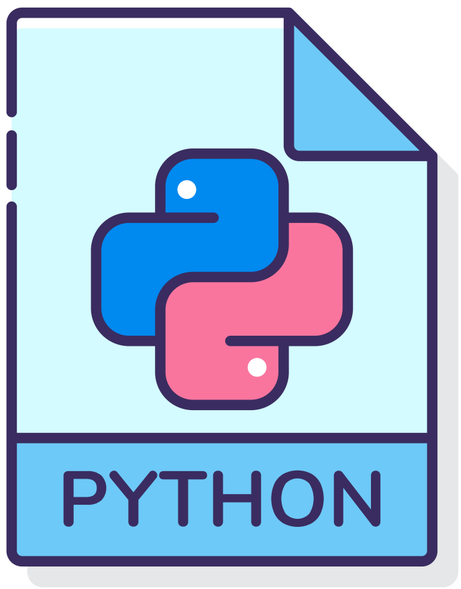
NESNE TABANLI PROGRAMLAMA - BIL207

Öğr. Gör. Buse Yaren TEKİN

KASTAMONU ÜNİVERSİTESİ
2006



Sunum Ana Hatları

BUGÜNÜN BAŞLIKLARI

- Sanal Ortam Kavramı
- Değişken Kavramı
- Python Programlama Diline Giriş ve Sözdizimi
- Sayılar ve Aritmetik İşlemler

DÜŞÜNÜLECEK ŞEY

Hızlı ve gelişmiş bilgisayarlar tembel programcılar üretir.

ROBERT HUMMEL

Nesne Tabanlı Programlama | Nesneye Yönelik Programlama

NYP'nin altında yatan birimselliğin ana fikri, her bilgisayar programının (izlence), ***etkileşim içerisinde olan birimler veya nesneler kümesinden oluştuğu varsayımıdır***. Bu nesnelerin her biri, kendi içerisinde veri işleyebilir, ve diğer nesneler ile çift yönlü veri alışverişinde bulunabilir. **Hâlbuki NYP'den önce var olan tek yaklaşımda (Yordamsal programlama), programlar sadece bir komut dizisi veya birer işlev (fonksiyon) kümesi olarak görülmektedirler.**

Nesne Tabanlı Programlama | Nesneye Yönelik Programlama

Günümüzde çok çeşitli nesne tabanlı programlama dilleri olmasıyla beraber, en popüler diller sınıflar üzerine kurulmuşlardır **(class-based).** Bu dillerde nesneler sınıfların birer üyesidir ve nesnelerin tipini de bu sınıflar belirlerler.
En yaygın NYP dillerinden bazıları, Python, C++, Objective-C, Smalltalk, Delphi, Java, Swift, C#, Perl, Ruby ve PHP' dir.

Nesne yönelimli programlama dilleri yukarıda adı geçen tüm ögelere sahip olurken, Ada, JavaScript, Visual Basic gibi nesne tabanlı programlama dilleribirkaçından yoksundur, bu dillerin başlıca yoksunluğu kalıtıma sahip olmamalarıdır.

Neden Nesne Tabanlı Programlama?

Gerçek hayatta gördüğümüz birçok nesnenin bilgisayar ortamına aktarılma şeklidir. Yani bir nesnenin rengi, durumu, ismi, üretim yılı gibi birçok özelliklerin bilgisayar ortamında gösterilmesi buna örnek olarak verilebilir.

Yazılımların karışıklığı ve boyutlarının artması, belirli bir nitelik düzeyini korumak için gereken maliyeti, zamanı ve çabayı arttırıyordu. OOP bu soruna karşı çözüm olarak getiren özelliği yazılımdaki birimselliği yüksek oranda benimsemesidir.

Nesne Yönelimli Programlamanın Faydaları

- Nesne oluşturma bir sınıf içerisinde toplanır ve tüm projelerde kullanılabilirliğe olanak sağlar.
- Sınıfların 1 kez oluşturulması sayesinde uzun kodları tekrardan yazmak yerine kısa kodlamalar ile çalıştırılabilir.
- Uzun kodların tekrar yazılmasının engellenmesi sayesinde geliştirme süreci kısalır.
- Nesneler birbirinden bağımsız olduğundan bilgi gizliliği konusunda avantaj sağlar.
- Sınıflar sayesinde tüm projelerde değişiklik yapmak yerine tek bir sınıfta değişiklik yapılıp tüm projelerde çalışması sağlanır. Bu zaman kaybını büyük ölçüde azaltır.

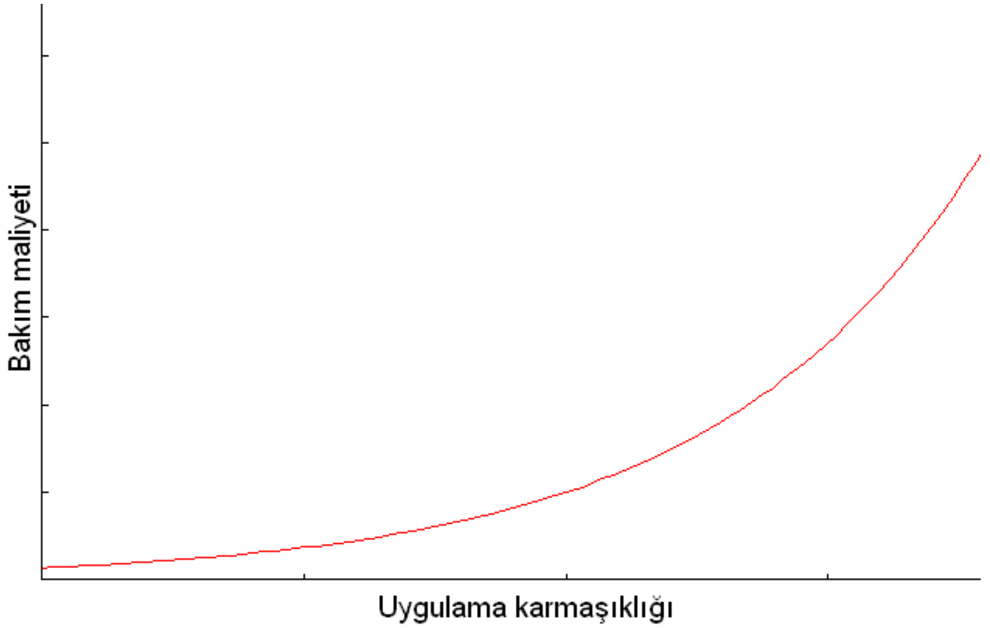
Nesne Tabanlı Programlama | Nesneye Yönelik Programlama

Nesne yönelimli programlama (NYP), (İngilizce: Object - Oriented Programming)(OOP), Her işlevin nesneler olarak soyutlandığı bir programlamayaklaşımıdır. NYP destekleyen programlama dilleri yüksek seviye diller olarak adlandırılır.

1960'lı yılların sonuna doğru ortaya çıkan bu yaklaşım, o dönemin yazılım dünyasında beliren bir bunalımın sonucudur. **Yazılımların karmaşıklığı ve boyutları sürekli artıyor, ancak belli bir nitelik düzeyi korumak için gereken bakımın maliyeti zaman ve çaba olarak daha da hızlı artıyordu. NYP'yi bu soruna karşı bir çözüm haline getiren başlıca özelliği, yazılımda birimselliği (modularity) benimsemesidir.**

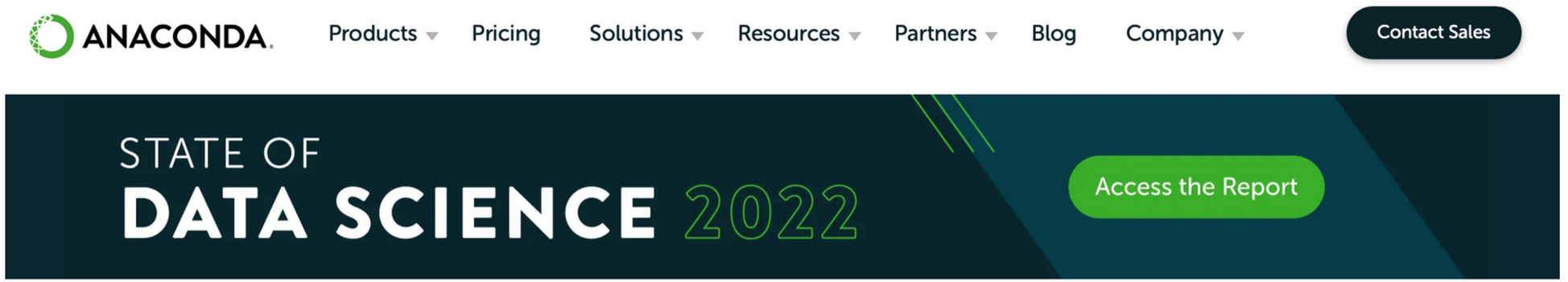
Nesne Tabanlı Programlama | Nesneye Yönelik Programlama

NYP ayrıca, bilgi gizleme (information hiding), veri soyutlama (data abstraction), çok biçimlilik (polymorphism) ve kalıtım (inheritance) gibi yazılımın bakımını ve aynı yazılım üzerinde birden fazla kişinin çalışmasını kolaylaştıran kavramları da yazılım literatürüne kazandırmıştır. Sağladığı bu avantajlardan dolayı, NYP günümüzde geniş çaplı yazılım projelerinde yaygın olarak kullanılmaktadır.



NYP'dan önceki uygulamaların bakım maliyeti ve karmaşıklık ilişkisi

OOP Çalışma Ortamı - Anaconda IDE



Anaconda IDE

Anaconda ücretsiz ve açık kaynaklı, Python ve R programlama dillerinin bilimsel hesaplama kullanımında paket yönetimini kolaylaştırmayı amaçlayan bir özgür ve açık kaynaklı dağıtımdır. Paket sürümleri conda paket yönetim sistemi ile yönetilir. Anaconda dağıtımı Windows, Linux ve MacOS işletim sistemlerinde kullanılabilen veri bilimi paketleri içerir.

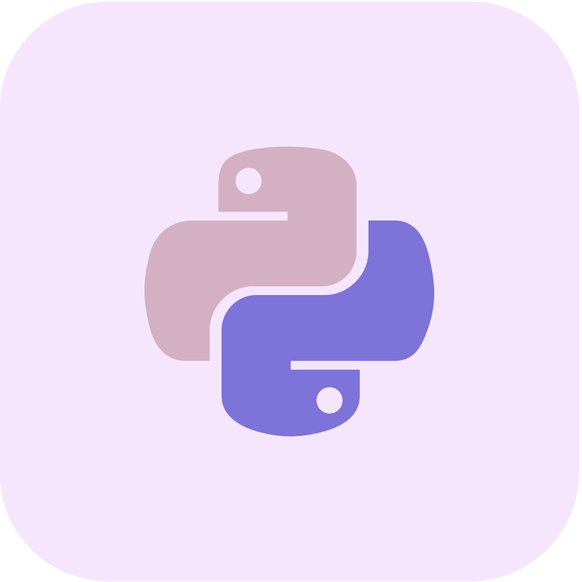
Anaconda IDE Genel Bakış

Anaconda dağıtımı, 1.500'den fazla paketin yanı sıra conda paketi ve sanal çevre yöneticisiyle birlikte gelir Ayrıca komut satırı arabirimine (CLI) grafiksel bir alternatif olarak bir GUI, Anaconda Navigator içerir.

Conda ve pip paket yöneticisi arasındaki en büyük fark, paket gereksinimlerinin yönetim farkıdır, **bu da Python veri bilimi ve condanın varlığının nedeni için önemli bir zorluktur.**

Python için Paket Yöneticileri

Paket yönetim sistemi ya da paket yöneticisi; yazılım paketlerinin ve kütüphanelerinin kurulum, güncelleme, konfigürasyon, kaldırılması işlemlerinin tutarlı ve stabil bir şekilde yürütülmesini sağlayan olan sistemlerdir. Tipik olarak paket ve kütüphanelerin hangi versiyonunun kurulduğunu ve birbirlerine olan bağımlılıklarını da hesaba katarlar. Modern paket yöneticilerinin birçoğu merkezi bir kaynaktan yazılım ve kütüphanelerin indirilip yüklenmesi işlevine sahiptirler.



Python için Paket Yöneticileri

Pip bir paket yüklediğinde, önceden yüklenmiş paketlerle çakışıp çakışmadığını kontrol etmeden, bağımlı Python paketlerini otomatik olarak kurar. Mevcut kurulumun durumuna bakılmaksızın bir paket ve bağımlılıklarından herhangi birini kuracaktır. Bu nedenle, örneğin Google Tensorflow'un çalışan bir kurulumuna sahip bir kullanıcı, gerekli olan numpy kütüphanesinin Tensorflow tarafından kullanılandan farklı bir sürümünü gerektiren farklı bir paket yüklemek için pip kullanmayı bıraktığını görebilir. Bazı durumlarda, paket çalışıyor gibi görünebilir ancak ayrıntılı olarak farklı sonuçlar verebilir.

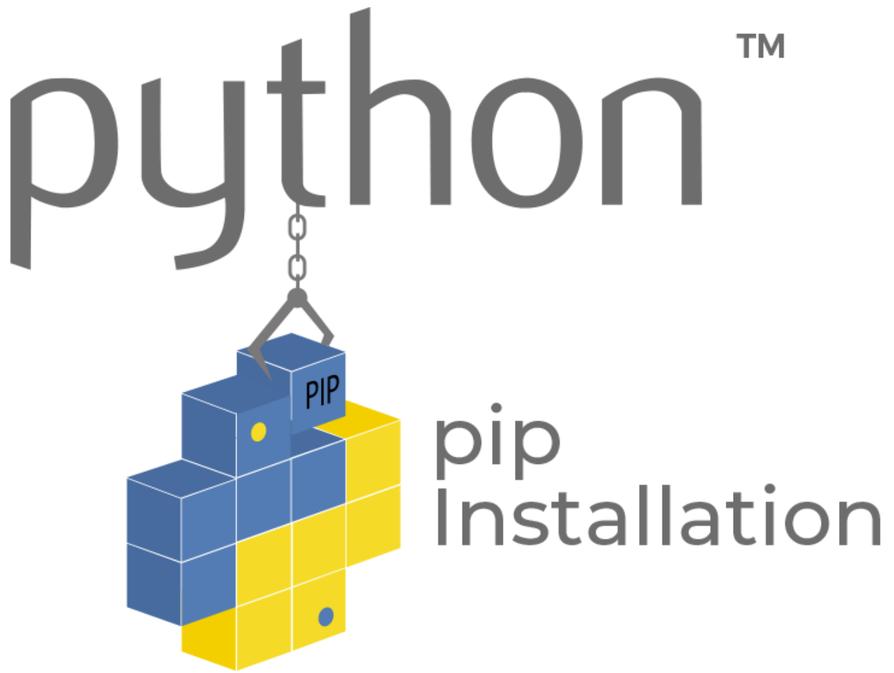
Python için Pip Paket Yöneticisi

Python için Paket Yükleyici, Python'da yazılmış fiili ve önerilen paket yönetim sistemidir ve yazılım paketlerini kurmak ve yönetmek için kullanılır.

```
pip -h
```

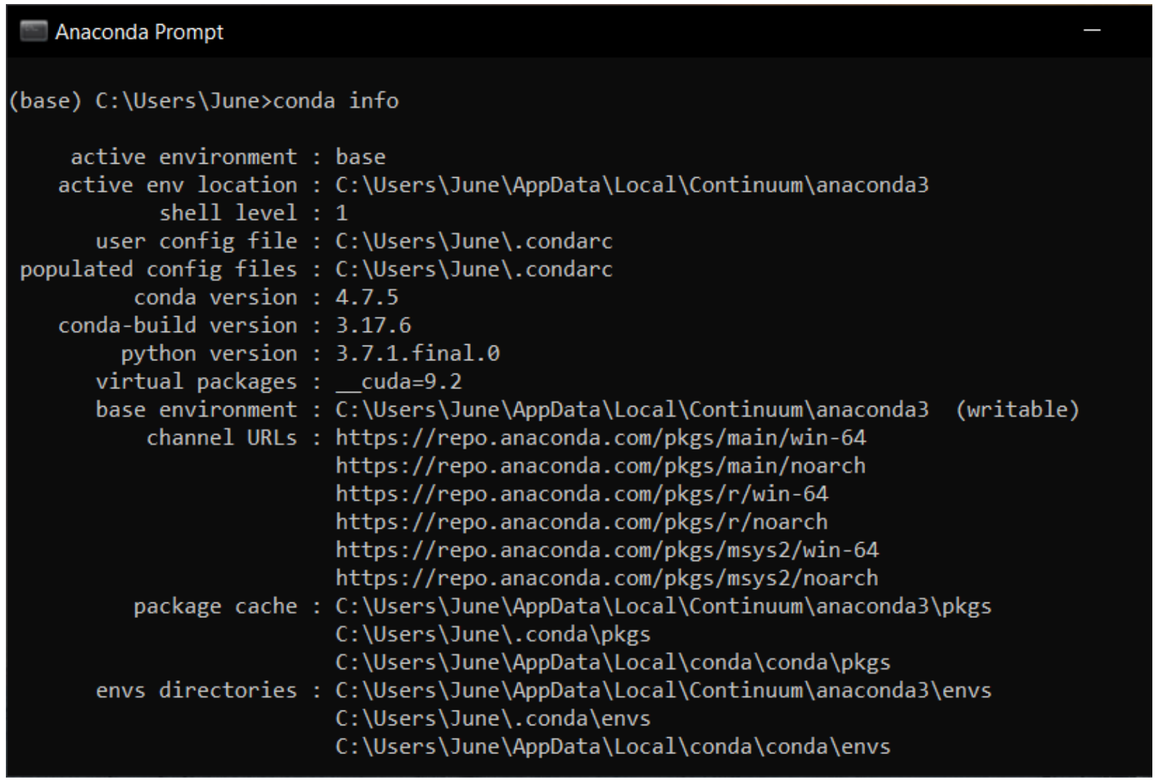
```
Usage:
  pip <command> [options]

Commands:
  install       Install packages.
  uninstall     Uninstall packages.
  freeze        Output installed packages in requirements format.
  list          List installed packages.
```



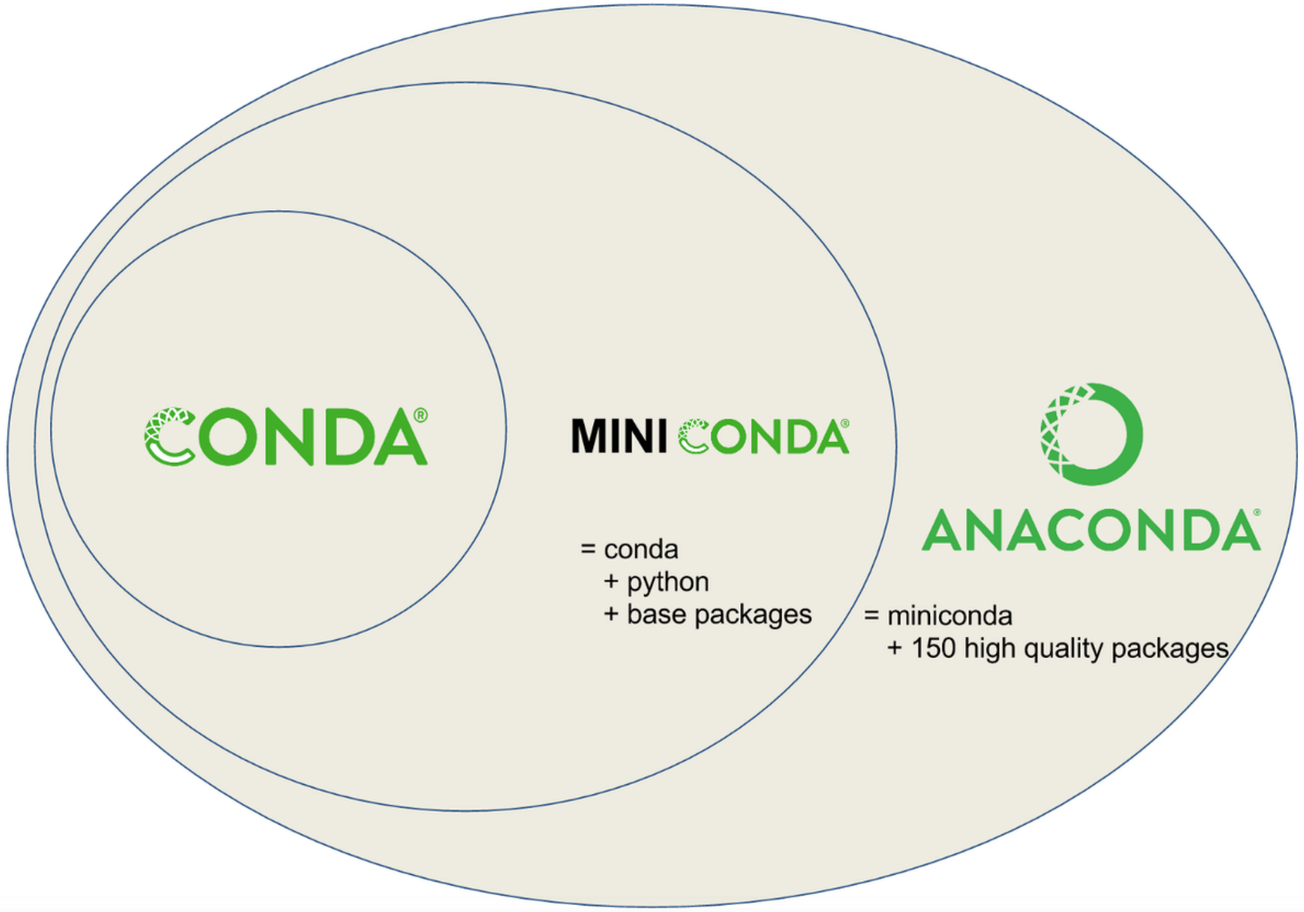
Python için Conda Paket Yöneticisi

Conda, açık kaynaklı, platformlar arası, dilden bağımsız bir paket yöneticisi ve çevre yönetim sistemidir. Başlangıçta Python veri bilimcilerinin karşılaştığı zorlu paket yönetimi zorluklarını çözmek için geliştirildi ve bugün Python ve R için popüler bir paket yöneticisidir.



Python için Paket Yöneticileri

Buna karşılık, conda, şu anda yüklü olan her şeyi içeren mevcut ortamı analiz eder ve belirtilen herhangi bir sürüm sınırlamasıyla birlikte (örneğin, kullanıcı Tensorflow sürüm 2.0 veya daha yüksek bir sürüme sahip olmak isteyebilir), uyumlu bir bağımlılık kümesinin nasıl kurulacağını ve bu yapılamazsa bir uyarı gösterir.



Anaconda IDE Genel Bakış

Açık kaynak paketleri, *conda install* komutu kullanılarak Anaconda deposundan, Anaconda Cloud'dan (anaconda.org) veya kendi özel *conda install* ile kurulur. PyPI'de mevcut olan her şey pip kullanılarak bir conda ortamına kurulabilir ve conda, neyin kurulduğunu ve hangi pip'in kurulduğunu takip edecektir.

Özel paketler *conda build* komutu kullanılarak yapılabilir ve Anaconda Cloud, PyPI veya diğer depolara yüklenerek başkalarıyla paylaşılabilir. Anaconda2'nin varsayılan kurulumu Python 2.7'yi ve Anaconda3 Python 3.7'yi içerir. Ancak, conda ile paketlenmiş herhangi bir Python sürümünü içeren yeni ortamlar oluşturmak mümkündür.

Anaconda Navigator

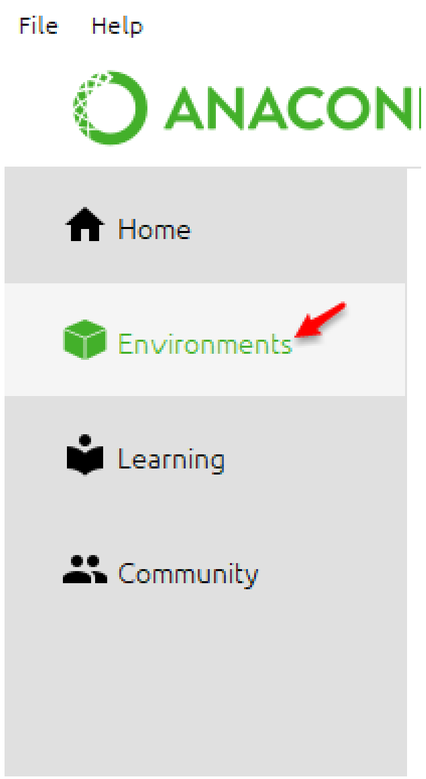
Anaconda Navigator, Anaconda dağıtımında bulunan ve kullanıcıların komut satırı komutlarını kullanmadan uygulamaları başlatmasını ve conda paketlerini, ortamları ve kanalları yönetmesini sağlayan bir masaüstü grafik kullanıcı arabirimidir (GUI). Navigator, paketleri Anaconda Cloud'da veya yerel bir Anaconda Deposunda arayabilir, bir ortama kurabilir, paketleri çalıştırabilir ve güncelleyebilir. Windows, macOS ve Linux için kullanılabilir.

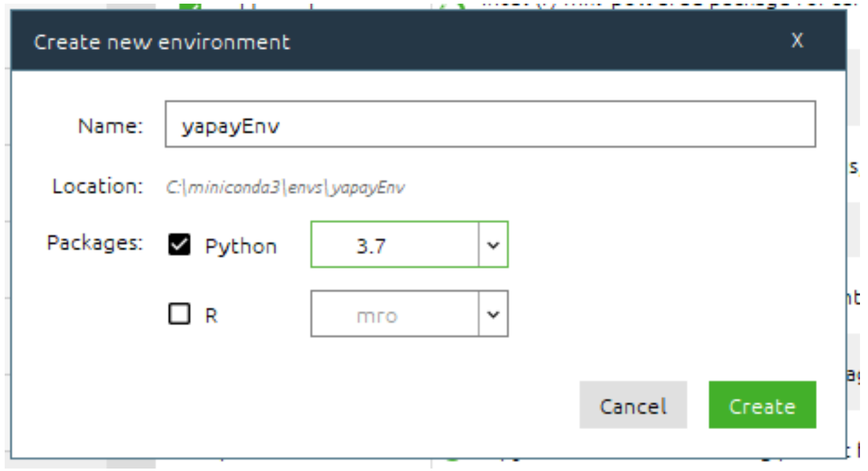
Navigator uygulamasında aşağıdaki uygulamalar varsayılan olarak mevcuttur:

- JupyterLab
- Jupyter Not Defteri
- QtConsole [8]
- Spyder
- Glue_ (yazılım)
- Orange
- RStudio
- Visual Studio Code

Sanal Ortam Oluşturma

Sanal ortamın tek amacı çalışma ortamının boyutunu küçültmek değildir. Sanal ortam, farklı projelerdeki paketleri yönetmeye yarar. Sanal ortam kullanmak, global olarak paket yükledikten sonra çıkabilecek sorunları ortadan kaldırır. Bu tür hatalardan kaçınmak için Python uygulamaları geliştirirken her zaman sanal ortam kullanılması önerilir.





Sanal Ortam Oluşturma

Eğer Anaconda Navigator yüklü değilse ya da kullanmak istemiyorsanız aşağıdaki komutlar yardımıyla da aynı işlemleri yapabilirsiniz. Miniconda'yı yükledikten sonra yeni bir Ortam oluşturmak için Anaconda Prompt ekranında;

> *$ conda create — name yapayEnv python=3.7*

Oluşturulan Ortam'ı aktive etmek için

> *$ activate yapayEnv*

Ortam'ta bulunan paketleri göstermek için

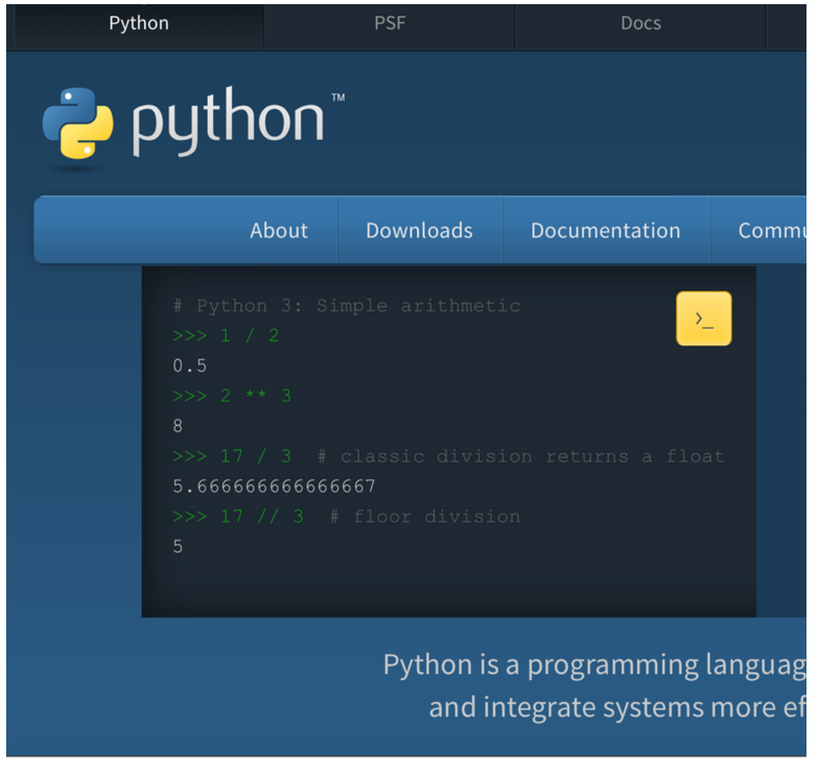
> *$ conda list*

komutlarını kullanabilirsiniz.

Python Programlama Dili

Python, nesne yönelimli, yorumlamalı, birimsel (modüler) ve etkileşimli yüksek seviyeli bir programlama dilidir.

Girintilere dayalı basit söz dizimi, dilin öğrenilmesini ve akılda kalmasını kolaylaştırır. Bu da ona söz diziminin ayrıntıları ile vakit yitirmeden programlama yapılmaya başlanabilen bir dil olma özelliği kazandırır.



Python Ne Yapabilir?

- Python, web uygulamaları oluşturmak için bir sunucuda kullanılabilir.
- Python, iş akışları oluşturmak için yazılımla birlikte kullanılabilir.
- Python, veritabanı sistemlerine bağlanabilir. Ayrıca dosyaları okuyabilir ve değiştirebilir.
- Python, büyük verileri işlemek ve karmaşık matematik işlemlerini gerçekleştirmek için kullanılabilir.
- Python, hızlı prototipleme veya üretime hazır yazılım geliştirme için kullanılabilir.

Python Programlama Dili - İşlemler

`*` işleci, çarpma işlemleri için

`/` işleci, bölme işlemleri için

`//` işleci, tam sayı bölme işlemleri için

`+` işleci, toplama işlemleri için

`-` işleci, çıkarma işlemleri için

`%` işleci, mod alma işlemleri için

`<` işleci, 'küçüktür' anlamına gelir

`>` işleci, 'büyüktür' anlamına gelir

`==` işleci, 'eşittir' anlamına gelir

`<=` işleci, 'küçük eşittir' anlamına gelir

`>=` işleci, 'büyük eşittir' anlamına gelir

`!=` işleci, 'eşit değil' anlamına gelir

`**` işleci, 'üs alma' anlamına gelir

`True` işleci, 'doğru' anlamına gelir

`False` işleci, 'yanlış' anlamına gelir

`and` işleci, 've' anlamına gelir

`or` işleci, 'veya' anlamına gelir

`not` işleci, 'değil' anlamına gelir.

İfadeler ve Akış Kontrolü

Python ifadeleri şunları içerir:

- `if` ifadesi, bir kod blokunu belli bir koşula bağlı olarak, `else` ve `elif` (else-if'in kısaltılması) ile birlikte çalıştırır.
- `for` ifadesi, yinelenebilir bir nesne üzerinden yineleme yapar, bu sırada ilgili bloktaki her ögeyi bir yerel değişkene atar.
- `while` ifadesi, koşulu doğru olduğu sürece o kod bloğunu çalıştırır.
- `try` ifadesi, eklenmiş kod bloğundan dolayı oluşan hataları yakalar ve except maddeleriyle; ayrıca finally blokundaki temizleme kodunun blok nasıl sonlanırsa sonlansın çalıştırılmasını sağlar.
- `class` ifadesi, bir kod blokunu çalıştırır ve onun yerel adalanını bir sınıfa atar, bu sayede o sınıf Nesne Yönelimli Programlamada kullanılabilir.
- `def` ifadesi ile fonksiyon tanımlanır.
- `with` ifadesi, bir kod blokunu bir içerik yöneticisine hapseder (örneğin bir kod bloku çalışıtırılmadan önce bir kilit edinir ve sonrasında bu kilidi çözer)
- `pass` ifadesi bir NOP görevi görür ve geçici kod bloku yerine kullanılabilir.
- `print` ifadesi bir ekrana yazdırma görevi görür. Bu ifade Python 3 sürümüyle birlikte `print()` fonksiyonu ile değiştirildi.

Her ifadenin kendi sözdizimi vardır, örneğin `def` ifadesi diğer ifadelerin genelinin aksine blokunu anında çalıştırmaz.

Yorum Satırları

```
#Tek satırlık bir python yorumu
```

```
"""Bu çok satırlı
bir python
yorumu"""
```

Veri Tipleri

Veri tipi	Değiştirilebilir	Açıklama	Örnekleri
`bool`	hayır	Boole'sal değer	`True` `False`

`bytearray`	evet	dızi	`bytearray(b'Some ASCII')`
			`bytearray(b"Some ASCII")`
			`bytearray([119, 105, 107, 105])`

`bytes`	hayır	Bytelardan oluşan dizi	`b'ASCII'`
			`b"ASCII"`
			`bytes([119, 105, 107, 105])`

Veri Tipleri

Veri tipi	Değiştirilebilir	Açıklama	Örnekleri
`dict`	evet	Komut çizelgesi	`{'anahtar1': 1.0, 3: False}{}`
`float`	hayır	Kayan nokta	`1.414`
`int`	hayır	Tamsayı	`42`
`list`	evet	Farklı veri tiplerinden oluşan bir liste, array olarak ifade edilebilir	`[4.0, 'string', True]` `[]`
`NoneType`	hayır	Değerin yokluğunu ifade eden veri tipi, diğer dillerde null olarak bilinir.	`None`

Veri Tipleri

Veri tipi	Değiştirilebilir	Açıklama	Örnekleri
`range`	hayır	İki sayı arasındaki aralık, `for` döngülerinde yaygın olarak kullanılır	`range(1, 10)` `range(10, -5, -2)`
`set`	evet	Sırasız küme, değerler tekrar edemez	`{4.0, 'string', True}` `set()`
`str`	hayır	Karakter dizisi	`'Vikipedi' "Vikipedi"` `""" Birden Fazla Satır"""`
`tuple`	hayır	değiştirilemez liste	`(4.0, 'string', True)` `('element',)()`

Neden Python?

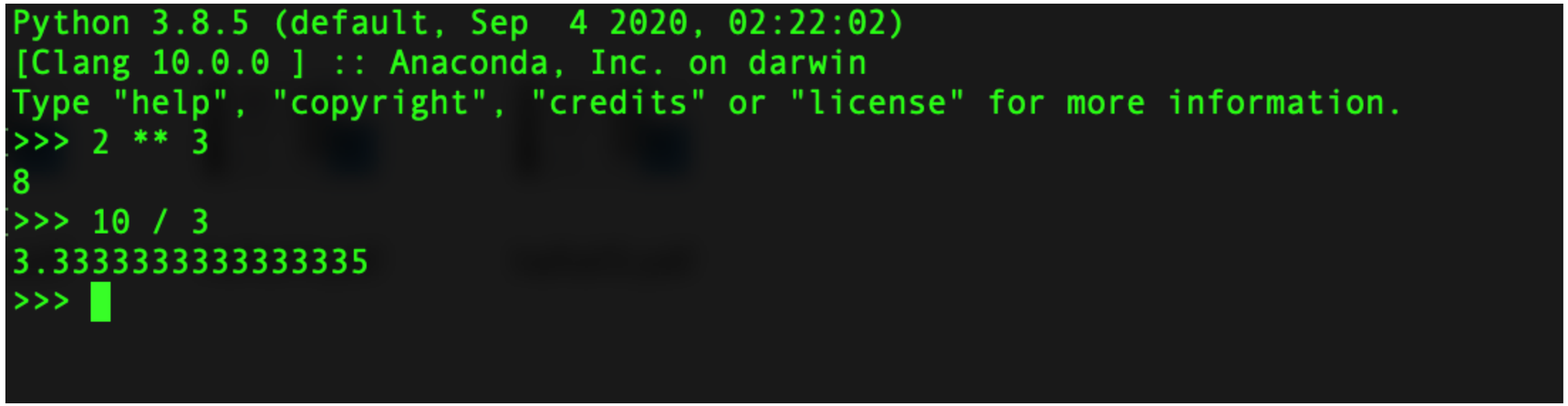
- Python farklı platformlarda çalışır (Windows, Mac, Linux, Raspberry Pi, vb.).
- Python, İngilizceye benzer basit bir sözdizimine sahiptir.
- Python, geliştiricilerin diğer bazı programlama dillerinden daha az satırlı programlar yazmasına izin veren sözdizimine sahiptir.
- Python bir yorumlayıcı sistemde çalışır, yani kod yazıldığı anda çalıştırılabilir. Bu, prototiplemenin çok hızlı olabileceği anlamına gelir.
- Python, prosedürel bir şekilde, nesne yönelimli bir şekilde veya işlevsel bir şekilde ele alınabilir.

Python Sözdizimi ve Diğer Programlama Dilleri

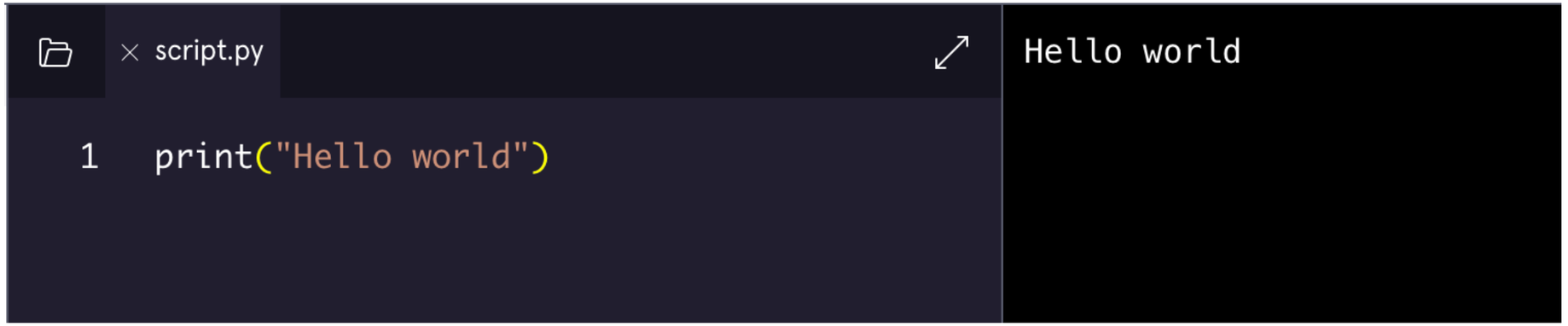
- Python okunabilirlik için tasarlanmıştır ve matematikten etkilenen İngilizce ile bazı benzerliklere sahiptir.
- Python, genellikle noktalı virgül veya parantez kullanan diğer programlama dillerinin aksine, bir komutu tamamlamak için yeni satırlar kullanır.
- Python, kapsamı tanımlamak için boşluk kullanarak girintiye güvenir; döngüler, fonksiyonlar ve sınıfların kapsamı gibi. Diğer programlama dilleri genellikle bu amaç için süslü parantezler kullanır.

Python’da Sayılar ve Aritmetik İşlemler

Python’un çalışma ekranında (IDLE) aritmetik işlem yaparken bir değişkene ihtiyaç duyulmadan rakamlar doğrudan kullanılabilir. Burada dikkat edilmesi gereken, sonucun yanlış çıktığı zamanlarda kullanılan sayı tiplerinin sonuca uygun olmamasıdır.



Python’da Başlangıç Kodları



Python’da Başlangıç Kodları

Python'da kısa miktarda kodu test etmek için bazen kodu bir dosyaya yazmamak en hızlı ve en kolay yoldur. Python'un kendisi bir komut satırı olarak çalıştırılabildiği için bu mümkün olur.

Windows, Mac veya Linux komut satırına aşağıdakini yazın:

```
C:\Users\Your Name>python
```

Python’da Başlangıç Kodları

Python'da kısa miktarda kodu test etmek için bazen kodu bir dosyaya yazmamak en hızlı ve en kolay yoldur. Python'un kendisi bir komut satırı olarak çalıştırılabildiği için bu mümkün olur. Windows, Mac veya Linux komut satırına aşağıdakini yazın:

```
C:\Users\Your Name>python
```

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Python’da Başlangıç Kodları

Python komut satırında işiniz bittiğinde, python komut satırı arayüzünden çıkmak için aşağıdakini yazmanız yeterlidir:

```
exit()
```

Python Programlama Dili - Girintileme

Python'da ifade bloklarını sınırlandırmak için süslü ayraçlar ya da anahtar kelimeler yerine beyaz boşluk girintileme kullanılır. Belli ifadelerden sonra girinti artar; girintinin azalması geçerli blokun sonlandığını gösterir.

```
#A is height B is radius

def cone (a, b):
    formula = (3.14 * .33 *a)*(b * b)
    return formula
```

Python’da Girintiler

Girinti, bir kod satırının başındaki boşlukları ifade eder.
Diğer programlama dillerinde koddaki girinti yalnızca okunabilirlik içindir, Python'daki girinti çok önemlidir.
Python, bir kod bloğunu belirtmek için girinti kullanır.

```
if 5 > 2:
  print("Five is greater than two!")
```

Sözdizimi hatası:

```
if 5 > 2:
print("Five is greater than two!")
```

Python’da Girintiler

Bir programcı olarak boşluk sayısı size kalmış, en yaygın kullanım dört ama en az bir olması gerekiyor.

```
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
        print("Five is greater than
two!")
```

Python’da Girintiler

Aynı kod bloğunda aynı sayıda boşluk kullanmanız gerekir, aksi takdirde Python size bir hata verecektir:

Sözdizimi hatası:

```
if 5 > 2:
 print("Five is greater than two!")
        print("Five is greater than two!")
```

Python’da Değişkenler

Bir veriyi kendi içerisinde depolayan birimlere değişken adı verilir. Değişken tanımlamada Python, veri tipi olarak karakter için string, tamsayı için int, ondalık için float komutları kullanılır. Bu veri tipleri gerektiğinde birbirlerine dönüştürülebilirler. Python'da, ona bir değer atadığınızda değişkenler oluşturulur:

```
x = 5
y = "Hello, World!"

print(x)
print(y)

x = 5
y = "John"
print(x)
print(y)

x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0

x = 5
y = "John"
print(type(x))
print(type(y))
```

```
5
Hello, World!
5
John
<class 'int'>
<class 'str'>
```

Tek veya Çift Alıntılar?

Dize değişkenleri, tek veya çift tırnak kullanılarak bildirilebilir:

```
x = "John"
print(x)
#double quotes are the same as single quotes:
x = 'John'
print(x)
```

```
John
John
```

Python - Değişken Adları

Bir değişkenin kısa bir adı (x ve y gibi) veya daha açıklayıcı bir adı (yaş, carname, toplam_hacim) olabilir. Python değişkenleri için kurallar:

- Değişken adı bir harf veya alt çizgi karakteri ile başlamalıdır.
- Değişken adı bir sayı ile başlayamaz.
- Değişken adı yalnızca alfasayısal karakterler ve alt çizgiler içerebilir (Az, 0-9 ve _).
- Değişken adları büyük/küçük harfe duyarlıdır (yaş, Yaş ve YAŞ üç farklı değişkendir).

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Geçersiz değişken adları:

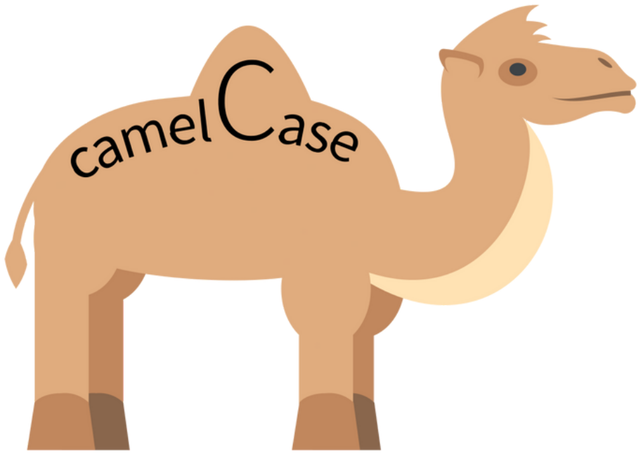
```
2myvar = "John"
my-var = "John"
my var = "John"
```

Çoklu Kelime Değişken İsimleri

Birden fazla kelime içeren değişken isimlerinin okunması zor olabilir.
Bunları daha okunaklı hale getirmek için kullanabileceğiniz birkaç teknik vardır:

1. **Camel Case**
   İlk kelime hariç her kelime büyük harfle başlar:

```
myVariableName = "John"
```



Çoklu Kelime Değişken İsimleri

2. **Pascal Case**
   Her kelime büyük harfle başlar:

```
MyVariableName = "John"
```

3. **Snake Case**
   Her kelime bir alt çizgi karakteri ile ayrılır:

```
my_variable_name = "John"
```

Referanslar

- Sams Teach Yourself Object Oriented Programming in 21 Days, Second Edition
- Mesleki ve Teknik A. L., Bilişim Teknolojileri, Nesne Tabanlı Programlama
- Nesne Tabanlı Programlama I, Kitap
- Python programlama, Wikipedia
- W3Schools, Tutorial

İletişim Bilgileri

ÖĞR. GÖR. BUSE YAREN TEKİN

bytekin@kastamonu.edu.tr