

FEF210 - Python ile Veri Analizi

Python ve Programlama Paradigmaları

Dr. Mustafa Murat ARAT | 18.12.2025

Şirketlerin ve Kurumların Yapay Zeka Diyince Aklına Gelen



ASLINDA OLAN



Kısıtlar

Yasal Kısıtlar

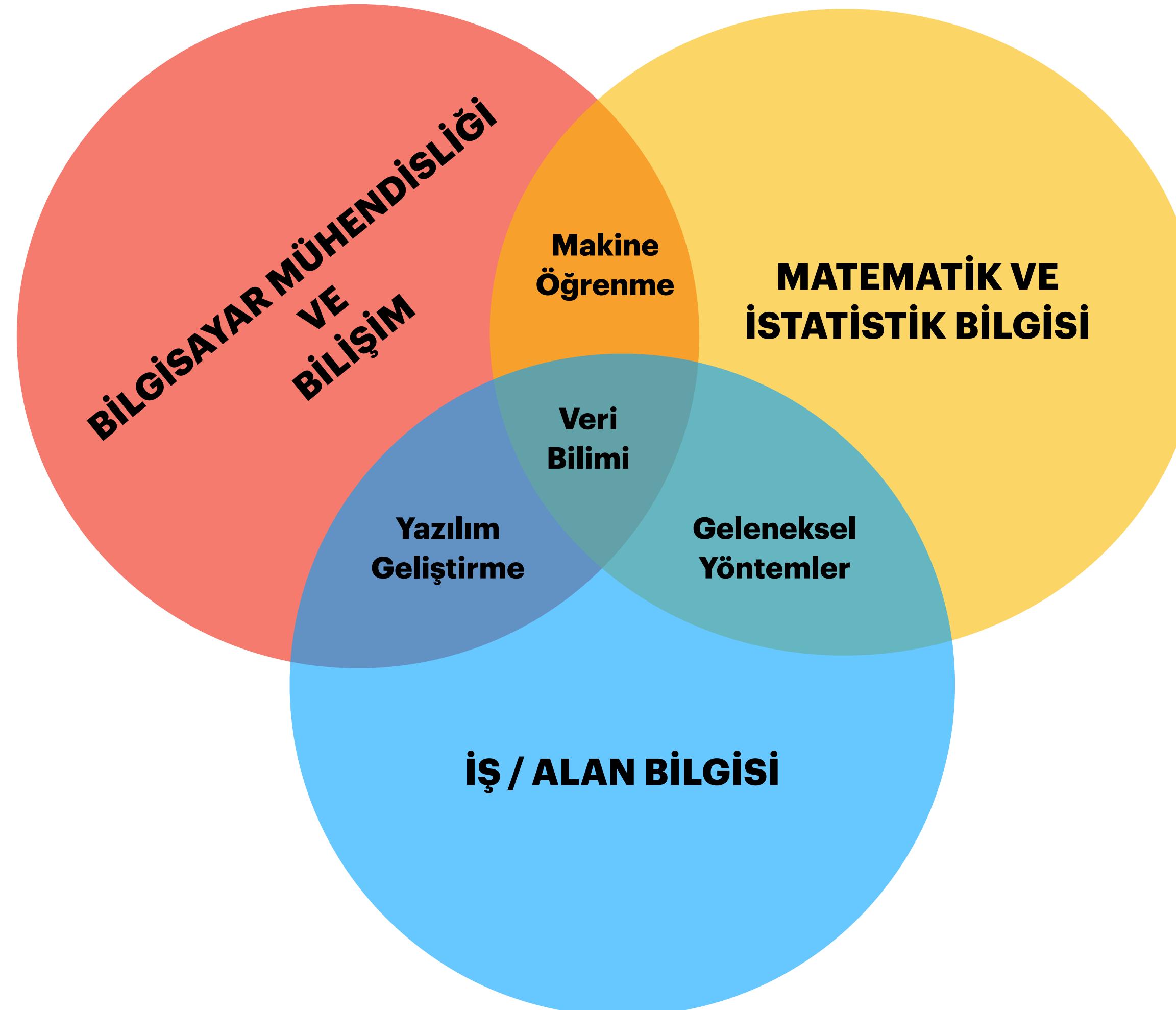
Etik Kısıtlar / Şeffaflık

Tarihi Kısıtlar (Yan)

Güvenlik

Veri Bilimi Venn Diyagramı

Drew Conway, 2013



Verimi bilimi nedir?

Veri Bilimi temelde disiplinler arası bir konudur, üç farklı ve örtüsen alandan oluşur:

- * (büyük) veri setlerinin nasıl modelleneceğini ve özetleneceğini bilen bir **istatistikçinin** becerileri;
- * Bu verileri verimli bir şekilde depolamak, işlemek ve görselleştirmek için algoritmalar tasarlayabilen ve kullanabilen bir **bilgisayar bilimcisinin** becerileri; ve
- * **alan uzmanlığı** - bir konuda "klasik" eğitim olarak düşünebileceğimiz şey - hem doğru soruları formüle etmek hem de yanıtlarını bağlama oturtmak için gereklidir.

Peki, Veri Bilimcisi nedir?

Josh Wills, Ex-statistician. Software engineer

Josh Wills
@josh_wills

...

Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

7:55 PM · May 3, 2012 · Twitter Web Client

1,815 Retweets **80** Quote Tweets **1,912** Likes

言论图标 转发图标 喜欢图标 分享图标

Veri Bilimi Alanları

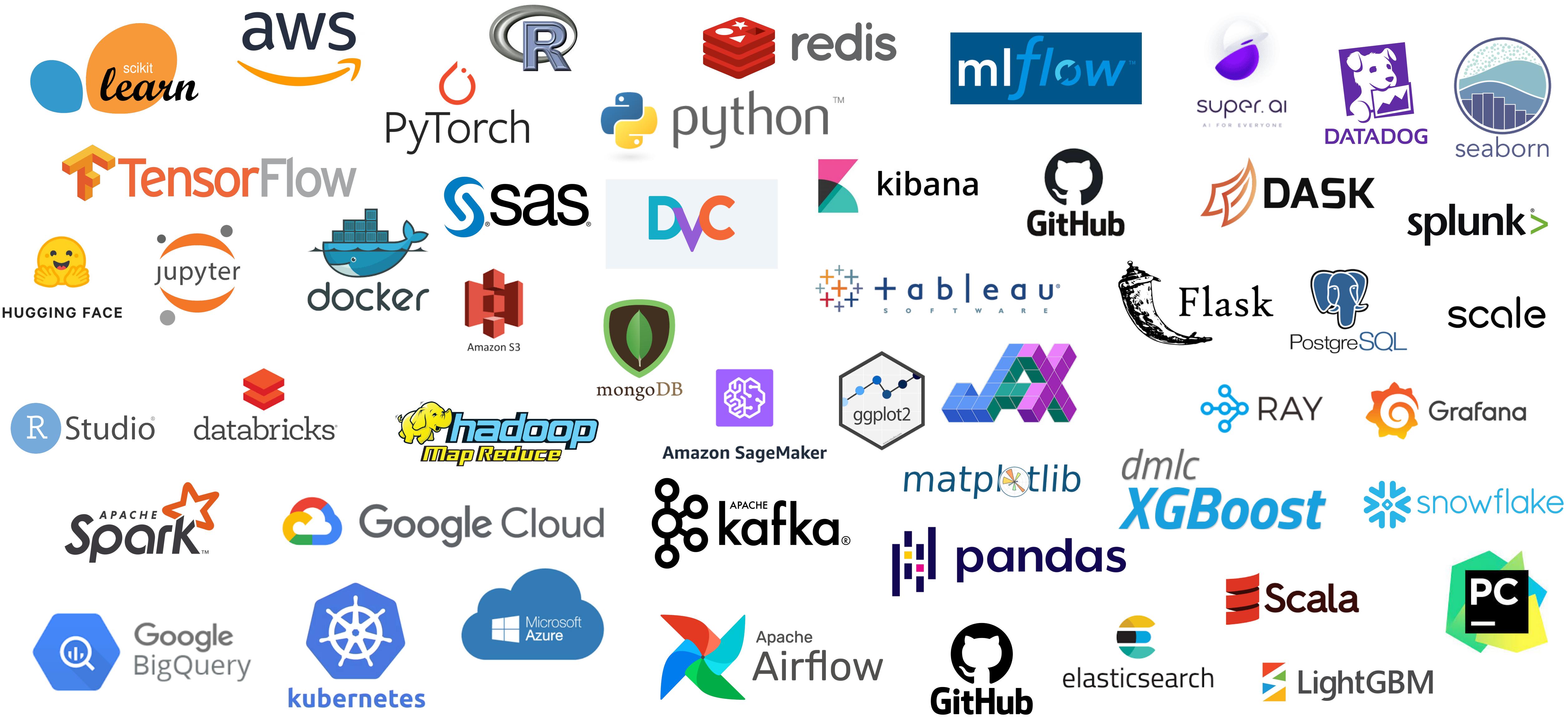
- Büyük Veri
- Veri Mühendisliği ve Veri Ambarlama
- Veri Madenciliği ve İstatistiksel Veri Analizi
- Bulut ve Dağıtılmış Bilgi İşlem
- Veritabanı Yönetimi ve Mimarisi
- İş Zekası ve Stratejisi
- Makine Öğrenmesi ve Derin Öğrenme
- Veri Görselleştirme ve Sunum
- Yapay Zeka / Bilgisayarlı Görü / Doğal Dil İşleme
- Sektöre Özgü Veri Analitiği

Sektöre Özgü Veri Analitiği

- Devlet İşleri
- Medya, Sosyal Medya ve Teknoloji
- İnternet ve Siber Güvenlik
- Telekom
- Bankacılık / Finans
- Sigortacılık
- Eczacılık, Tıp ve Sağlık Bilimleri
- Pazarlama, Tedarik Zinciri, Perakende ve İmalat
- Eğitim
- Seyahat ve Ulaşım
- Biyoteknoloji
- Spor



Bir Veri Bilimcinin Alet Takımı



Veri ve İş Problemi

Herhangi bir veri biliminin iş akışının temel parçasının *Veridir*.

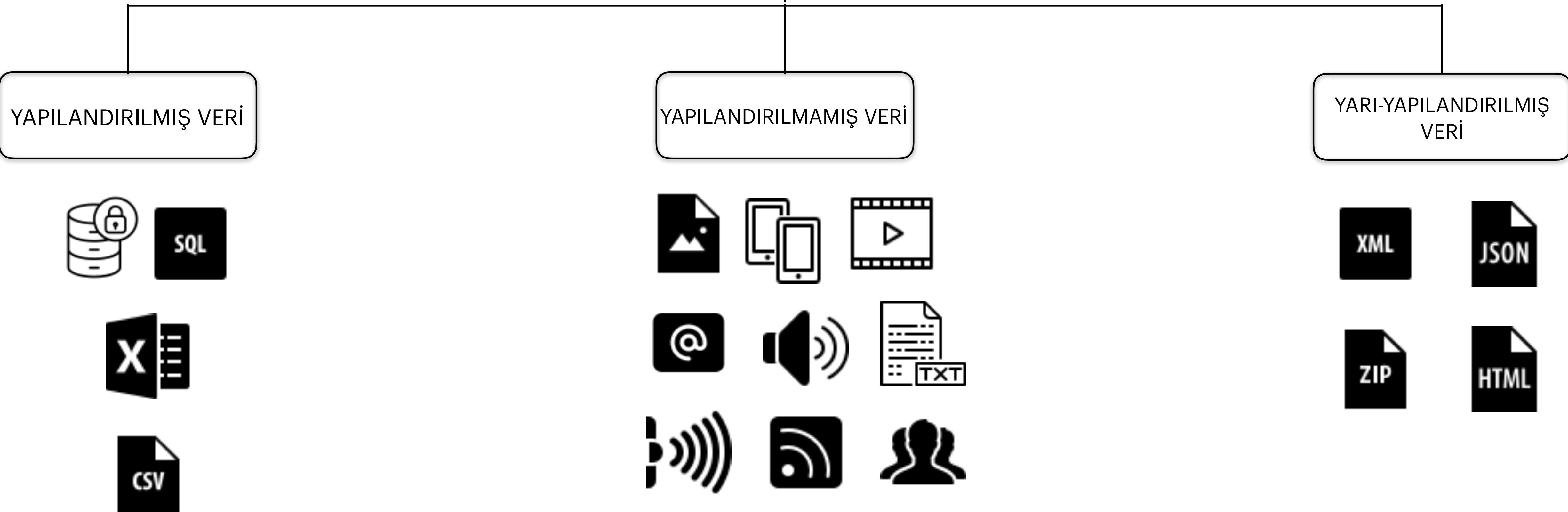
İyi veri kümelerinin toplanması, Makine Öğrenmesi modelinin kalitesi ve performansı üzerinde büyük bir etkiye sahiptir. Literatürdeki meşhur alıntı

“Çöp içeri, çöp dışarı (Garbage In, Garbage Out)”

makine öğrenmesi bağlamında, bir makine öğrenmesi modelinin yalnızca elinizdeki verileriniz kadar iyi olduğu anlamına gelir. Bu nedenle, bir makine öğrenmesi modelinin eğitilmesi için kullanılan veriler dolaylı olarak üretim sisteminin genel performansını etkilemektedir.

Veri kümelerinin miktarı ve kalitesi genellikle elinizdeki probleme göre değişebilir ve deneySEL olarak incelemesi yapılabilir.

VERİ TÜRLERİ

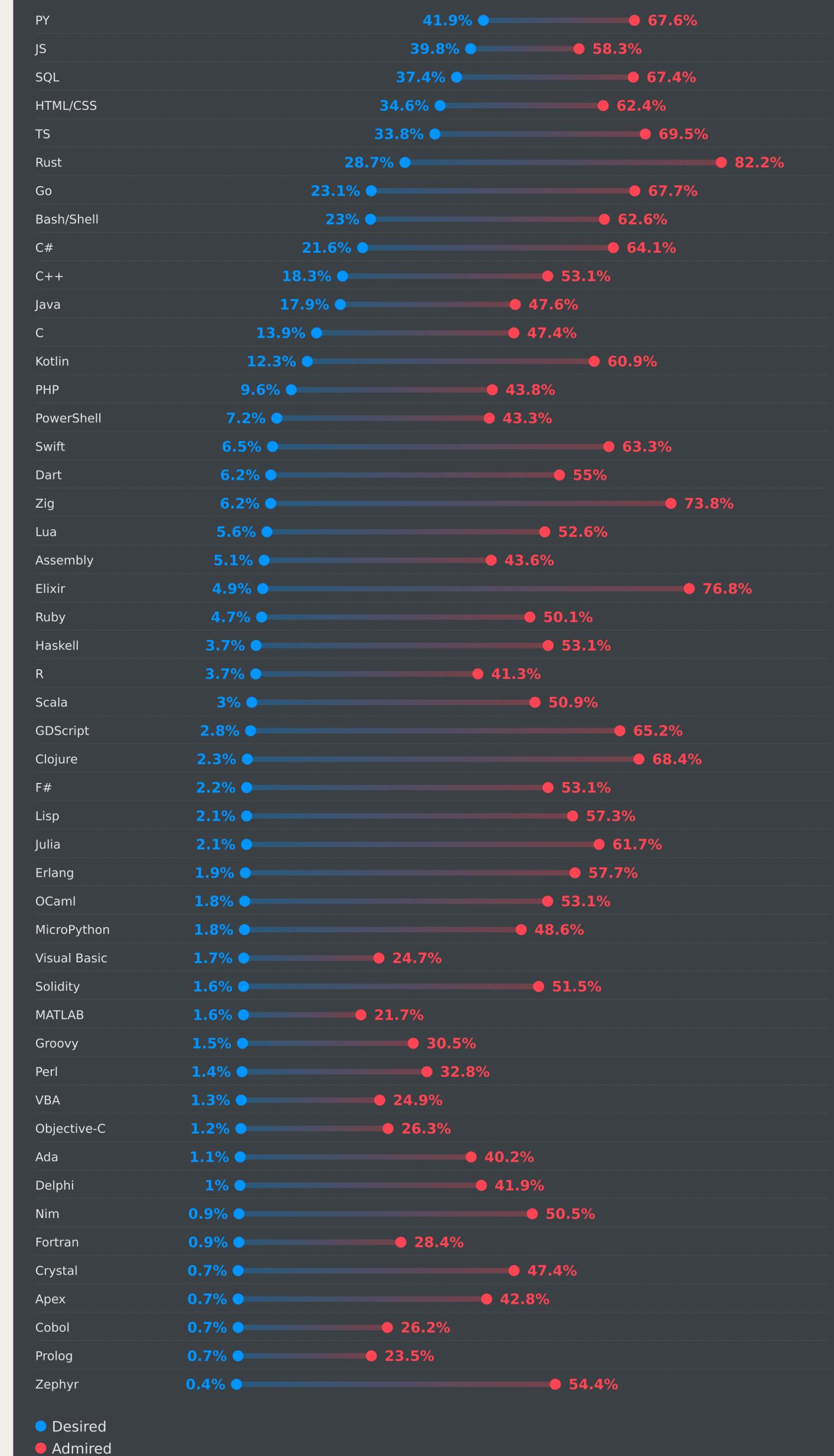


2024 Stackoverflow Developer Survey

<https://survey.stackoverflow.co/2024/>

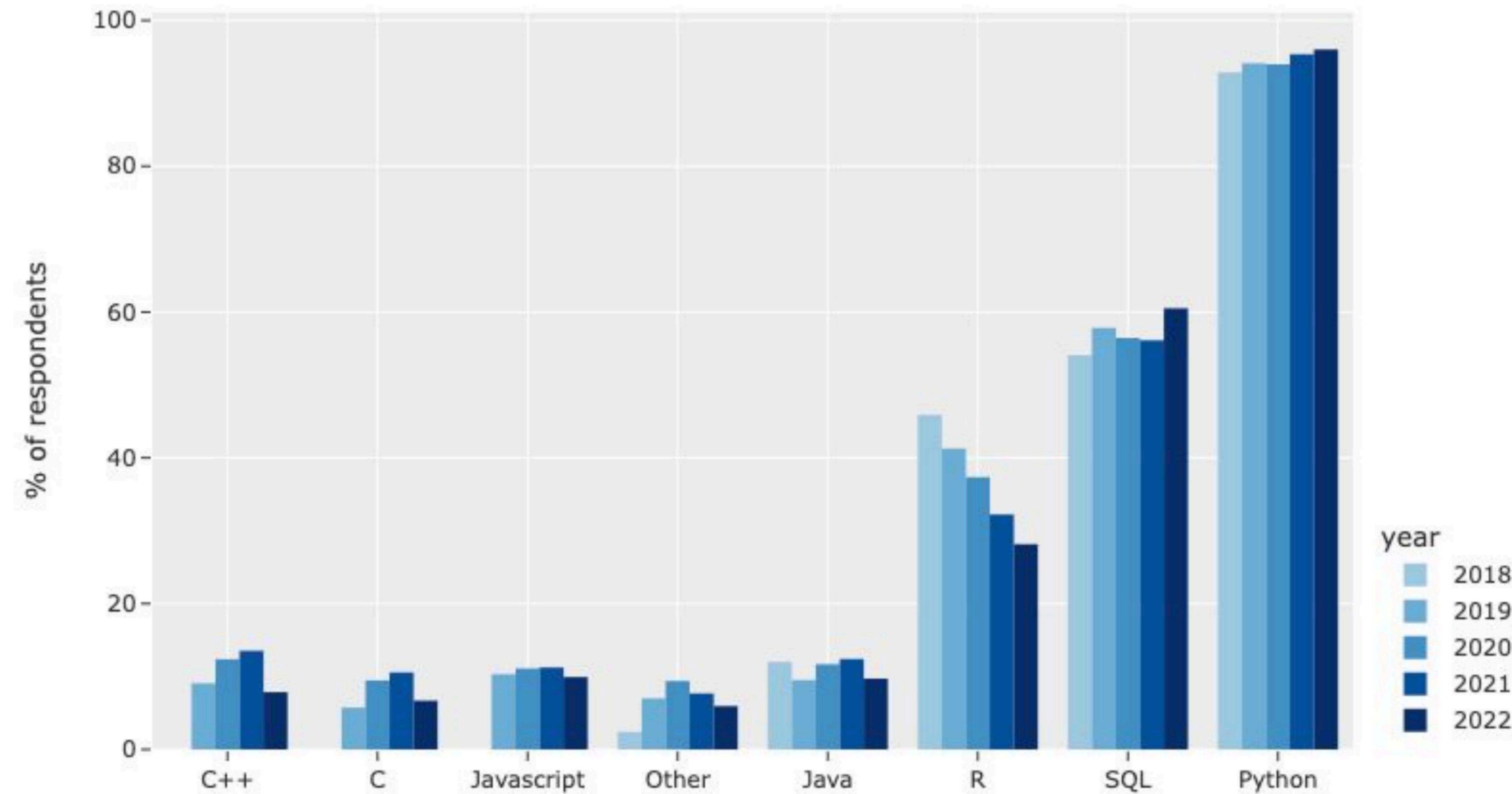
Admired and Desired / Desired and Admired

Programming, scripting, and markup languages



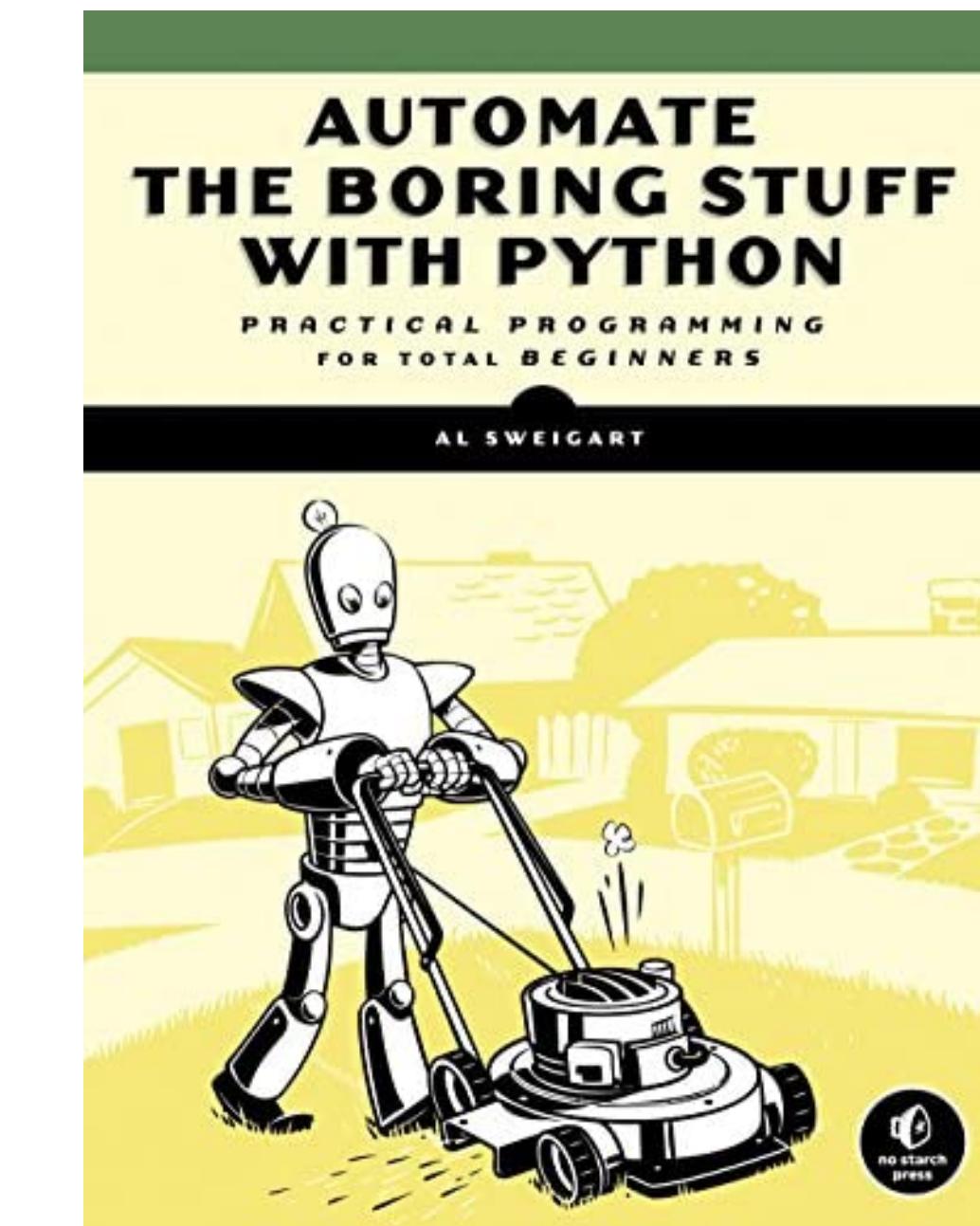
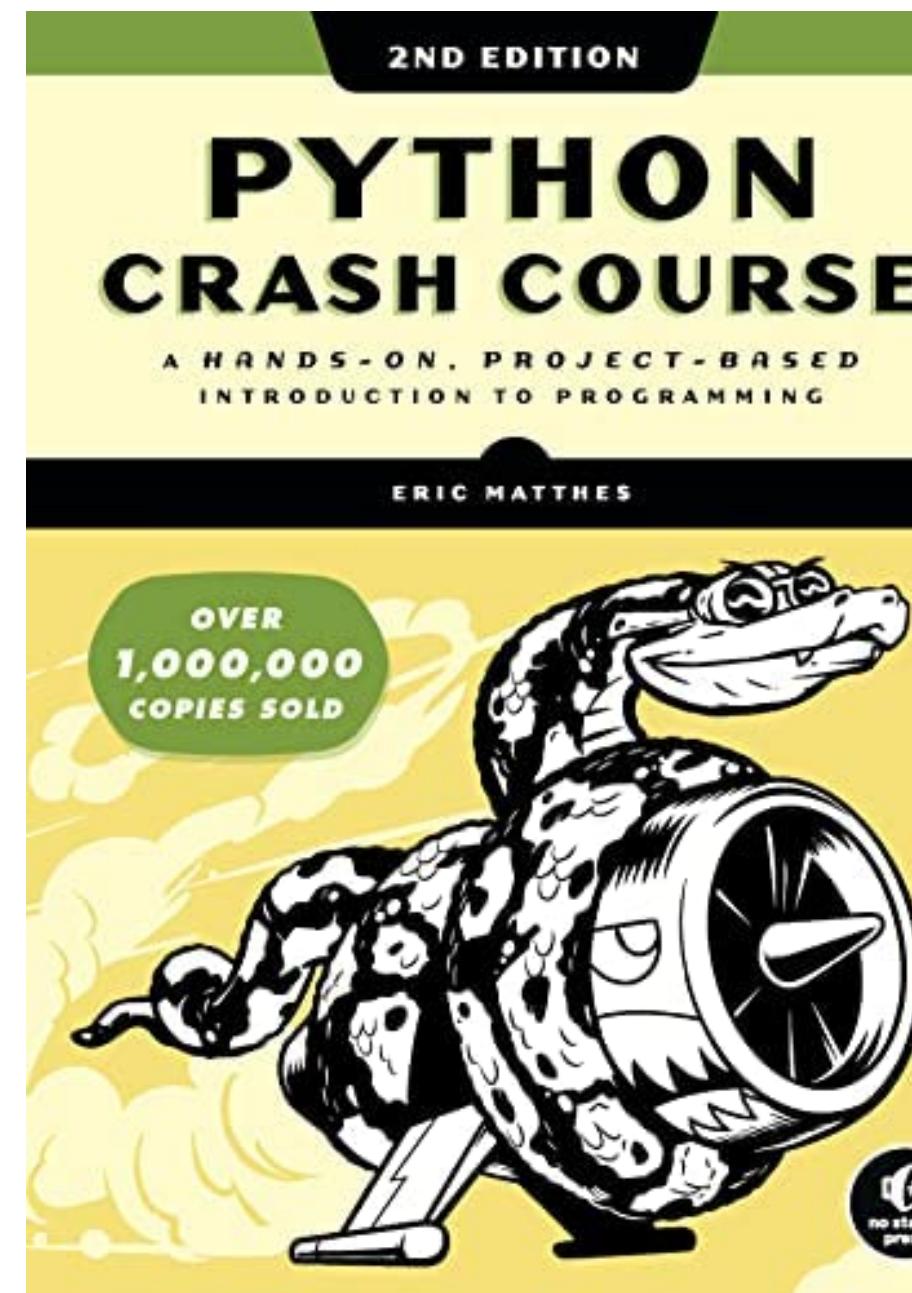
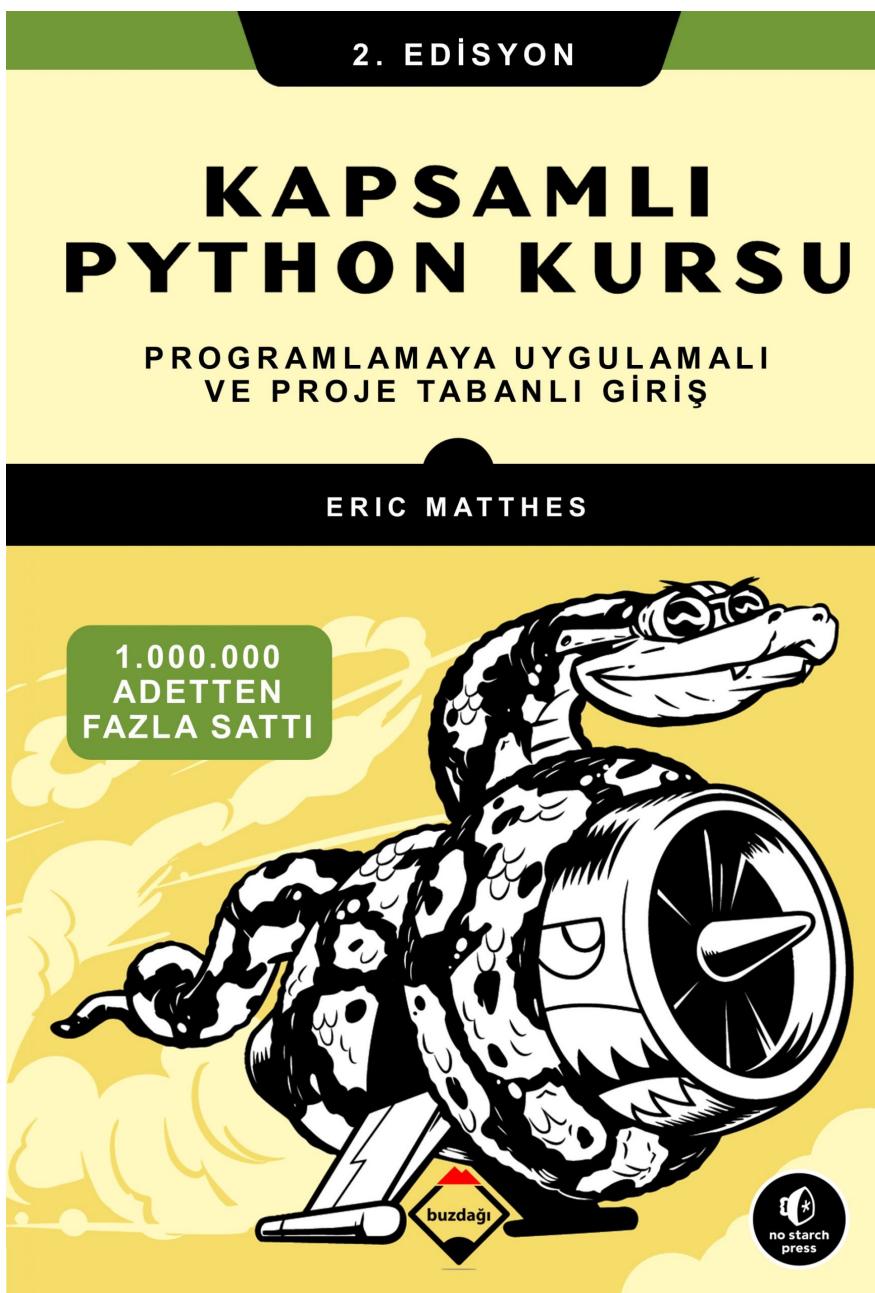
2022 Kaggle Veri Bilimi ve Makine Öğrenmesi Anketi

Veri bilimcileri için programlama becerilerine bakıldığında, Python ve SQL en yaygın yazılım dili olmaya devam etmektedir. R, Java, JavaScript, C ve C++ tercih edilen diğer dillerdir.



Python

Python, veri bilimi için giderek daha popüler hale gelen genel amaçlı bir programlama dilidir.



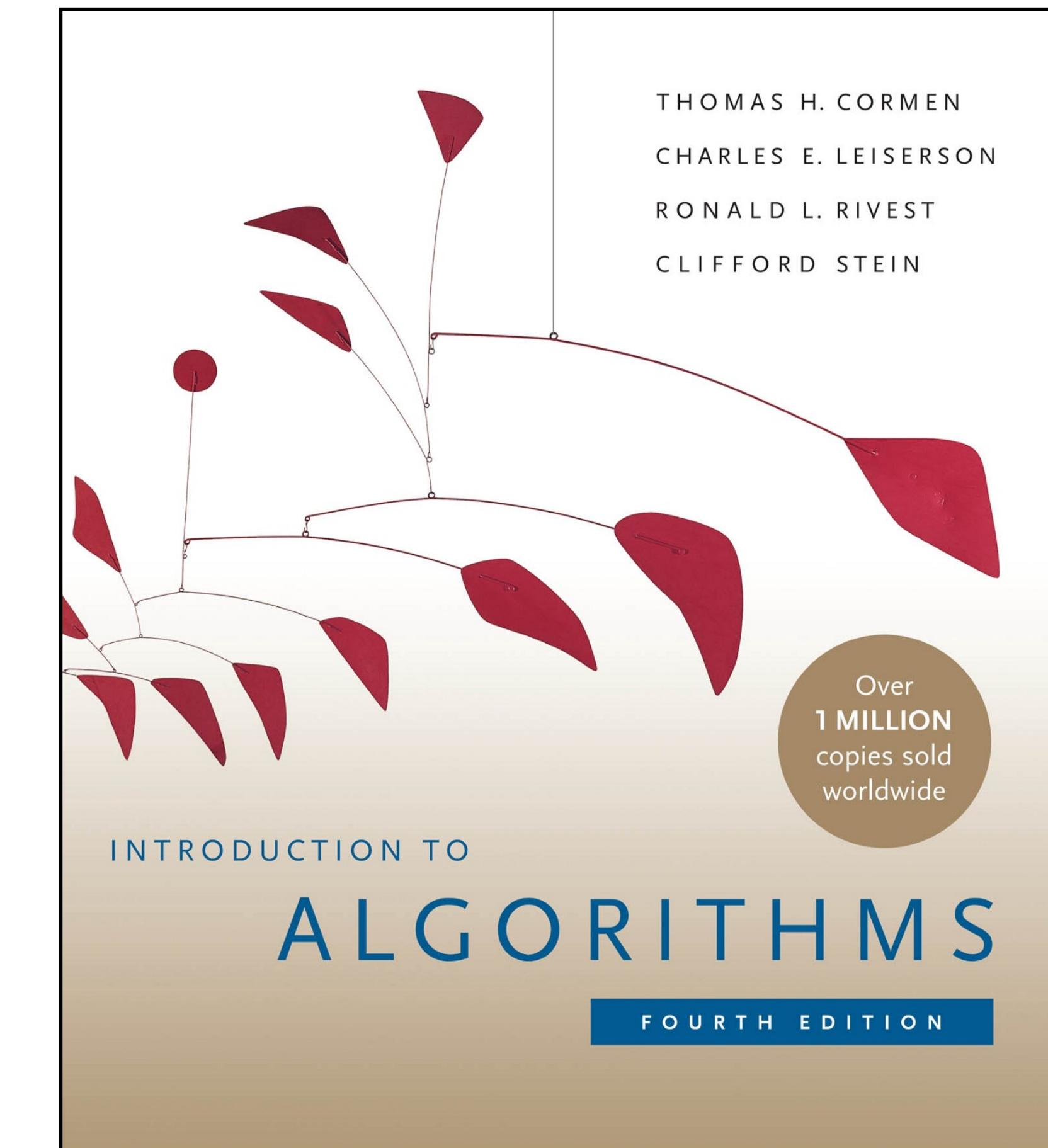
Python Editörleri

- Jupyter Lab / Jupyter Notebook
- PyCharm
- Visual Studio Code
- Google Colaboratory - <https://colab.research.google.com>
- Amazon SageMaker Studio Lab - <https://studiolab.sagemaker.aws>

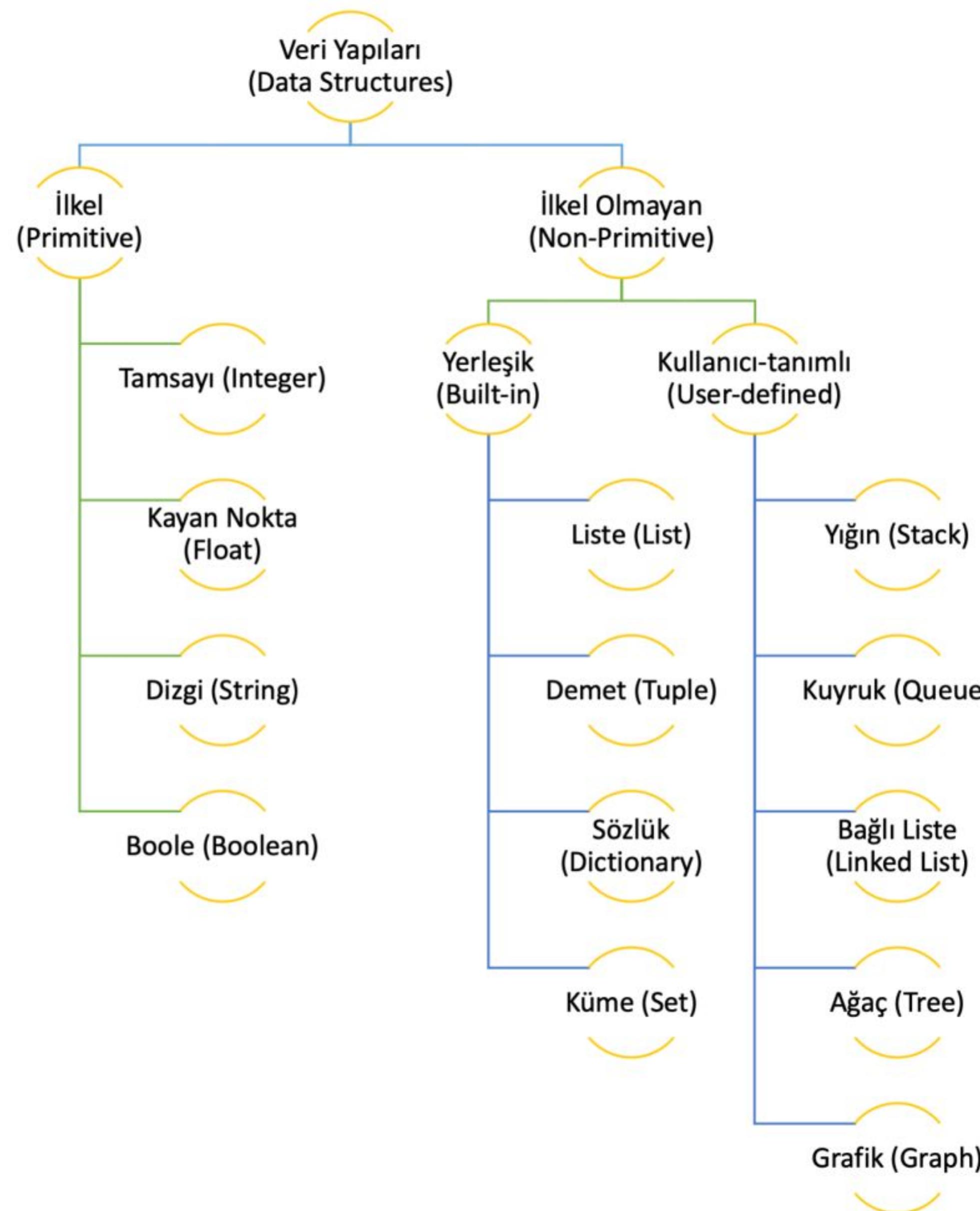
Veri Yapıları ve Algoritmalar

Veri yapısı ve algoritmalar, bilgisayar biliminin en önemli yönlerinden ikisidir. Veri yapıları, verileri düzenlememize ve saklamamıza izin verirken, algoritmalar bu verileri anlamlı bir şekilde işlememize izin verir.

1. Arrays
2. Linked Lists
3. Doubly Linked List
4. Circular Linked List
5. Circular Doubly List
6. Stacks
7. Queues
8. Hash Tables
9. Trees
10. Heaps
11. Graphs



Python - Veri Yapıları ve Algoritmalar

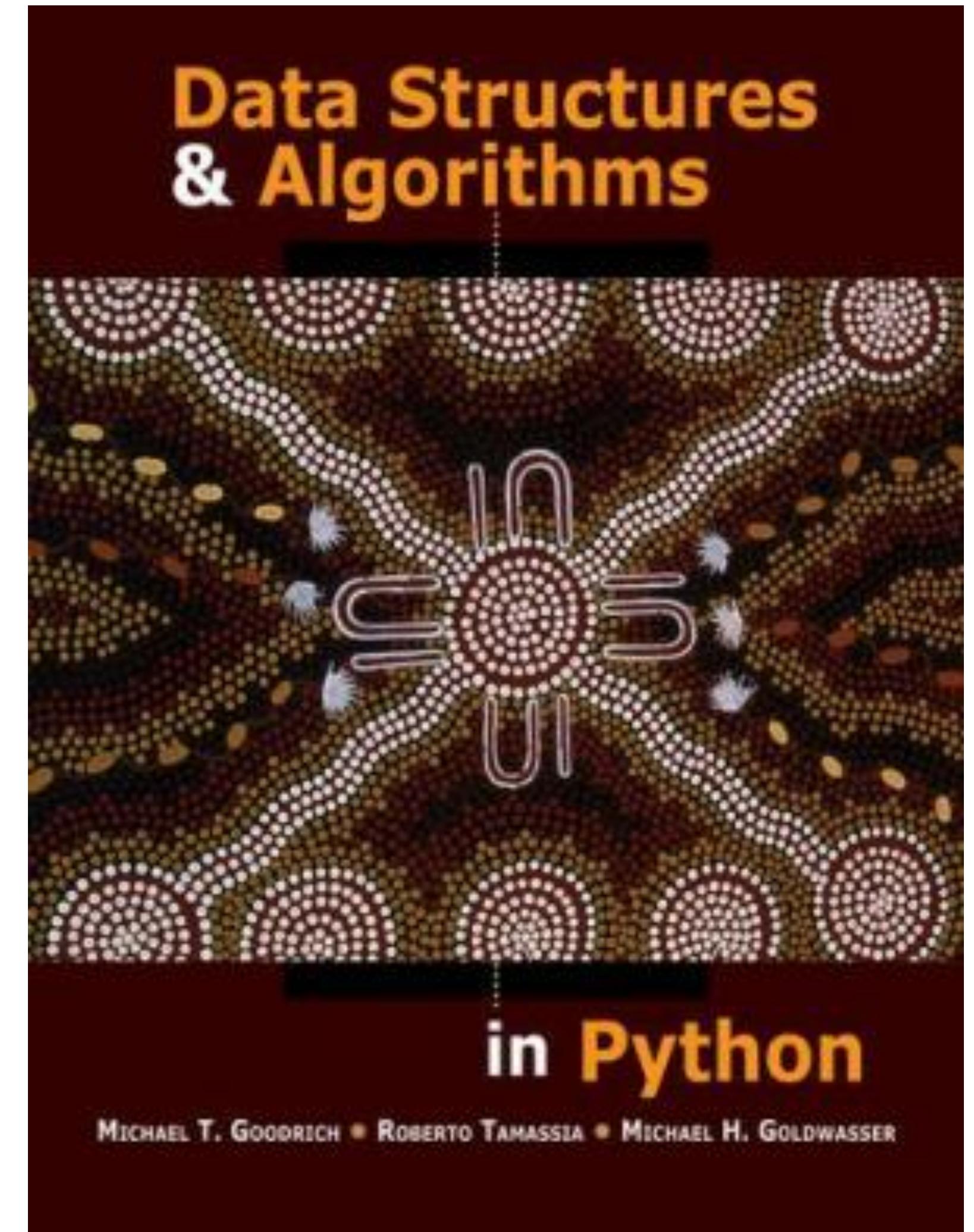


Her veri yapısı ve algoritması her programlama dilinde bulunabilir. Örneğin, Python, kayan nokta (float), tamsayı (integer), dizgi (string) ve Boole (Boolean) gibi ilkel (primitive) veri yapılarına sahiptir. Python, ayrıca, liste (list), demet (tuple), sözlük (dictionary) ve küme (set) gibi ilkel olmayan veri yapılarına sahiptir. İlkel olmayan veri yapıları, tek bir değer yerine çeşitli biçimlerdeki bir değerler koleksiyonunu depolar. Bazıları, veri yapıları içerisinde başka veri yapıları tutabilir ve veri depolama yeteneklerinde derinlik ve karmaşıklık sağlayabilir.

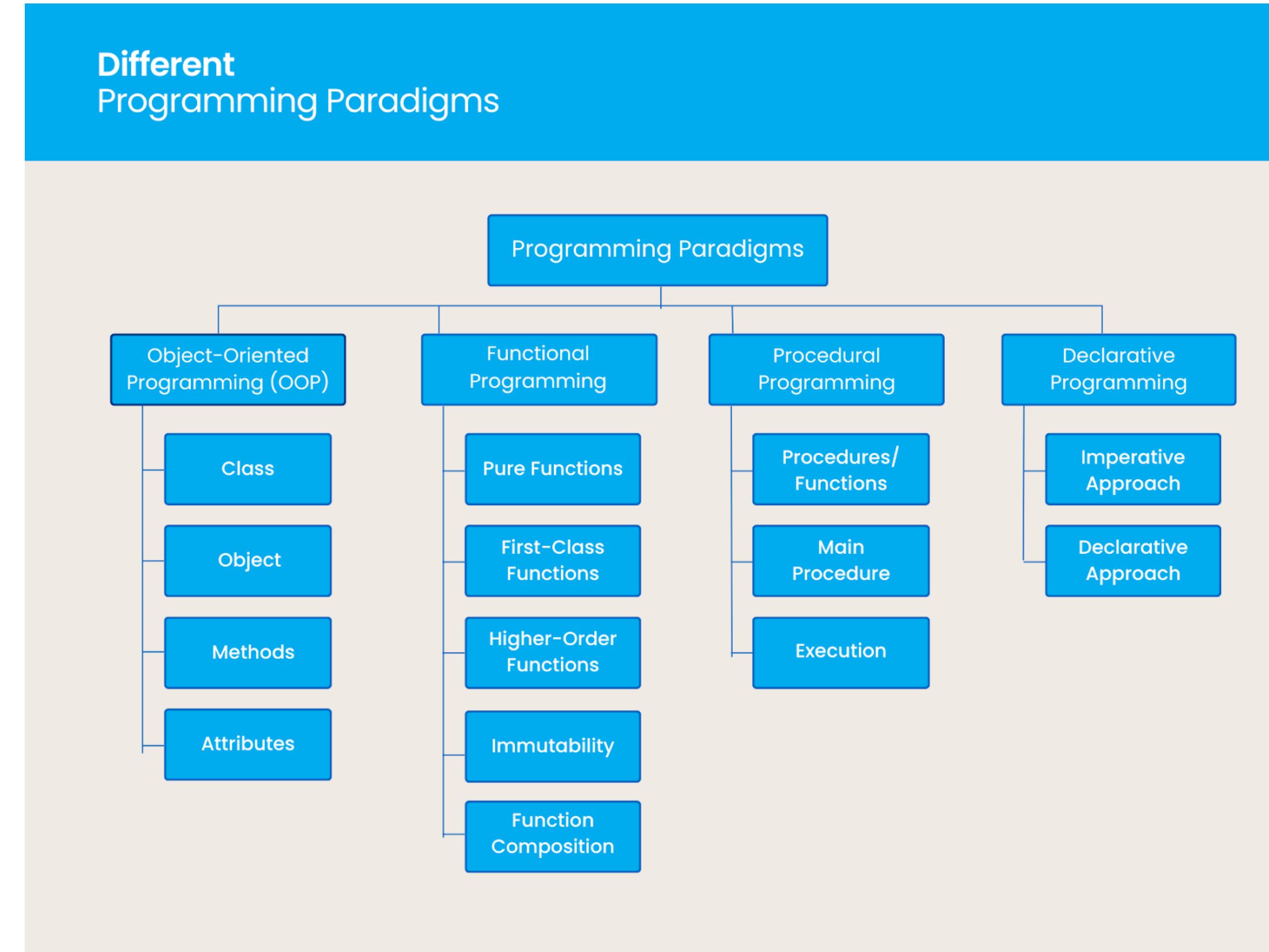
Bazı diller yerleşik olarak bazı veri yapılarını ve algoritmalarını destekleyebilirken, bazıları destekleyemez. Ancak, bu, o yapıyı veya algoritmayı, seçtiğiniz programlama dilinde oluşturamayacağınız anlamına gelmez. Örneğin, Yığınlar (Stacks) ve Kuyruklar (Queues), bilgisayar biliminde tanımlanan en eski veri yapılarıdır ve Python, ne yazık ki, native olarak bu yapıları desteklememektedir. Ancak, siz kendiniz bu yapıları bir Class (sınıf) ile kullanıcı tanımlı olarak kolaylıkla oluşturabilirsiniz.

Python - Veri Yapıları ve Algoritmalar

Python'da Veri Yapıları ve Algoritmaları üzerine kendinizi geliştirmek istiyorsanız, en mükemmel kaynak Michael T. Goodrich, Roberto Tamassia, ve Michael H. Goldwasser tarafından yayınlanan "Data Structures and Algorithms in Python" kitabıdır.



Programlama Paradigmaları



<https://www.freecodecamp.org/news/an-introduction-to-programming-paradigms/>

Python

Python, emirli (imperative), prosedürel (procedural), fonksiyonel (functional) ve nesne-tabanlı programlama (object-oriented programming) olmak üzere dört ana paradigmayı destekleyen ve genel amaçlı programlama (general purpose programming) için tasarlanmış üst düzey bir programlama dilidir (programming language).



İnsanların "**çoklu-paradigma (multi-paradigm)**" dedikleri zaman kastettikleri budur.

Python

Programlama paradigmaları, programlama dillerini özelliklerine göre sınıflandırmanın bir yoludur.

Programlama paradigmaları, belirli bir programın ya da programlama dilinin düzenlenebileceği farklı yaklaşımlar veya stiller olarak tanımlanır. Her paradigma, belirli yapıların, özelliklerin ve yaygın programlama problemlerinin nasıl ele alınması gereğine dair görüşlerin bir birleşimini içerir.



Neden birden fazla programlama paradigmاسının var olduğu sorusu, neden birçok programlama dilinin var olduğu sorusuya benzerlik göstermektedir. Bazı paradigmalar belirli problem türlerine daha uygun olduğundan, farklı projelerde farklı paradigmaların kullanılması mantıklıdır.

Python

Ortak programlama paradigmaları **emirli (imperative)** ve **bildirimli (declarative)**'dır.

Imperative sözcüğü Türkçe'de zorunlu, gerekli, emirli anlamına gelmektedir. Declarative sözcüğü ise bildirici, açıklayıcı, ifade edici anlamındadır.

Programlamada ise imperative, bir işlemi nasıl yapacağını (how it will happen) anlattığın, declarative ise bir işlemin ne yapacağını (what will happen) anlattığın programlama şekli olarak tanımlanır.



Python

Python, emirli (imperative) bir dildir!

Emirli paradigmada, bir eylem için gereken tüm adımları belirlersiniz ve bilgisayar, çıktıları döndürmek için bu adımları yürütür.

SQL, bildirimli (declarative) bir dildir.

Bildirimli paradigmada ise, istediğiniz çıktıları belirlersiniz ve bilgisayar, size, sorgulanınan çıktıları elde etmek için gereken adımları belirler.



Python

Python'da, istenilen sonuca ulaşmak için bilgisayarın atması gereken adımları açıkça tanımlayan kodlar yazarsınız. Bu, makineye bir dizi komut vermeyi içerir.

```
...  
  
# Example: Calculate the sum of the first 10 natural numbers  
  
# Initialize the sum variable  
sum_of_numbers = 0  
  
# Iterate over the first 10 natural numbers  
for number in range(1, 11):  
    # Add each number to the sum  
    sum_of_numbers += number  
  
# Output the result  
print("The sum of the first 10 natural numbers is:", sum_of_numbers)
```

Python

```
...  
# Example: Calculate the sum of the first 10 natural numbers  
  
# Initialize the sum variable  
sum_of_numbers = 0  
  
# Iterate over the first 10 natural numbers  
for number in range(1, 11):  
    # Add each number to the sum  
    sum_of_numbers += number  
  
# Output the result  
print("The sum of the first 10 natural numbers is:", sum_of_numbers)
```

Açıklama:

Bu kod, bilgisayara açık bir şekilde bir değişkeni (`sum_of_numbers`) başlatmasını (`initialize`), belirli bir sayı aralığında (1'den 10'a kadar) döngü oluşturmasını ve her adımda mevcut sayıyı `sum_of_numbers` değişkenine eklemesini talimat verir.

Son olarak, sonucu ekrana yazdırır.

Kodun her satırı, bilgisayarın gerçekleştirmesi gereken belirli bir eylemi tanımlar ve bu, emirli (imperative) paradigmanın karakteristik bir özelliğidir.

Python

SQL (Structured Query Language - Yapılandırılmış Sorgu Dili), neyin yapılmasını istediğinizizi belirttiğiniz, ancak nasıl gerçekleştirileceğini belirtmediğiniz bildirimli programlama paradigmasının mükemmel bir örneğidir. Veritabanı (database) motoru, sorguyu (query) yürütmenin en uygun yolunu kendi belirler.

...

```
SELECT first_name, salary  
FROM employees  
WHERE department = 'Sales';
```

Python

...

```
SELECT first_name, salary  
FROM employees  
WHERE department = 'Sales';
```

Açıklama:

Bu sorguda (query), ‘Satış (Sales)’ departmanındaki tüm çalışanların ilk isimlerini ve maaşlarını istediğinizizi belirtiyorsunuz.

Veritabanının (database) bu bilgiyi nasıl getirmesi gerektiğini (örneğin, hangi algoritmayı kullanması gerektiğini veya tabloyu (table) nasıl taraması gerektiğini) belirtmiyorsunuz.

SQL motoru (engine), sorguyu arka planda en verimli şekilde yürütmenin yolunu belirler ve bu, bildirimli programmanın özünü oluşturur. Emirli programmanın aksine, SQL kontrol akışını soyutlayarak doğrudan istenen sonuca odaklanır.

Python

Bazı programlama dilleri belirli bir paradigmaya bağlı kalmak üzere tasarılanırken, diğer diller birden çok paradigmaya sahip olabilir. Örneğin, C++, C# , Java, JavaScript, Visual Basic, Python ve Object Pascal, birden çok paradigmayı destekleyebilen programlama dillerinin örnekleridir.

Buraya geri döneceğiz!

Python

C ile yazılan Python uygulamasına CPython da denilmektedir.
<https://stackoverflow.com/a/17130986>

CPython, Python dilinin varsayılan ve en yaygın olarak kullanılan geleneksel (original) uygulamasıdır.

Ancak, birkaç "üretim kalitesinde" Python uygulaması (implementation) daha vardır...

Java sanal makinesi (Java virtual machine - JVM) için Java ile yazılmış Jython, RPython (Restricted Python) ile yazılmış ve C'ye çevrilmiş PyPy ve Ortak Dil Altyapısı (Common Language Infrastructure) için C# ile yazılmış IronPython.

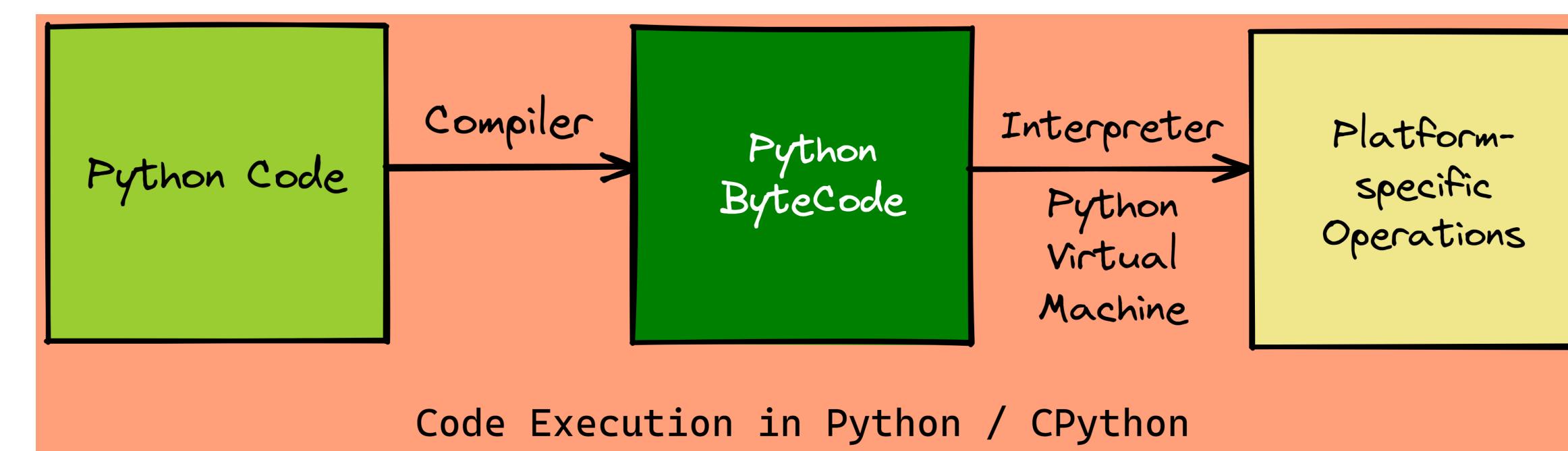
Ayrıca birkaç deneysel uygulama daha vardır. CPython ile ilgilendiğimizden C dilinin paradigmاسına bakmak iyi bir fikirdir. <https://lnkd.in/dFDXSpUD>

Python

CPython, Python'un orijinal uygulamasıdır. Python.org'dan indirilen Python sürümü bu uygulamadır. Hem C hem de Python dili ile yazılmıştır.

CPython, hem **derleyici (compiler)** hem de **yorumlayıcı (interpreter)** olarak adlandırılabilir çünkü yazdığımız Python kodu ilk olarak (gizli bir şekilde) Python bayt koduna (bytecode) derlenir, ardından bu bayt kodu, Python Sanal Makinesi (Python Virtual Machine - PVM) tarafından platforma-özgü operasyonlara dönüştürülerek yorumlanır.

Python dili (`python-the-language`) ile Python kodunu çalıştıran (yani “run” eden) şey birbirinden farklıdır!



Python



Emirli Programlama
(Imperative Programming)

Prosedürel Programlama
(Procedural Programming)

Fonksiyonel Programlama
(Functional Programming)

Nesne-tabanlı Programlama
(Object-oriented Programming)

C, başlangıçta Dennis Ritchie tarafından 1969 ve 1973 yılları arasında AT&T Bell Laboratuarlarında geliştirilen genel amaçlı bir dildir.

C, belirli bir sorunu çözmek için izlenmesi gereken kesin prosedürü adım adım açıklayan **prosedürel paradigmayı** takip eder.

Python



Emirli Programlama
(Imperative Programming)

Prosedürel Programlama
(Procedural Programming)

Fonksiyonel Programlama
(Functional Programming)

Nesne-tabanlı Programlama
(Object-oriented Programming)

Python, ayrıca bir prosedürel programlama dili olarak da kullanılabilir; bu durumda, görevleri (tasks) yerine getirmek için prosedürler (procedures) veya fonksiyonlar (functions) yazarsınız ve programın kontrol akışı (control flow), büyük ölçüde bu prosedürlerin **sıralı** bir şekilde çağrılmalarıyla tanımlanır.

Prosedürel programlama, her fonksiyonun belirli bir görevi yerine getirdiği ve kontrol akışının fonksiyonlar aracılığıyla açık bir şekilde yönetildiği bir yaklaşımı vurgular.

Python

```
...  
  
# Function to orchestrate the process of baking a cake  
def bake_cake():  
    # Step 1: Mix the ingredients  
    mix_ingredients()  
    # Step 2: Preheat the oven  
    preheat_oven()  
    # Step 3: Bake the cake  
    bake()  
    # Step 4: Decorate the cake  
    decorate()  
  
# Function to mix ingredients  
def mix_ingredients():  
    print("Mixing ingredients")  
  
# Function to preheat the oven  
def preheat_oven():  
    print("Preheating oven")  
  
# Function to bake the cake  
def bake():  
    print("Baking the cake")  
  
# Function to decorate the cake  
def decorate():  
    print("Decorating the cake")  
  
# Call the main function to bake the cake  
bake_cake()
```

Prosedürel diller, durumu (*state*) değişkenler (*variables*) aracılığıyla takip etme eğilimindedir ve genellikle bir sıra ile adımları yürütülürler.

Diğer bir deyiş ile bu yaklaşım, bir görevi yerine getirmek için adım adım talimatları sıra ile çalıştırır.

Python



Emirli Programlama
(Imperative Programming)

Prosedürel Programlama
(Procedural Programming)

Fonksiyonel Programlama
(Functional Programming)

Nesne-tabanlı Programlama
(Object-oriented Programming)

Python aynı zamanda fonksiyel programlamayı da destekler.

Python



Emirli Programlama
(Imperative Programming)

Prosedürel Programlama
(Procedural Programming)

Fonksiyonel Programlama
(Functional Programming)

Nesne-tabanlı Programlama
(Object-oriented Programming)

Fonksiyonel programlamada, belirli bir işi çok iyi yapan küçük ve yeniden kullanılabilir kod parçaları (diğer bir deyiş ile fonksiyonlar) oluşturursunuz ve verileri doğrudan değiştirmekten kaçınırsınız.

Böylelikle daha net ve daha açık, tekrar kullanılabilir (reproducible) kod yazılır.

Python

```
...  
  
# Function to orchestrate the process of baking a cake  
def bake_cake():  
    # Step 1: Mix the ingredients  
    mix_ingredients()  
    # Step 2: Preheat the oven  
    preheat_oven()  
    # Step 3: Bake the cake  
    bake()  
    # Step 4: Decorate the cake  
    decorate()  
  
# Function to mix ingredients  
def mix_ingredients():  
    print("Mixing ingredients")  
  
# Function to preheat the oven  
def preheat_oven():  
    print("Preheating oven")  
  
# Function to bake the cake  
def bake():  
    print("Baking the cake")  
  
# Function to decorate the cake  
def decorate():  
    print("Decorating the cake")  
  
# Call the main function to bake the cake  
bake_cake()
```

Örneğin, yan taraftaki kod parçasında, her bir fonksiyon tekrar tekrar kullanılarak çok çeşitli kek'in pişirilmesini sağlayabilir.

Python



Emirli Programlama
(Imperative Programming)

Prosedürel Programlama
(Procedural Programming)

Fonksiyonel Programlama
(Functional Programming)

Nesne-tabanlı Programlama
(Object-oriented Programming)

Python, Nesne Tabanlı Programlama paradigmalarının da bir parçasıdır.

Python

Nesne Tabanlı Programlama'yı bir inşaatın proje planı (blueprint) olarak düşünebilirsiniz.



Python

Nesne tabanlı bir programlama (OOP) dili **sınıf (class)** oluşturmayı ve bu sınıfından **nesneler (objects)** yaratmayı ve bu nesneler ile çalışmayı sağlar.

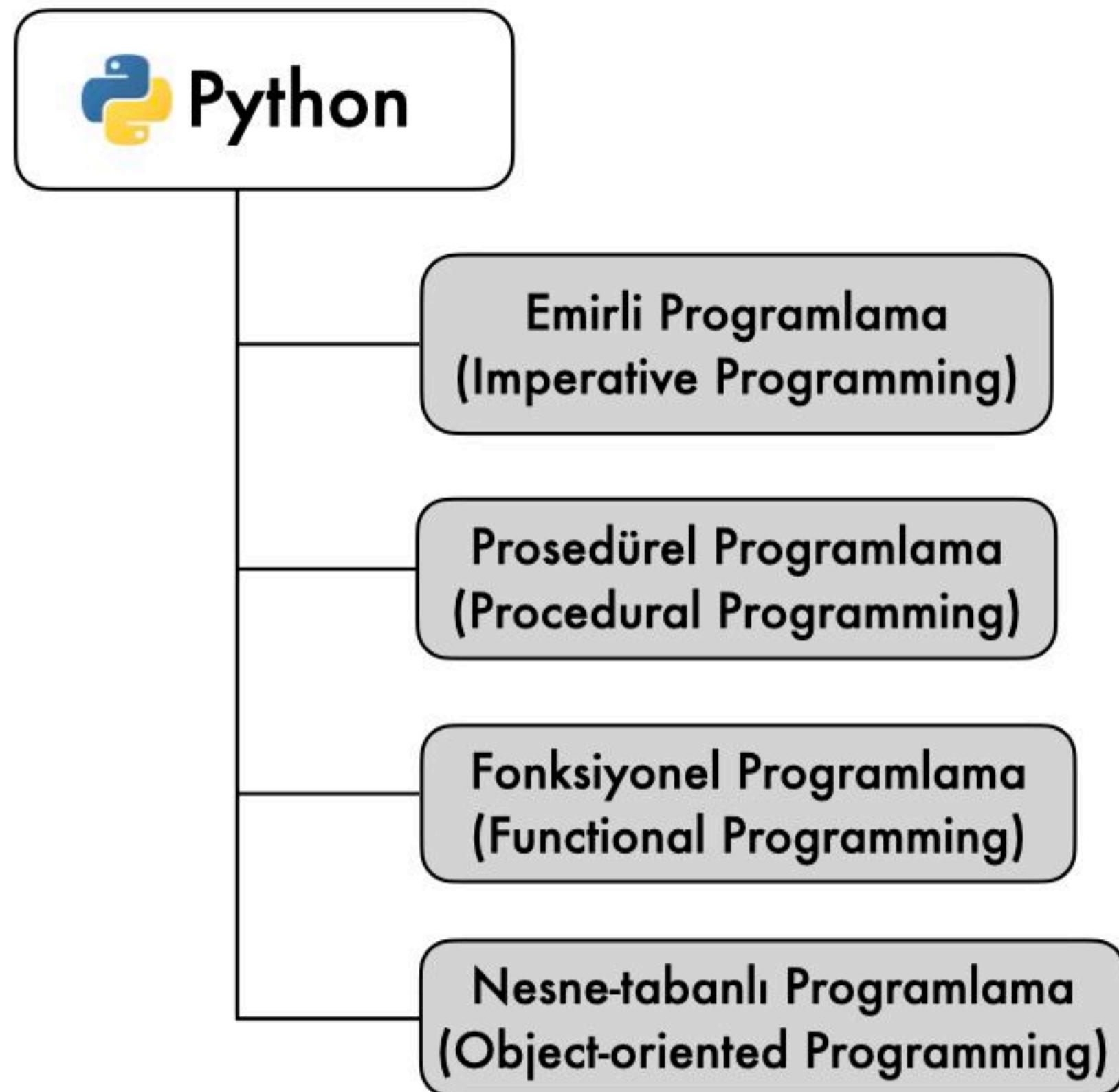
OOP'de, hem verileri (nitelikleri (attributes) - nesnelerin özellikleri denilebilir) hem de davranışları (metotlar (methods)) bir sınıf içinde kapsülleyen (encapsulation) nesnelerin tasarıımına odaklanırsınız.



Python

```
•••  
  
# Define a class called 'Car'  
class Car:  
    # Constructor method to initialize attributes  
    def __init__(self, brand, model, year, mileage):  
        self.brand = brand      # Attribute for car brand  
        self.model = model      # Attribute for car model  
        self.year = year        # Attribute for car manufacturing year  
        self.mileage = mileage  # Attribute for car mileage  
  
    # Method to describe the car  
    def describe(self):  
        return f"{self.year} {self.brand} {self.model} with {self.mileage} miles"  
  
    # Method to drive the car, which increases mileage  
    def drive(self, distance):  
        self.mileage += distance  
        print(f"You've driven {distance} miles. The car's new mileage is {self.mileage} miles.")  
  
# Create an instance (object) of the Car class  
my_car = Car("Toyota", "Corolla", 2020, 15000)  
  
# Accessing attributes and methods  
print(my_car.describe()) # Call method to describe the car  
# 2020 Toyota Corolla with 15000 miles  
  
my_car.drive(500)         # Call method to simulate driving the car  
# You've driven 500 miles. The car's new mileage is 15500 miles.  
  
# Create another instance of the Car class  
another_car = Car("Honda", "Civic", 2018, 30000)  
print(another_car.describe()) # Describe another car  
# 2018 Honda Civic with 30000 miles
```

Python



Farklı programlama paradigmaları türleri arasında bir karşılaştırma olmadığına dikkat etmek önemlidir. Bir yazılım (software) bilgiyi temsil etmekten başka bir şey olmadığı için, "**Problemimi temsil etmenin en iyi yolu nedir?**" sorusunun cevabı belirli bir programlama paradigmاسını seçmektir. Ancak, bu paradigmalar birbirini dışlamak zorunda değildir. Python'a bakarsanız, fonksiyonları (functions) ve sınıfları (classes) destekler, ancak aynı zamanda fonksiyonlar dahil HER ŞEY bir nesnedir (object). Fonksiyonel / Nesne-tabanlı Programlama / Prosedürel stili tek bir kod parçasında karıştırabilir ve eşleştirebilirsiniz.