama geliştirebilirsiniz.
- Eğer herhangi bir programlama dili biliyorsanız, Python'u öğrenme hızınız neredeyse okuma hızınızla doğru oranda olacaktır.
- Python'un temiz söz dizimi sayesinde Python'da program yazmak veya bir başkasının yazdığı programı okumak diğer dillere nispeten çok daha kolaydır.
- Ek olarak söz dizimi sadece girintilere bağlı olduğu için uygulama geliştiricileri söz dizimi ile uğraşarak vakit kaybetmezler.
- Python çapraz (cross) platform desteği syesinde birçok sistem üzerinde çalıştırılabilir. Pek çok Linux dağıtımının içerisinde Python 2.x sürümü yüklü gelmektedir. Ayrıca ülkemizde TUBİTAK tarafından geliştirilen Linux dağıtımı Pardus'un da bel kemiğini yine Python oluşturmaktadır. Popüler Linux dağıtımları da Python'u çeşitli uygulamalarını geliştirmek için kullanmaktadırlar. (Örnek olarak; Ubuntu Software Center)

- Python dünya çapında büyük üne sahip Google, Youtube, Yahoo! gibi şirketler tarafından yazılım geliştirmek için kullanılmaktadır. Ayrıca Google, ileri düzeyde Python bilgisine sahip kişilere iş imkanları sağlamaktadır. Python geliştiricisi Guido van Rossum 2005 – 2012 yılları arasında Google'da çalışmıştır.
- Python kullanarak masaüstü programlama, oyun programlama, taşınabilir cihaz programlama, web programlama ve ağ programlama çalışmaları rahatlıkla yürütülebilir.
- Python ile programlama yaparken kullanabileceğiniz pek çok IDE (Integrated Development Enviroment – Tümleşik Geliştirme Ortamı) mevcuttur. Bunlardan bazıları:
  Eclipse, Pydev, Eric, Komodo IDE, PyCharm
- Python insan beynindeki düşünme olayı temel alınarak tasarlanmıştır. Yani bir şeyin nasıl olması gerektiğini düşünüyorsanız, Python o şekilde gerçeklenimini sağlar.
- Python, içerisinde barındırdığı Garbage Collector (çöp toplayıcı) sayesinde uygulamanızın bellek kullanımını optimize eder. Bu durum uygulamanızın kararlılığını ve performansını arttıracaktır.
- Python, tamamı ile nesne yönelimli bir programlama dilidir. Popüler OOP dilleri ile yarışacak seviyede bir altyapıya sahiptir.
- Python, Java ve .NET platformları ile entegre biçimde çalışma yeteneğine sahiptir.
- Hepsinden önemlisi Python "özgür" bir dildir.

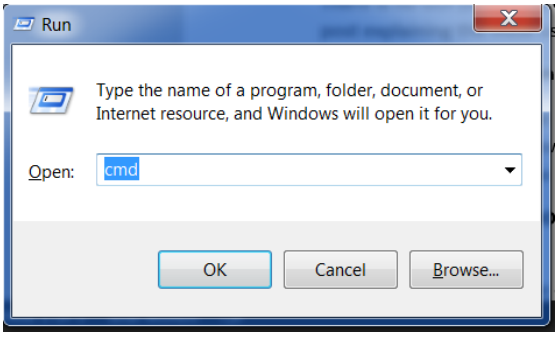
Windows için Python kurulumu

Öncelikle Windows + Pause/Break tuş kombinasyonlarına basarak bilgisayarınızın Windows'un 32-bit versiyonunu mu 64-bit versiyonu mu çalıştırdığını kontrol edin. Bu size Sistem bilgisini açacak. Açılan bu pencerede "Sistem tipi" satırına bakın. Windows için Python'ı indirmek için resmi siteyi ziyaret edebilirsiniz: https://www.python.org/downloads/windows/. "Son Python 3 Sürümü - Python x.x.x" bağlantısına tıklayın. Eğer bilgisayarınız **64-bit** versiyon Windows çalıştırıyorsa, **Windows x86-64 çalıştırılabilir yükleyici**'yi indirin. Değilse, **Windows x86 çalıştırılabilir yükleyici**'yi indirin. Yükleyiciyi indirdikten sonra, çalıştırmalısınız (üzerine çift tıklayarak) ve oradaki talimatları takip etmelisiniz.

Dikkat edilmesi gereken bir şey: Yükleme esnasında "Setup" ("Kur") işaretli bir pencere farkedeceksiniz. "Add Python 3.6 to PATH" ("PATH'e Python 3.6'yı ekle) kutucuğunun işaretli olduğundan emin olun ve aşağıda gösterildiği gibi "Install Now" ("Şimdi Yükle") 'a tıklayın:



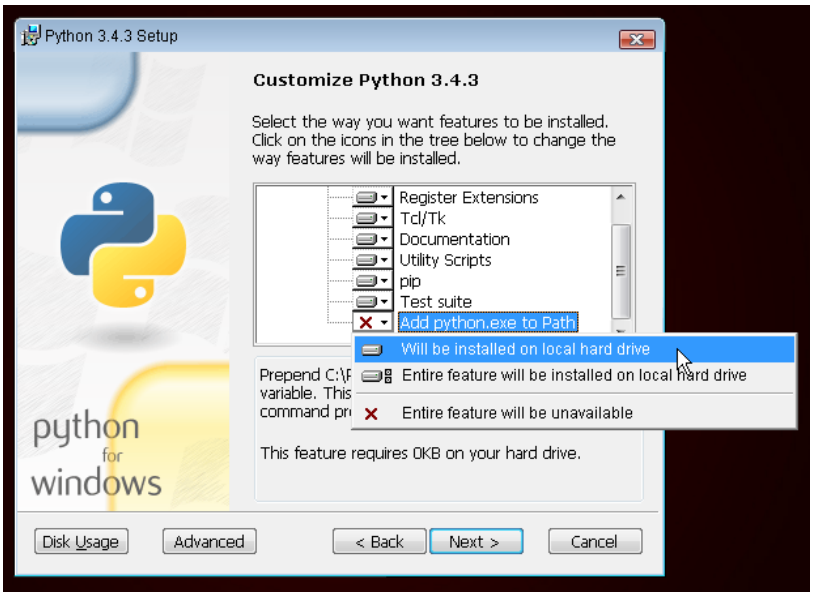
Önümüzdeki adımlarda, Windows Komut Satırını kullanıyor olacaksınız (ki ayrıca bahsedeceğiz). Şimdilik, bazı komutlar girmeniz gerekirse, Başlat menüsüne gidin ve arama alanına "Komut istemi" yazıp enter'a basın. (Windows'un eski sürümlerinde, komut satırını Başlat menüsü → Windows sistemi → Komut istemi ile başlatabilirsin.) Ayrıca "Çalıştır" penceresi açılana kadar Windows tuşunu basılı tutup "R"-tuşuna basabilirsin. Komut Satırı'nı açmak için, "cmd" yazın ve "Çalıştır" penceresinde enter'a basın.



Not: eğer Windows'un eski bir versiyonunu (7, Vista ya da herhangi bir eski versiyon) kullanıyor sanız ve Python 3.6.x yükleyicisi hata veriyorsa, aşağıdakilerden birini deneyebilirsiniz:

1. bütün Windows Güncellemeleri'ni yükleyin ve Python 3.6'yı tekrar yüklemeyi deneyin; ya da
2. eski bir Python versiyonu, örneğin 3.4.6 yükleyin.

Eğer Python'ın eski bir versiyonunu yüklerseniz, yükleme ekranı yukarıda gösterilenden biraz farklı görünebilir. "python.exe'yi Path'e ekle" görene dek aşağı kaydırdığınızdan emin olun, daha sonra soldaki butona tıklayın ve "Yerel sabit sürücüye yüklenecek"'i seçin:



Mac İçin Python Kurulumu:

Not Python'ı OS X'te yüklemeden önce, Mac ayarlarınızın App Store'dan olmayan paketleri yüklemeye izin verdiğinden emin olmalısınız. Sistem Tercihleri'ne (Uygulamalar klasöründe) gidin, önce "Güvenlik & Gizlilik"'e ve daha sonra da "Genel" sekmesine tıklayın. Eğer sizin "Şuradan yüklenen uygulamalara izin ver:" ayarınız "Mac App Store"'a ayarlıysa, onu "Mac App Store and kimliği bilinen geliştirici."lere çevirin

Python kurulum dosyasını indirmek için resmi siteye gitmelisiniz: https://www.python.org/downloads/release/python-361/

- *Mac OS X 64-bit/32-bit yükleyici* dosyasını indirin,
- *python-3.6.1-macosx10.6.pkg* 'a çift tıklayarak yükleyiciyi çalıştırın.

Linux için Python Kurulumu:

Muhtemelen sisteminizde Python zaten yüklüdür. Yüklü olup olmadığını (ya da hangi versiyon olduğunu) kontrol etmek için komut satırını açın ve aşağıdaki komutları girin:

komut satırı

```
$ python3 --version
Python 3.6.1
```

Eğer farklı bir 'mikro sürüm' Python yüklüyse, örn: 3.6.0, o zaman yükseltme yapmak zorunda değilsiniz. Python yüklü değilse ya da farklı bir versiyon edinmek istiyorsanız aşağıdaki adımları takip edin:

Kod Editörü Seçimi:

Birçok farklı kod editörü var, hangi editörü kullanacağınız kişisel tercihinize bağlı. Çoğu Python programcısı PyCharm gibi karmaşık fakat son derece güçlü IDE'leri (Integrated Development Environments-Entegre Geliştirme Ortamları) kullanır. Başlangıç seviyesi için bunlar muhtemelen pek uygun olmayacaktır. Bizim önerdiklerimiz aynı derecede güçlü fakat çok daha basit editörler olacak.

Bizim tavsiyelerimiz aşağıdakiler, ama mentörünüze danışmak isteyebilirsiniz.

Gedit

Gedit açık kaynaklı, ücretsiz bir editördür. Tüm işletim sistemlerinde kullanılabilir.

Buradan indirin

Sublime Text 3

Sublime Text, ücretsiz deneme süresi olan çok popüler bir editördür ve tüm işletim sistemleri için mevcuttur.

Buradan indirin

Atom

Atom başka bir popüler editör. Ücretsiz, açık kaynaklı ve Windows, OS X ve Linux için kullanılabilir. Atom GitHub </ 0> tarafından geliştirilmiştir.</p>

Buradan indirin

Neden bir kod editörü kuruyoruz?

Neden Word ya da Notepad kullanmak yerine, özel bir kod editörü yazılımı kurduğumuzu merak ediyor olabilirsiniz.

Birinci nedeni yazdığımız kodun **sade yazı** olması gerektiği içindir ki Word gibi programlar sade yazı üretmezler (font ve şekillendirme desteklerler) ve RTF (Rich Text Format) gibi özel formatlar kullanılar.

İkinci sebep ise kod editörlerinin kod düzenlemeye destek vermek - örneğin özel kelimeleri renklendirme ve otomatik parantez kapatma gibi - için özelleşmiş olmasıdır.

Python'a giriş

Python komut istemi (prompt)

Komut satırı nedir?

Genellikle **komut satırı** veya **komut satırı arabirimi** adı verilen pencere, bilgisayarınızdaki dosyaları görmek, düzenlemek ve yönetmek için kullanılan metin tabanlı bir uygulamadır. Bu tıpkı windows gezgini yada mac'teki finder gibi fakat grafiksel arayüzü olmadan. Komut satırının diğer adları: *cmd*, *CLI*, *komut istemcisi*, *konsol* veya *terminal (uçbirim)*dir.

Komut satırı arabirimini açın

Birkaç deneme yapmak için önce komut satırı arabirimini açmamız gerekir.

Windows'da komut satırını açmak

Başlat'a gidin → Windows sistemi → Komut istemcisi.

Daha eski bir windows sistemi için, Başlat menüsü →Tüm programlar → Aksesuarlar → Komut istemcisi.

Mac için komut satırını açmak

Uygulamalar → Araçlar →Terminal.

Linux'da komut satırını açmak

Muhtemelen Uygulamalar → Donatılar → Terminal altında olmalı, fakat sistemler arası farklılık gösterebilir. Eğer orada değilse İnternet'te arayın. :)

Hazır olduğunuzda, aşağıdaki talimatları takip edin.

Bir Python konsolu açmak istiyoruz; öyleyse Windows'ta `python`, Mac OS/Linux'ta `python3` yazıp, `enter`'a basın.
komut satırı

```
$ python3
Python 3.6.1 (...)
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

İlk Python komutunuz!

Python komutunu çalıştırdıktan sonra, komut istemi `>>>` şeklinde değişti. Bizim için bu, şimdi yalnızca Python dilinde komutlar kullanabileceğimiz anlamına geliyor. `>>>` yazmanıza gerek yok - Python bunu sizin için yapacak.
Eğer herhangi bir zamanda Python komut satırından çıkmak isterseniz, yalnızca `exit()` yazmanız ya da Windows için Ctrl + Z, Mac/Linux için Ctrl + D kısa yolunu kullanmanız yeterli. Bunu yaptığınız taktirde artık `>>>` yazısını görmeyeceksiniz.
Şimdilik, Python konsolundan çıkmak istemiyoruz. Bu konuda daha fazla bilgi edinmek istiyoruz. Biraz matematik yazarak başlayalım (2 + 3 gibi) ve `enter` 'a basalım.
komut satırı

```
>>> 2 + 3
5
```

Güzel! Cevap nasıl da çıktı görüyor musun? Python matematik biliyor! Sen de diğer komutları şöyle deneyebilirsin:

- `4 * 5`
- `5 - 1`
- `40 / 2`

Üstel hesaplama yapmak için 2 üzeri 3 deyin, şöyle yazalım:

komut-satırı

```
>>> 2 ** 3
8
```

Gördüğün gibi Python çok iyi bir hesap makinesi. Eğer başka neler yapabileceğini merak ediyorsan devam edelim...

String'ler (dizeler)

Mesela ismin? İsmini tırnak işaretleri içerisinde şu şekilde yaz:

komut satırı

```
>>> "Zeynep"
'Zeynep'
```

İlk string'ini oluşturdun! String (dize), bilgisayar tarafından işlenebilen ve karakterlerden oluşan dizilerin genel adıdır. Bir string her zaman aynı özel karakterle başlamalı ve aynı özel karakterle bitmelidir. Tek tırnak (') veya çift tırnak (") olabilir (aralarında herhangi bir fark yok!). Tırnak işaretleri Python'da içlerinde olan şeyin bir string olduğunu ifade eder.
Stringler birbirlerine eklenebilir. Şunu dene:

komut satırı

```
>>> "Merhaba " + "Zeynep"
'Merhaba Zeynep'
```

Ayrıca stringleri bir sayı ile çarpabilirsin:

komut satırı

```
>>> "Zeynep" * 3
'ZeynepZeynepZeynep'
```

Eğer stringinin içerisine bir tırnak işareti koymak istiyorsan, bunun için iki seçeneğin var.

Çift tırnak kullanarak:

komut satırı

```
>>> "Turgut Uyar'ın dizeleriyiz"
"Turgut Uyar'ın dizeleriyiz"
```

veya backslash (\) kullanarak:
komut satırı

```
>>> 'Turgut Uyar\'ın dizeleriyiz'
"Turgut Uyar'ın dizeleriyiz"
```

Hoş değil mi? İsminin tamamını büyük harf yapmak için, sadece şunu yazman yeterli:

komut satırı

```
>>> "Zeynep".upper()
'ZEYNEP'
```
String'in üzerinde upper **fonksiyon**unu kullandın! Bir fonksiyon (upper() gibi), çağırıldığında (calling) Python'un girdi olarak verilen bir obje ("Zeynep") üzerinde gerçekleştirmesi gereken bir dizi işleme denilir. Eğer ismindeki harflerin sayısını öğrenmek istiyorsan bunun için de bir **fonksiyon** var!

komut satırı

```
>>> len("Zeynep")
6
```
Fonksiyonları neden bazen stringin sonunda bir . ile ("Zeynep".upper() gibi) ve bazen de önce fonksiyonu çağırıp sonra parantez içerisine stringi yazarak kullandığımızı merak ediyor musun? Pekala, bazı durumlarda, fonksiyonlar bir takım nesnelere aittirler, mesela upper(), yalnızca stringler üzerinde kullanılabilir. Böyle durumlarda, bu tarz fonksiyonlara biz **method** ismini veriyoruz. Diğer durumlarda, bir fonksiyon özel olarak bir nesneye ait olmayıp, farklı çeşitlerde nesneler üzerinde de kullanılabilir, aynı len() gibi. İşte bu nedenle "Zeynep" stringini len fonksiyonuna bir parametre olarak veriyoruz.

Özet

Tamam, stringlerden yeterince bahsettik. Şu ana kadar şu konuları öğrendin:

- **komut istemi** – komutları (kod) Python'un komut istemine yazdığınızda Python'da sonuçlandırarak yanıtlar üretir
- **sayılar ve dizeler** – Python'da sayılar matematik için dizeler ise metin nesneleri için kullanılmaktadır
- **operatörler** + ve * gibi, değerleri birleştirerek yeni bir değer üretmek için kullanılmaktadır
- **fonksiyonlar** upper() ve len() gibi, nesneler üzerinde eylemler gerçekleştirmektedirler.

Bunlar öğreneceğiniz her programlama dilinin temelleri. Biraz daha zor bir şey için hazır mısın?

Hatalar

Şimdi yeni bir şey deneyelim. Bir sayının uzunluğunu, bir string'in uzunluğunu bulduğumuz gibi bulabilir miyiz? Bunu görmek için len(304023) yazıp enter a basalım:
komut satırı

```
>>> len(304023)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'int' has no len()
```
İlk hatamızı aldık! ikonu çalıştırmak üzere olduğunuz programın beklediğiniz gibi çalışmayacağı konusunda sizi ikaz eder. Hatalar yapmak (kasıtlı olanlar bile) öğrenmenin önemli bir kısmı!

Nesne türü "int" (tam sayılar, tüm sayılar) in uzunluğu olmadığını söylüyor. Şimdi ne yapabiliriz? Belki de rakamı bir string olarak yazabiliriz? Stringlerin bir uzunluğu var, değil mi?

komut satırı

```
>>> len(str(304023))
6
```
İşe yaradı! str fonksiyonunu len fonksiyonunun içinde kullandık. str her şeyi string'e çeviriyor.

- str fonksiyonu, değişkenleri **stringe** çeviriyor
- int fonksiyonu değişkenleri **integera** çeviriyor

Önemli: Sayıları metin durumuna getirebiliriz ama metni sayılar durumuna tam olarak getiremeyebiliriz - yine de int('hello') olsa ne olurdu?

Değişkenler

Programlamada en önemli konulardan biri değişkenlerdir. Değişken (variable), daha sonra kullanmak istediğiniz bir yapıya verdiğiniz isimdir. Programcılar değişkenleri verileri tutmak ya da kodlarını daha okunabilir ve anlaşılabilir kılmak için kullanırlar ve böylece her şeyi sürekli akıllarında tutmaya gerek kalmaz.

`name` adında bir değişken yaratmak istediğimizi varsayalım:
komut satırı

```
>>> name = "Ayşe"
```

name (isim) eşittir "Ayşe" yazalım.

Farkettiğiniz gibi, program daha öncekilerinin aksine bu kez hiçbir cevap vermedi. O zaman böyle bir değişkenin gerçekten tanımlı olduğunu nasıl bilebiliriz? Basitçe, `name` yazıp `enter` tuşuna basalım:
komut satırı

```
>>> name
'Ayşe'
```

Yaşasın! İşte bu senin ilk değişkenin! :) Bu değişkenin işaret ettiği şeyi her zaman değiştirebilirsin:

komut satırı

```
>>> name = "Suzan"
>>> name
'Suzan'
```

Bu değişkeni fonksiyonlar içinde de kullanabilirsin:

komut satırı

```
>>> len(name)
5
```

Harika değil mi? Tabii ki değişkenler, sayılar da dahil herhangi bir şey olabilir. Şunu deneyin:

komut satırı

```
>>> a = 4
>>> b = 6
>>> a * b
24
```

Peki ya değişkenin adını yanlış kullanırsak? Ne olacağını tahmin ediyor musun? Deneyelim!

komut satırı

```
>>> city = "Tokyo"
>>> ctiy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'ctiy' is not defined
```

Bir hata! Gördüğünüz gibi, Python birçok hata çeşidine sahip ve bu hatanın adı **NameError**, yani İsimlendirme Hatası. Tanımlamadığınız bir değişkenin adını kullanmaya çalışırsanız, Python size bu hatayı verir. Eğer bu hata ile daha sonra karşılaşırsanız, kodunuzdaki değişkenlerin adını doğru yazıp yazmadığınızı kontrol edin.

Bununla biraz oynayıp, neler yapabildiğinizi görün!

Yazdırma (Print) İşlevi

Şunu deneyin:

komut satırı

```
>>> name = 'Merve'
>>> name
'Merve'
>>> print(name)
Merve
```

Sadece name yazdığınız zaman, Python yorumlayıcısından 'name' değişkeninin dize olarak *temsili* döner, yani tek tırnaklar içine alınmış M-e-r-v-e harfleri. Eğer print(name) derseniz Python ekrana değişkenin içeriğini yazdıracaktır, bu kez tırnaklar olmaksızın, daha temiz biçimde.
Daha ileride göreceğimiz gibi print(), işlevlerin içindeyken bir şey yazdırmak istediğimizde ya da bazı şeyleri birden fazla satırda yazdırmak istediğimizde de kullanışlıdır.

Listeler

Python, string (dize) ve integerın (tam sayı) yanı sıra, çok değişik türlerde nesnelere sahiptir.
Şimdi, **liste** türünü tanıtacağız. Listeler tam da düşündüğünüz gibidir: diğer nesnelerin listesi olan nesne. :)

Yeni bir liste yaratmakla devam edelim:

komut satırı

```
>>> []
[]
```

Evet, liste boş. Çok kullanışlı sayılmaz, değil mi? Hadi loto numaralarıyla liste oluşturalım. Sürekli kendimizi tekrar etmek istemeyiz, o yüzden listeyi değişkene atayalım:

komut satırı

```
>>> lottery = [3, 42, 12, 19, 30, 59]
```

Pekala, listeyi oluşturduk! Onunla ne yapabiliriz? Hadi listede kaç tane loto numarası olduğunu görelim. Hangi fonksiyonu kullanman gerektiği hakkında bir fikrin var mı? Zaten bildiğin bir fonksiyon!

komut satırı

```
>>> len(lottery)
6
```

Evet! len() listedeki nesne sayısını verir. Kullanışlı, değil mi? Belki de şu an listeyi sıralarız:
komut satırı

```
>>> lottery.sort()
```

Bu hiçbir cevap vermez, sadece listedeki numaraların sırasını değiştirir. Şimdi listeyi yazdıralım ve ne olduğunu görelim:

komut satırı

```
>>> print(lottery)
[3, 12, 19, 30, 42, 59]
```

Gördüğünüz gibi, listedeki sayılar artık küçükten büyüğe sıralı. Tebrikler!

Belki de sıralamayı ters çevirmek isteriz? Hadi yapalım!

komut-satırı

```
>>> lottery.reverse()
>>> print(lottery)
[59, 42, 30, 19, 12, 3]
```

Kolay, değil mi? Listeye yeni bir eleman eklemek isterseniz, şu komutu yazarak yapabilirsiniz:

komut-satırı

```
>>> lottery.append(199)
>>> print(lottery)
[59, 42, 30, 19, 12, 3, 199]
```

Sadece listedeki ilk elemanı göstermek isterseniz, **indexes** (indeksler) ile yapabilirsiniz. İndeks elemanın listede nerede olduğunu belirten numaradır. Programcılar sıfırdan başlamayı tercih ederler, bu yüzden listedeki ilk eleman listenin 0. indeksindedir, sonraki 1. indeksindedir ve böyle devam eder. Şunu deneyin:

komut satırı

```
>>> print(lottery[0])
59
>>> print(lottery[1])
42
```

Gördüğünüz gibi, listedeki nesnelere listenin ismi ve köşeli parantez içindeki nesnenin indeksini kullanarak ulaşabilirsin.

Listenizden bir şeyler silmek için, yukarıda öğrendiğimiz gibi **indeksleri** ve `pop()` metodunu kullanmamız gerekecektir. Bir örnekle öğrendiklerimizi pekiştirelim; listeden ilk numarayı sileceğiz.
komut satırı

```
>>> print(lottery)
[59, 42, 30, 19, 12, 3, 199]
>>> print(lottery[0])
59
>>> lottery.pop(0)
59
>>> print(lottery)
[42, 30, 19, 12, 3, 199]
```

Kusursuz çalıştı!

Daha fazla eğlence için diğer indeksleri de deneyin: 6, 7, 1000, -1, -6 veya -1000. Denemeden önce komutların sonuçlarını tahmin etmeye çalışın. Sonuçlar mantıklı mıydı?

Bütün liste fonksiyonlarını Python dökümantasyonunun bu bölümünde bulabilirsin: https://docs.python.org/3/tutorial/datastructures.html

Sözlükler (Dictionaries)

Sözlük listeye benzerdir ancak sözlük değerlerine indeks yerine anahtar ile ulaşılır. Anahtar metin veya numara olabilir. Boş bir sözlük oluşturmak için kullanılan söz dizimi şudur:

komut satırı

```
>>> {}
{}
```

Bu boş bir sözlük oluşturduğunuzu gösterir. Yaşasın!

Şimdi, bu komutu yazmayı deneyin (kendi bilgilerinizle değiştirmeyi deneyin):

komut satırı

```
>>> participant = {'name': 'Ayşe', 'country': 'Türkiye', 'favorite_numbers': [7, 42, 92]}
```

Bu komutla, üç anahtar-değer çifti ile `participant` (katılımcı) isminde bir değişken oluşturdunuz:

- Anahtar `name` 'Ayşe' (`string` nesnesi) değerine işaret eder,
- `country` Türkiye (bir diğer `string`) değerine),

- ve favorite_numbers [7, 42, 92] (3 numaralı bir list) değerine işaret eder.

  Bu söz dizimi ile tek bir anahtarın içeriğini kontrol edebilirsin:

  komut satırı

  ```
  >>> print(participant['name'])
  Ayşe
  ```
  Gördünüz mü, bu listeye benzer. Fakat indeksi hatırlamanıza gerek yok - sadece ismini hatırlayın.

  Python'a olmayan bir anahtarın değerini sorarsak ne olur? Tahmin edebiliyor musun? Hadi deneyip görelim!

  komut satırı

  ```
  >>> participant['age']
  Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
  KeyError: 'age'
  ```
  Bir başka hata! **KeyError** hatası. Python yardımseverdir ve sana 'age' anahtarının sözlükte bulunmadığını söyler.
  Ne zaman sözlük veya liste kullanmalısın? Düşünmek için güzel bir nokta. Sonraki satıra bakmadan önce cevap üzerinde bir düşünün.

- Sıralı elemanlara mı ihtiyacın var? Liste ile devam et.
- İleride hızlıca (anahtarlar ile) değerlere ulaşmak istediğin için anahtarlar ile ilişkilendirilmiş değerlere mi ihtiyacın var? Sözlük kullan.

  Sözlükler de listeler gibi değişebilirdir (*mutable*), yani oluşturulduktan sonra değiştirilebilirler.
  Oluşturulduktan sonra sözlüklere anahtar/değer çifti ekleyebilirsiniz, aşağıdaki gibi:

  komut-satırı

  ```
  >>> participant['favorite_language'] = 'Python'
  ```
  Listeler gibi, len() metodu sözlükteki anahtar-değer çiftlerinin sayısını bize verir. Devam edip şu komutu yazın:
  komut satırı

  ```
  >>> len(participant)
  4
  ```
  Umarım şu ana kadar mantıklı gelmiştir :) Sözlüklerle biraz daha eğlenceye hazır mısın? İlginç şeyler için sonraki satıra atla.

  Sözlükten bir maddeyi silmek için pop() metodunu kullanabilirsin. Mesela, 'favorite_numbers' anahtarına karşılık gelen elemanı silmek istersen, şu komutu yaz:
  komut satırı

  ```
  >>> del participant['favorite_numbers']
  >>> participant
  {'country': 'Türkiye', 'favorite_language': 'Python', 'name': 'Ayşe'}
  ```
  Çıktıdan görebildiğin gibi, 'favorite_numbers' anahtarına karşılık gelen anahtar-değer çifti silindi.

  Bunun yanı sıra, sözlükteki daha önce oluşturulmuş anahtarın değerini değiştirebilirsiniz. Şunu yazın:

  komut satırı

  ```
  >>> participant['country'] = 'Almanya'
  >>> participant
  {'country': 'Almanya', 'favorite_language': 'Python', 'name': 'Ayşe'}
  ```

Gördüğün gibi, `'country'` anahtarının değeri `'Türkiye'`den `'Almanya'`ya çevrildi. :) Heyecan verici değil mi? Yaşasın! Bir başka harika şey öğrendin.

Özet

Harika! Şu an programlama hakkında birçok şey biliyorsun. Bu kısımda, şunları öğrendin:

- **errors** - eğer Python yazdığın komutu anlamazsa çıkan hataları nasıl okuyacağını ve anlayacağını artık biliyorsun
- **değişkenler** - daha kolay kod yazmanı sağlayan ve kodunu daha okunabilir yapan nesnelerin isimleri
- **listeler** - belirli bir sırada tutulan nesnelerin listesi
- **sözlükler** - anahtar-değer çifti olarak tutulan nesneler

Bir sonraki bölüm için heyecanlı mısınız? :)

Karşılaştırma

Programlamanın büyük kısmı karşılaştırma içerir. Karşılaştırması en kolay olan şey nedir? Tabi ki sayılar. Bakalım nasıl çalışıyor:

komut satırı

```
>>> 5 > 2
True
>>> 3 < 1
False
>>> 5 > 2 * 2
True
>>> 1 == 1
True
>>> 5 != 2
True
```

Python'a birkaç sayı karşılaştırmasını söyledik. Gördüğünüz gibi, sadece sayıları karşılaştırmakla kalmadı, aynı zamanda metodların sonuçlarını da karşılaştırdı. Güzel değil mi?

İki sayının eşit olup olmadığını öğrenmek için neden iki tane eşittir işaretini `==` yan yana koyduk? Değişkenlere içerik verirken, tek `=` işaretini kullanıyoruz. Her zaman ama **her zaman** ikisini birden koyman gerekir – `==` – eğer birbirlerine eşit olup olmadıklarını kontrol etmek isterseniz. Sayıların birbirine eşit olmaması durumunu da kontrol edebiliriz. Bunun için, yukarıdaki örnekteki gibi `!=` sembolünü kullanıyoruz. Python' a iki görev daha verin:

komut satırı

```
>>> 6 >= 12 / 2
True
>>> 3 <= 2
False
```

`>`'ı ve `<`'ı gördük, ama `>=` ve `<=` ne anlama geliyor? Onları böyle okuyabilirsin:

- `x > y` : x büyüktür y
- `x < y` : x küçüktür y
- `x <= y` : x küçük eşittir y
- `x >= y` : x büyük eşittir y

Harika! Birkaç denemeye daha ne dersiniz? Şunu deneyin:

komut satırı

```
>>> 6 > 2 and 2 < 3
True
>>> 3 > 2 and 2 < 1
False
>>> 3 > 2 or 2 < 1
True
```
Python'a istediğiniz kadar sayıyı karşılaştırmak için verebilirsiniz, ve size hepsinin cevabını verecek. Çok akıllı değil mi?

- **and** - Mantıkta kullandığımız "ve" anlamına geliyor, yani iki taraf da True, yani doğruysa, cevap da True olacak
- **or** - Bu da "veya" anlamına geliyor, karşılaştırılan iki taraftan tek bir tanesi bile True ise bize True cevabını verecek

Portakallarla elmaları karşılaştılaştırabilir miyiz? Bunun Python'daki eşdeğerini deneyelim:

komut satırı

```
>>> 1 > 'django'
Geri görüş (en son çağrı):
  Dosya "<stdin>", satır 1, <module>'in içinde
HataTürü: '>' 'int' ve 'str' örnekleri arasında desteklenmiyor
```
Gördüğünüz gibi Python tam sayılar(int) ve kelimeleri(yani stringleri, str) karşılaştıramıyor. Onun yerine, **TypeError** göstererek iki farklı tipteki değişkenin karşılaştırılamayacağını söylüyor.

Boolean (Mantıksal)

Bu arada, python'da yeni bir nesne türü öğrendin. Buna **Boolean** denir.

Yalnızca iki Boolean nesnesi vardır:

- True
- False

Python'un bunu anlaması için her zaman "True" (ilk harf büyük, geri kalanları küçük) yazmanız gerekiyor. **true, TRUE, tRUE işe yaramaz -- sadece True doğru.** (Aynısı "False" için de geçerli.)

Boolean'lar değişken de olabiliyor! Bakınız:

komut-satırı

```
>>> a = True
>>> a
True
```
Ayrıca bu şekilde de yapabilirsiniz:

komut-satırı

```
>>> a = 2 > 5
>>> a
False
```
Boolean'lar ile aşağıdaki komutları deneyerek biraz oynayın:

- `True and True`
- `False and True`
- `True or 1 == 1`
- `1 != 2`

Tebrikler! Boolean'lar programlamadaki en havalı özelliklerden, ve az önce onları nasıl kullanmanız gerektiğini öğrendiniz!

Kaydet!

Şimdiye kadar kodumuzu bizi sadece tek satır yazmaya limitleyen yorumlayıcı üzerinde yazdık. Normal programlar dosyalar içine kaydedilir ve programlama dilimizin **yorumlayıcısıyla** veya **derleyicisiyle** çalıştırılır. Şimdiye kadar programlarımızı Python **yorumlayıcısında** teker satır teker satır çalıştırdık. Bundan sonraki görevlerde, birden fazla satıra ihtiyacımız olacak, bu yüzden şunlara ihtiyacımız olacak:

- Python yorumlayıcısından çıkın
- Seçtiğiniz kod düzenleyicisini açın
- Yeni Python dosyasına kod kaydedin
- Çalıştırın!

Kullandığımız Python yorumlayıcısından çıkmak için `exit()` fonksiyonunu yazın
komut satırı

```
>>> exit()
$
```

Bu sizi komut satırına geri yönlendirecektir.

Biraz önce kod editörü bölümünden bir kod editörü seçmiştik. Şimdi o editörü açmalı ve yeni bir dosya içine kod yazmalıyız:

editör

```
print('Merhaba, Django girls!')
```

Açıkça, artık oldukça deneyimli Python programcısısın, bu yüzden bugün öğrendiğin kodları yazmaktan çekinme.

Şimdi dosyayı tanımlayıcı bir isimle kaydetmemiz gerekir. Dosyanın ismine **python_intro.py** diyelim ve masaüstüne kaydedelim. Dosyaya istediğimiz ismi verebiliriz, burada önemli olan kısım dosyanın **.py** uzantısı ile bitmesidir. **.py** uzantısı işletim sistemimize bu dosyanın bir **python çalıştırılabilir dosyası** olduğunu ve Python'un bu dosyayı çalıştırabileceğini belirtiyor.

Not Kod editörleriyle ilgili en harika şeylerden birine dikkat etmelisiniz: renkler! Python konsolunda herşey aynı renkteydi; şimdi bakın `print` fonksiyonu dizeden farklı renkte. Bunun ismi "söz dizimi vurgulama" ve kod yazarken gerçekten yararlı bir özellik. Koddaki renkler ipucu verir, kapanmamış dizeler gibi ya da aşağıda göreceğimiz (`def` fonksiyonu gibi imla hatası içeren anahtar kelimeler olabilir). Bu kod düzenleyicisi kullanma nedenlerimizden biri. :)

Dosyayı kaydettiğimize göre artık çalıştırabiliriz! Konsoldan **klasör değiştirme** yaparak masaüstüne ulaşın, komut satırı bölümünde öğrendiklerinizi hatırlayın.

Change directory: OS X

Change directory: Linux

Change directory: Windows Command Prompt

Change directory: Windows Powershell

Takılırsanız, yardım isteyin. Eğitmenler bunun için var!

Şimdi dosyadaki komutları çalıştırmak için Python'u kulllanın:

komut-satırı

```
$ python3 python_intro.py
Merhaba, Django girls!
```

Not: Windows'ta 'python3' bir komut olarak geçmez. Onun yerine, dosyayı çalıştırmak için 'python'ı kullanın:

komut satırı

```
> python python_intro.py
```

Tamam! Bir dosyaya kaydedilen ilk Python programınızı çalıştırdınız. Harika hissediyor musunuz?

Şimdi programlamanın olmazsa olmaz bir aracını öğrenme zamanı:

If … elif … else

Kodunuzdaki bir çok şeyi sadece belirli bir durum sağlanıyorsa çalıştırmayı isteyeceksiniz. İşte tam da bu yüzden Python'da **if deyimi** isminde bir yapı bulunuyor.

python_intro.py dosyasındaki kodunuzu şununla değiştirin:

python_intro.py

```
if 3 > 2:
```

Eğer bunu kaydetmiş ve çalıştırmış olsaydık, bunun gibi bir hata görecektik:

komut satırı

```
$ python3 python_intro.py
File "python_intro.py", line 2
      ^
SyntaxError: unexpected EOF while parsing
```

Python bizden kendisine 3 > 2 durumu (veya True) sağlandığında neyi çalıştıracağını söylememizi bekliyor. Python'a "Çalışıyor!" yazmasını söyleyelim. **python_intro.py** dosyanızdaki kodu şununla değiştirin:
python_intro.py

```
if 3 > 2:
    print('Çalışıyor!')
```

4 tane boşluk karakteri bıraktığımıza dikkat ettiniz mi? Bunu yaparak if ifadesine yazılan durum doğru olduğunda neyi çalıştırması gerektiğini Python'a söylemiş oluyoruz. Aslında tek bir boşlukla da yapabilirsiniz, ama hemen hemen bütün Python programcıları kodlarının temiz görünmesi için 4 boşluk bırakıyor. Metin düzenleyiciniz ayarlıysa bir tab karakteri de 4 boşluk karakteri olarak sayılacaktır. Seçiminizi yaptıktan sonra değiştirmeyin! Eğer girintilerde 4 boşluk kullandıysanız, gelecek girintilerde de 4 boşluk kullanmaya devam edin - aksi halde sorunlarla karşılaşabilirsiniz.

Kaydedip çalıştırmayı deneyelim:

komut satırı

```
$ python3 python_intro.py
Çalışıyor!
```

Not: Windows'ta 'python3'ün komut olarak geçerli olmadığını unutmayın. Bundan böyle dosyayı çalıştırmak için 'python3'ü 'python'la değiştirin.

Ya bir koşul True (Doğru) değilse?

Önceki örneklerde kod sadece koşullar sadece True (doğru) olduğunda çalışıyordu. Ama Python ayrıca elif ve else ifadelerine de sahip:
python_intro.py

```
if 5 > 2:
   print("5 gerçekten de 2'den büyüktür")
else:
   print("5  2'den büyük değildir")
```
Bu kod çalıştığında aşağıdaki çıktıyı verecektir:

komut satırı

```
$ python3 python_intro.py
5 gerçekten de 2'den büyüktür
```
Eğer 2 5'ten büyük bir sayı olsaydı, ikinci komut çalıştırılmış olacaktı. Bakalım `elif` nasıl çalışıyor:
python_intro.py

```
name = 'Zeynep'
if name == 'Ayşe':
   print('Selam Ayşe!')
elif name == 'Zeynep':
   print('Selam Zeynep!')
else:
   print('Selam yabancı!')
```
ve çalıştırılınca:

komut-satırı

```
$ python3 python_intro.py
Selam Zeynep!
```
Gördünüz mü? Eğer önceki if ifadeleriniz doğru olmazsa kontrol edilmek üzere `elif` ifadeleri ekleyebilirsiniz. `if` ifadesinden sonra istediğiniz kadar `elif` ifadesi ekleyebilirsiniz. Mesela:
python_intro.py

```
volume = 57
if volume < 20:
   print("Çok sessiz.")
elif 20 <= volume < 40:
   print("Güzel bir fon müziği")
elif 40 <= volume < 60:
   print("Harika, her notayı duyabiliyorum")
elif 60 <= volume < 80:
   print("Parti başlasın")
elif 80 <= volume < 100:
   print("Biraz gürültülü!")
else:
   print("Kulaklarım ağrıyor! :(")
```
Python sırayla her sorguyu çalıştırır ve sonucu ona göre yazar:

komut satırı

```
$ python3 python_intro.py
Harika, her notayı duyabiliyorum
```

Yorumlar

Yorumlar `#` ile başlayan satırlardır. İstediğiniz her neyse `#` den sonra yazabilirsiniz ve Python onu gözardı eder. Yorumlar kodunuzu diğer insanların anlamasını daha kolaylaştırabilir.
Bakalım nasıl gözüküyor:

python_intro.py

```
# Çok yüksek ya da çok düşük olduğunda ses seviyesini değiştirme
if volume < 20 or volume > 80:
    volume = 50
    print("That's better!")
```

Kodun her satırı için bir açıklama yazmaya ihtiyacınız yoktur, ama kodunuzun niçin birşey yaptığını açıklamak ya da kompleks bir şey yaptığında bir özet sunmak için faydalıdırlar.

Özet

En son yaptığınız alıştırmalarda öğrendikleriniz:

- **kıyaslama yapmak** - Python'da `>`, `>=`, `==`, `<=`, `<` ve `ve`, `veya` operatörlerini kullanarak kıyaslama yapabilirsiniz
- **Boolean** - İki farklı değer alabilen bir nesne tipidir: Ya `True` (doğru) olur ya da `False` (yanlış)
- **Dosya kaydetmek** – kodlarımızı dosyalara kaydederek daha büyük programları çalıştırabiliriz.
- **if ... elif ... else** - ifadelerini sadece belirli durumlar sağlandığında çalıştırmak istediğimiz komutlar için kullanabiliriz.
- **yorumlar** - kodunuzu belgelemenize izin verecek şekilde Python'un çalışmayacağı satırlar

Bu bölümün son kısmının zamanı geldi!

Kendi fonksiyonlarınız!

Python'da çalıştırabileceğin `len()` gibi fonksiyonları hatırlıyor musun? Güzel, iyi haber - Şimdi kendi fonksiyonlarını nasıl yazacağını öğreneceksin!
Fonksiyon Python tarafından işlenmesi gereken yönergeler dizisidir. Python'da her fonksiyon `def` anahtar kelimesi ile başlar, bir isim verilir ve bazı parameterleri olabilir. Hadi başlayalım. **python_intro.py** içindeki kodu aşağıdaki ile değiştirelim:
python_intro.py

```
python
def hi():
    print('Merhaba!')
    print('Nasılsın?')

hi()
```

Tamam, ilk fonksiyonumuz hazır!

Fonksiyon adını neden dosyanın en altına yazdığımızı merak edebilirsiniz. Bunun nedeni, Python'ın dosyayı okuyup, onu yukarıdan aşağı doğru işlemesi. Yani fonksiyonumuzu kullanabilmek için, onu en alt kısımda yeniden yazmalıyız.

Haydi şimdi bunu çalıştıralım ve neler olacağını görelim:

komut satırı

```
$ python3 python_intro.py
Merhaba!
Nasılsın?
```

Not: Eğer çalışmadıysa panik yapmayın! Çıktı neden olduğu hakkında bir fikir verir:

- Eğer bir `NameError` alırsanız, muhtemelen yanlış bir şey yazdığınız anlamına gelir, bu nedenle `def hi()`: lı fonksiyonu oluştururken ve `hi()` lıyı çağırırken aynı adı kullanıp kullanmadığınızı kontrol etmelisiniz.
- Eğer bir `IndentationError` alırsanız,`print` dizelerinin her ikisinin de satır başında aynı boşluğa sahip olduğunu kontrol et: python fonksiyonun içindeki tüm kodların düzenli bir şekilde hizalanmasını ister.
- Eğer tamamında da çıktı yoksa, son `hi()` *isn't* girintiliğini kontrol et - eğer öyleyse, bu dize fonksiyonunda bir parçası haline gelecek ve hiçbir zaman çalışmayacak.

İlk fonksiyonumuzu parametrelerle birlikte oluşturalım. Önceki örneği - çalışmaktaki kişiye merhaba diyen bi fonksiyon - ismiyle kullanacağız:

python_intro.py

```
def hi(name):
```

Gördüğünüz gibi, fonksiyonumuza name (isim) adında bir parametre ekledik:
python_intro.py

```
def hi(name):
    if name == 'Ayşe':
        print("Merhaba Ayşe!')
    elif name == 'Zeynep':
        print('Merhaba Zeynep!')
    else:
        print('Merhaba yabancı!')

hi()
```

Unutmayın: if içerisindeki print fonksiyonundan önce dört tane boşluk var. Bunun sebebi sadece durum sağlandığında çalışmasını istememiz. Bakalım nasıl çalışıyor:
komut satırı

```
$ python3 python_intro.py
Traceback (most recent call last):
File "python_intro.py", line 10, in <module>
  hi()
TypeError: hi() missing 1 required positional argument: 'name'
```

Üzgünüz, bir hata. Neyse ki, Python bize oldukça yararlı bir hata mesajı veriyor. hi() fonksiyonun (yukarıda tanımladığımız) bir değişken kullanımını gerektirdiğini (name isimli) ve bizim o değişkeni fonksiyonu çağırırken iletmeyi unuttuğumuzu söylüyor. Dosyanın alt kısmında hatayı düzeltelim:
python_intro.py

```
hi("Ayşe")
```

Ve tekrar çalıştıralım:

komut satırı

```
$ python3 python_intro.py
Selam Ayşe!
```

Eğer ismi değiştirirsek ne olur?

python_intro.py

```
hi("Zeynep")
```

Ve çalıştırın:

komut-satırı

```
$ python3 python_intro.py
Selam Zeynep!
```

Peki Ayşe veya Zeynep dışında başka bir isim yazdığımızda ne olacağını tahmin edebiliyor musunuz? Deneyin ve tahmininizin doğru olup olmadığını görün. Şunun gibi bir şey yazmalı:

komut satırı

```
Selam yabancı!
```

Süper değil mi? Böylece fonksiyona göndereceğiniz isim değiştiğinde aynı kodu tekrar tekrar yazmanıza gerek kalmayacak. İşte fonksiyonlara tam da bu yüzden ihtiyacımız var - aynı kodu tekrar yazmaya gerek yok!

Hadi daha zekice bir şey yapalım - İkiden fazla isim var ve her biri için bir şart yazmak zor olur değil mi?

python_intro.py

```
def hi(name):
    print('Merhaba ' + name + '!')

hi("Seda")
```

Şimdi kodu çağıralım:

komut satırı

```
$ python3 python_intro.py
Merhaba Seda!
```

Tebrikler! Az önce fonksiyonları nasıl yazacağınızı öğrendiniz! :)

Döngüler

Bu da zaten son parça. Hızlı oldu, değil mi? :)

Programcılar kendilerini tekrar etmeyi sevmezler. Programlama tamamen işleri otomatize etmedir, bu yüzden her insanı ismiyle selamlamak istemeyiz, değil mi? İşte burası döngülerin devreye girdiği yerdir.

Hala listeleri hatırlıyoruz değil mi? Haydi bir kızlar listesi yapalım:

python_intro.py

```
kızlar = ['Seda', 'Gül', 'Pınar', 'Ayşe', 'Sen']
```

Diyelim ki hepsine merhaba demek istiyoruz. Az önce yazdığımız hi fonksiyonunu döngü içinde kullanabiliriz:
python_intro.py

```
for name in girls:
```

for un davranışı if e benziyor; aşağıdaki kodda her iki satır girintili olmalı (dört boşluk ile).
Dosyada yer alacak tam kod aşağıdadır:

python_intro.py

```
def hi(name):
    print('Merhaba ' + name + '!')

girls = ['Seda', 'Gül', 'Pınar', 'Ayşe', 'Sen']
for name in girls:
    hi(name)
    print('Sıradaki kız')
```

Ve onu çalıştırdığımız zaman:

komut satırı

```
$ python3 python_intro.py
Selam Seda!
Sıradaki kız
Selam Gül!
Sıradaki kız
Selam Pınar!
Sıradaki kız
Selam Ayşe!
Sıradaki kız
Selam Sen!
```

```
Sıradaki kız
```
Gördüğünüz gibi, `for` cümlesinin içine boşluk karakteri ile koyduğunuz her şey `girls` listesi için tekrarlanıyor. Ayrıca `for`'u `range` fonksiyonuyla beraber sayılar üzerinde de kullanabilirsiniz:
python_intro.py

```
for i in range(1, 6):
    print(i)
```
Çalıştırırsak:

komut satırı

```
1
2
3
4
5
```
`range` fonksiyonu birbirini takip eden sayılardan bir liste oluşturur (bu sayıları da siz parametre olarak yazarsınız).
Sizin verdiğiniz ikinci parametrenin listede olmadığına dikkat edin. Yani `range(1, 6)` 1'den 5'e kadar sayıyor, 6 dahil edilmiyor. Çünkü "aralık" yarı açıktır, ve yani ilk değeri içerir, ama son değeri içermez.

Özet

İşte bu. **Harikasın, süpersin!** Bu bölüm biraz zordu, kendinle gurur duymalısın. Biz buraya kadar geldiğin için seninle gurur duyuyoruz!

Resmi ve tam python tutorialı için https://docs.python.org/3/tutorial/ adresini ziyaret edin. Bu size dil hakkında daha kapsamlı ve eksiksiz bir çalışma sunacaktır. Teşekkürler :)

Kaynaklar:

https://tutorial.djangogirls.org/

https://www.pythondersleri.com/

http://www.python.org