GO
PROGRAMLAMA DİLİ

GÖRKEM BERBEROĞLU

İçerik

- Programlama Dili Nedir ?
- Go Programlama Dili
- Söz Dizimi ve Veri Tipleri
- Örnekler

Programlama Dili Nedir ?

Programlama Dili Nedir ?

Programlama dili, belli bir standart formunda komutlar ve yazılımlar geliştirme imkanı sunan bilgisayar dilidir. Programlama dilleri sayesinde bir bilgisayarın hangi durumda ne çeşit çıktı verebileceği kontrol edilebilir.

Kısacası programlama dilleri sayesinde bilgisayarlar ve insanlar verimli bir iletişim sağlayabilirler.

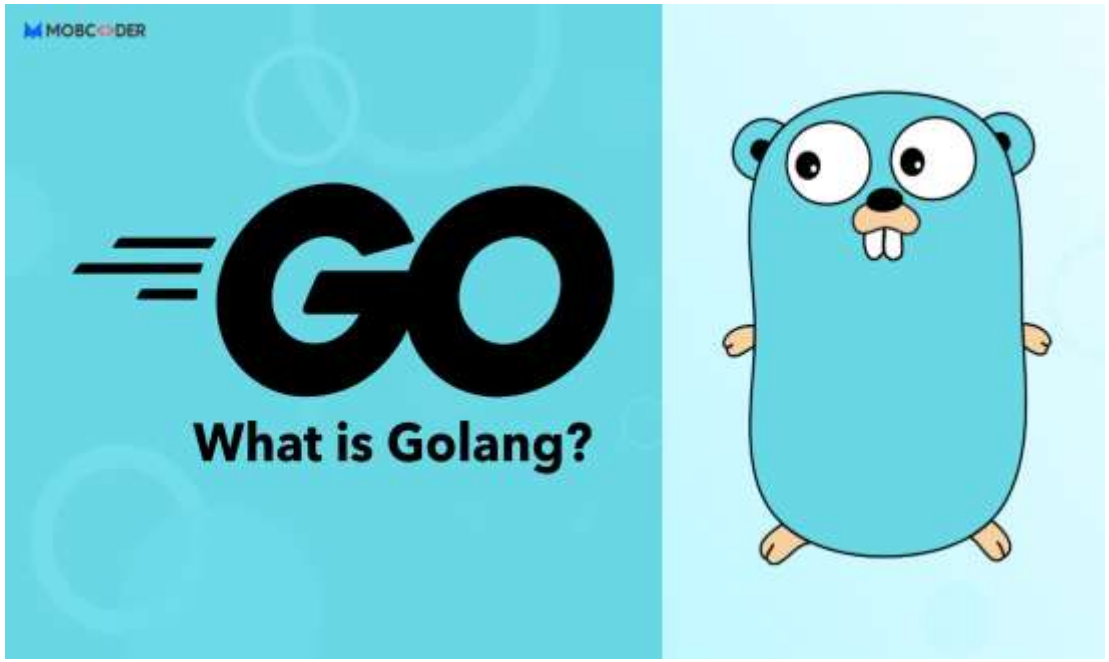
Programlama Dili Nedir ?

Bir program oluşturmak istiyorsak önce makinelerle anlaşabilmeli ve ona hangi durumda neleri yapması gerektiğini söylemeliyiz. Programlama dilleri ve algoritmalar bu iletişimi sürdürebilmemizin en sağlıklı yoludur.

Popüler Programlama Dilleri

Popüler Programlama Dilleri

Go Programlama Dili



Go Programlama Dili

- Go, 2007 yılında Google'ın geliştirmeye başladığı açık kaynak programlama dilidir.
- Go, Google mühendisleri tarafından ortaya çıkarılmıştır ve diğer dillerin bilinen eleştirilerini çözecek biçimde tasarlanmıştır.
- Kullanılabilir ilk sürüm 10 Kasım 2009'da, ilk kararlı sürüm 1.0 versiyonu ise 28 Mart 2012'de Go 1.0 olarak yayınlanmıştır.

Go Programlama Dili

- Google kendi iç sistemlerinde yıllardır kullandığı C, C++, Java, Python vb. bir çok farklı programlama dili ve teknolojinin farklı farklı avantaj ve dezavantajları vardır.
- Bunlar performans, derleme, güvenlik, uyumluluk, zaman yönetimi, kaynak*(donanım, para, enerji vb.)* yönetimi gibi bir çok başlıkta sıralanabilir.
- Google bunlar gibi birçok sorunu çözmek için Go Programlama dilini geliştirmeye başlamıştır**. Amaç, az dil özelliği ve kuralıyla esnek, hızlı, performanslı ve güçlü bir dil oluşturmaktır.**

Go Programlama Dili Özellikleri

- Go tek bir dosyaya derlenir.
- Go dilinde söz dizimi anlaşılır ve basittir.
- Go uygulamanızın içerisinde C, Python gibi farklı dilleri doğrudan kodunuza yazarak çalıştırabilirsiniz.
- Go dili kendi içinde gömülü olarak concurrency(eş zamanlılık) destekler ve bunu yüksek performanslı olarak gerçekleştirir.
- Go dili söz dizimi kurallarına çok önem verir ve dikkat eder!

Go Programlama Dili Hangi Amaçla Kullanılabilir ?

- Go dili sistem programlama odağıyla geliştirilmiştir. Yani sunucu ve alt sistemler yazmak için ideal bir çözüm olarak düşünülebilir.
- Web için hem hızlı geliştirme hem de yüksek performansıyla kaliteli projeler üretmenizi sağlayabilir.
- Go dili gömülü sistemler için de kullanılabilmektedir. Ancak genel olarak bu alan için henüz yeterli görülmemektedir.
- Henüz tam uyumlu kullanılabilir olmasa da Google, Android işletim sistemi üzerinde Go kullanılabilmesi için bir mobil proje yürütmektedir.

Go Programlama Dili Kullanıcıları

- Go dili yakın bir tarihte geliştirilmiş olsa da büyük projeler için ciddi tercih edilen bir teknoloji haline gelmesi uzun sürmedi. Bu girişim ve firmalardan bazıları şunlardır;

- Docker
- Koding
- Google
- Apple
- Twitter
- Amazon
- Ubuntu
- Dropbox
- Facebook
- Vb.

- https://github.com/golang/go/wiki/GoUsers

Go Programlama Dili Kurulumu ve Tanıtımı

Go dilinin resmi web sitesi golang.org‘dur.

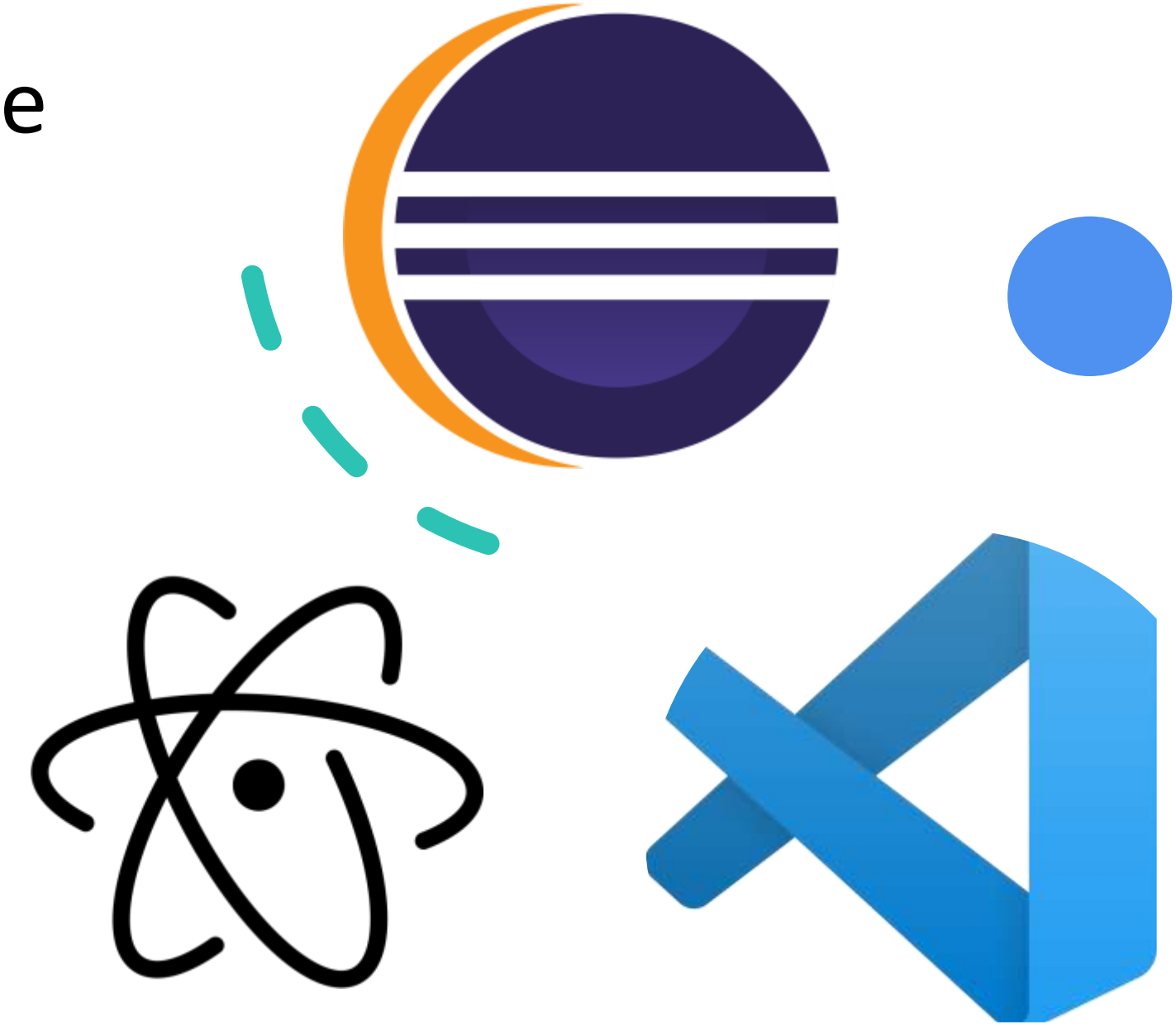
- Go’yu bilgisayarıma nasıl indiririm? https://golang.org/dl/
- Go dökümantasyonuna erişim; https://golang.org/doc/
- Go paketlerine online erişim; https://golang.org/pkg/
- Online Go IDE; https://play.golang.org/

Go Programlama Dili Geliştirme Araçları

- **Go tool:** Go dilinin en temel aracıdır. Go uygulamanızın derleme esnasındaki işlemleri konsol üzerinden parametrik olarak yönetmenizi sağlar.
- **Godoc:** Kod içerisinde belgelendirmeyi sağlar, örneğin bir metod üzerinde yazdığınız açıklama satırına göre bir dökümantasyon üretir.
- **Gofmt:** Go dili söz dizimi kurallarını(bazıları katıdır) yöneten bir araçtır. Go kodu söz dizimi standartlarını uygular.

Go Dili Geliştirme Ortamları

- Visual Studio Code
- Eclipse
- Atom
- Vim
- Emacs
- SublimeText
- IntelliJ
- LiteIDE



Söz Dizimi ve Veri Tipleri

Keyword ve Operatörler

- Go Programlama dilinde toplam **25 adet keyword** vardır. Diğer programlama dilleri ile karşılaştırıldığında en az keyword'e sahip dillerden birisidir.
- Bazı temel diller için keyword sayıları aşağıdaki gibidir.

- C++: 82
- Java: 50
- JavaScript: 38
- Python(3.7): 35

Keyword ve Operatörler

- Go dilindeki keyword listesi aşağıdaki gibidir:

```
break    default      func   interface select
case     defer        go     map       struct
chan     else         goto   package   switch
const    fallthrough  if     range     type
continue for          import
```

- Operatör ve sınırlayıcılar ise 47 adettir.

```
+    &    +=   &=    &&   ==   !=   (   )
-    |    -=   |=    ||   <    <=   [   ]
*    ^    *=   ^=    <-   >    >=   {   }
/    <<   /=   <<=   ++   =    :=   ,   ;
%    >>   %=   >>=    —   !     …   .   :
&^   &^=
```

Değişken Tanımlama

- Bir değer ataması yapmadan tanımlama: Değişken, tipine göre varsayılan değerine sahip olur.

```
var t int
```

- = operatörü ile değer ataması yaparak tanımlama:

```
var t int = 10
```

- *Kısa değişken tanımlaması* olarak adlandırabileceğimiz bir yöntem daha vardır. Bu tanımlamada := operatörü kullanılır ve **değişken tanımlanırken bir değer atanması zorunludur**.

```
t := 10
```

Değişken İsimlendirme

- Go programlama dilinde diğer birçok dilde olan **private, public, protected** gibi erişim belirleyici keyword'ler **yoktur**. Bu durumu da daha sade bir yol ile çözmüşlerdir.

- Eğer değişken ismi büyük harf ile başlıyor ise bu değişken tanımlı olduğu paket dışından da erişilebilir oluyor. Ancak küçük harf ile başlıyorsa sadece tanımlı olduğu paket içerisinde erişilebilir olur.

```
var FirstName string
var password string
```

Değişken İsimlendirme Kuralları

- ❑ Sadece harf, numara ve alt çizgi kullanılarak yapılmalıdır.

```
Doğru -> user_name, userName, signal4
Yanlış -> best.offer, user-name
```

- ❑ Bir numara ile başlamamalıdır

```
Doğru -> best4Game, house62
Yanlış -> 10cars
```

- ❑ Sembol kullanılmamalıdır

```
Doğru -> product
Yanlış -> $product
```

Çoklu Atama

- Go ile birden fazla değişkene tek satırda farklı data tipinde değer ataması yapılabilmektedir. Özellikle kod okunurluğu açısından çok kolaylık sağlayan bir özelliktir.

```
a,b,c := 12, true, "Google"

fmt.Println(a)
fmt.Println(c)
fmt.Println(b)
```

Çıktısı:

```
12
Google
true
```

Konsol Çıktısı Oluşturma

- Konsol çıktıları oluşturabilmek için kaynak kod içerisinde **fmt** paketinden yararlanabiliriz. **fmt** paketini import ettikten sonra paket içerisindeki **Print, Printf ve Println** fonksiyonları ile çıktılar oluşturabiliriz.

- Eğer standart bir string ifadeyi çıktı olarak kullanacaksak Println fonksiyonunu kullanabiliriz.

- Eğer çıktımıza bazı değerleri parametre olarak geçeceksek ve/veya bu parametre olarak geçilen değerleri formatlayarak göstereceksek Printf kullanmalıyız.

Yorum Satırları

- Go dilinde yorum satırları diğer birçok dil ile aynı yöntemler kullanılarak oluşturulmaktadır. Bu yöntemlerden en basiti iki adet // (slash) karakterini satırın başına eklemektir.

```
// Yorum satırı…
```

- Eğer bir satır bloğunu yorum satırı haline getirmek istiyorsak da bloğun en başına /* ,en sonuna da */ karakterlerini ekleyerek içeride kalan satırları yorum satırı haline getirmiş oluruz.

```
/*
 Bu satırların
 hepsi
 aslında yorum satırıdır
*/
```

Diziler

- Diziler içlerinde bir veya birden fazla değer tutabilen birimlerdir.

- Bir dizideki her değer sırasıyla numaralandırılır. Numaralandırma sıfırdan başlar.

```go
package main
import "fmt"
func main() {
  var a [3]string
  a[0] = "Ayşe" //Birinci değer
  a[1] = "Fatma" //İkinci değer
  a[2] = "Hayriye" //Üçüncü değer
  fmt.Println(a) //Çıktımız: [Ayşe Fatma Hayriye]
  fmt.Println(a[1])//Çıktımız: Fatma
}
```

Döngüler

- Programlama ile uğraşan birçok insanın bileceği üzere , programlama dillerinde while, do while ve for döngüleri vardır. Bu döngüler ile yapacağımız işlemin belirli koşullarda tekrarlanmasını sağlayabiliriz.
- Go dilinde ise diğer dillerin aksine sadece **for** döngüsü vardır.
- Ama bu while ve do while ile yapılanları yapamayacağımız anlamına gelmiyor. Go dilinde for döngüsü ile hepsini yapabiliriz.
- Yani dilin yapımcıları tek döngü komutu ile hepsini yapabilmemize olanak sağlamışlar.

Döngüler

- Go’da for döngüsü parametreleri parantez içine alınmaz.

```go
package main

import "fmt"

func main() {
        for i := 0; i < 10; i++ {
                fmt.Println(i)
        }
}
```

```go
package main

import "fmt"

func main() {
        deger := 0
        for deger < 10 {
                fmt.Println(deger)
                deger++
        }
}
```

If-Else-Else if

- **If-Else** akışı koşullandırmalar için kullanılır. Diğer dillerin aksine koşul parametresi parantezler içine yazılmaz.
- **If-Else** akışında birden fazla koşul kontrolü ekleyebiliriz. Bunu **else if** deyimi ile yapabiliriz.

```
1 i := 5
2 if i == 5 {
3   fmt.Println("i'nin değeri 5'tir.")
4 } else if i==3{
5   fmt.Println("i'nin değeri 3'tür.")
6 }else{
7   fmt.Println("i'nin değeri belirsiz.")
8 }
```

Fonksiyon Tanımlama

- Fonksiyonlarımızı oluşturmak için **func** anahtar kelimesini kullanırız. Yanına ise fonksiyonumuzun ismini yazarız. Parantez içine fonksiyonumuzun dışarıdan alacağı parametreler için değişken-tip tanımlaması yaparız. Parantezin sağına ise fonksiyonun döndüreceği return değerinin tipini yazarız.
- Süslü parantezler içinde fonksiyonumuzun işlemleri bulunur. Son olarak return ile veri tipini belirlediğimiz değeri elde etmiş oluruz.

```go
package main

import "fmt"

func topla(a int, b int) int {
        return a + b //a ve b'nin toplamını döndürür.
}

func main() {
        fmt.Println(topla(2, 5)) //2+5 sonucunu ekrana bastır
}
```

Örnekler

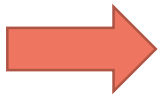
Örnek-1

```go
package main
import "fmt"

func main() {

    i := 1
    for i <= 3 {
        fmt.Println(i)
        i = i + 1
    }
    for j := 7; j <= 9; j++ {
        fmt.Println(j)
    }
    for {
        fmt.Println("for dongusu")
        break
    }
    for n := 0; n <= 5; n++ {
        if n%2 == 0 {
            continue
        }
        fmt.Println(n)
    }
}

```

```
1
2
3
7
8
9
for dongusu
1
3
5
```

Örnek-2

```go
package main
import "fmt"

func main() {

    var dizi1 [5]int
    fmt.Println("Dizi1 Elemanlari:", dizi1)

    dizi1[4] = 100
    fmt.Println("Dizi1in 5.elemani 100 yapildi:", dizi1)
    fmt.Println("Dizi1in 5.elemani:", dizi1[4])

    fmt.Println("Dizi1in Uzunlugu:", len(dizi1))

    dizi2 := [5]int{1, 2, 3, 4, 5}
    fmt.Println("Dizi2nin Elemanlari:", dizi2)

    var matris [2][3]int
    for i := 0; i < 2; i++ {
        for j := 0; j < 3; j++ {
            matris[i][j] = i + j
        }
    }
    fmt.Println("for dongusu ile olusan matris: ", matris)
}
```



```
Dizi1 Elemanlari: [0 0 0 0 0]
Dizi1in 5.elemani 100 yapildi: [0 0 0 0 100]
Dizi1in 5.elemani: 100
Dizi1in Uzunlugu: 5
Dizi2nin Elemanlari: [1 2 3 4 5]
for dongusu ile olusan matris:  [[0 1 2] [1 2 3]]
```

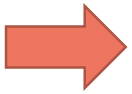
Örnek-3

```go
func main() {
        a := 8
        fmt.Println(&a) //Çıktımız: 0xc0000b8010
        b := &a
        fmt.Println(b) //Çıktımız: 0xc0000b8010
        fmt.Println(*b) //Çıktımız: 8
}
```

```go
package main

import "fmt"

func main() {
        a := 8
        b := &a
        *b = 10
        fmt.Println(a) //10
}
```

Örnek-4

```go
package main
import "fmt"

func guncelleme(sayi *int,metin *string) {
    *sayi = *sayi + 5
    *metin = *metin + " Teknik Üniversitesi"
    return
}

func main() {
    var number = 20
    var text = "Bursa"
    fmt.Println("ilk hali:",text,number)

    guncelleme(&number,&text)

    fmt.Println("Guncelleme sonrasi :",text,number)
}

```



```
ilk hali: Bursa 20
Guncelleme sonrasi : Bursa Teknik Üniversitesi 25
```

Teşekkürler

Görkem BERBEROĞLU