

ERMAT | **Binbaşı Reşatbey, Akçay Cd. no 25/1, 35410 Gaziemir/İzmir**

GOLANG EĞİTİM

GOLANG EĞİTİM KAYNAĞI

Deniz Korkmazer&ferhat korkmaz

2022

İçindekiler Tablosu

[golang eğitim 8](#_Toc119918888)

[başlangıç 8](#_Toc119918889)

[GO TEMEL(DEĞER) VERİ TİPLERİ 8](#_Toc119918890)

[TEMEL(DEĞER) VERİ TİPLERİ 8](#_Toc119918891)

[Golang komutları 8](#_Toc119918892)

[1-STRING 9](#_Toc119918893)

[2-INTEGER 10](#_Toc119918894)

[3-FLOATING-POINT 10](#_Toc119918895)

[4-BOOLEAN 10](#_Toc119918896)

[5-COMPLEX 11](#_Toc119918897)

[6-BYTE 11](#_Toc119918898)

[Hello World 12](#_Toc119918899)

[Yorum satırı 12](#_Toc119918900)

[Tam Sayı Tipleri 13](#_Toc119918901)

[String Tipler 14](#_Toc119918902)

[Boolean Tipler 14](#_Toc119918903)

[Operatörler 15](#_Toc119918904)

[ARİTMETİK OPERATÖRLER 15](#_Toc119918905)

[Aritmetik Operatörler Örneği 15](#_Toc119918906)

[ATAMA OPERATÖRLERİ 16](#_Toc119918908)

[Atama Operatörleri Örneği 16](#_Toc119918909)

[KARŞILAŞTIRMA OPERATÖRLERİ 17](#_Toc119918910)

[Karşılaştırma Operatörleri Örneği 17](#_Toc119918911)

[MANTIKSAL OPERATÖRLER 18](#_Toc119918912)

[Mantıksal Operatörler Örneği 18](#_Toc119918913)

[BİTSEL OPERATÖRLER 18](#_Toc119918914)

[Bitsel Operatörler Örneği 19](#_Toc119918915)

[Boş tanımlayıcılar 19](#_Toc119918916)

[Type casting 20](#_Toc119918917)

[Konsol çıkış işlemleri print 21](#_Toc119918918)

[Formatlama 21](#_Toc119918919)

[Konsol giriş işlemleri 22](#_Toc119918920)

[VKI ÖRNEK (vÜCUT kİTLE eNDEKSİ) 23](#_Toc119918921)

[Anonim foksiyonlar 23](#_Toc119918922)

[kök bulma 24](#_Toc119918923)

[konsol giriş işlemleri scanf 25](#_Toc119918925)

[koşullu durumlar 25](#_Toc119918926)

[If-Else 25](#_Toc119918927)

[ELSE-IF KULLANIMI 26](#_Toc119918928)

[**Koşullar İçerisinde Operatör Kullanımı** 26](#_Toc119918929)

[IF ELSE ile Not Uygulaması 27](#_Toc119918930)

[IF ELSE ile hesap makinesi 28](#_Toc119918932)

[Switch(CaSe) 29](#_Toc119918933)

[Koşullu Switch 30](#_Toc119918934)

[Koşulsuz Switch 30](#_Toc119918935)

[Sonraki Koşulu Kontrol Ettirme 31](#_Toc119918936)

[Switch'e Özel Değişken Tanımlama 31](#_Toc119918937)

[Switch case hesap makinesi 32](#_Toc119918938)

[Döngüler 33](#_Toc119918939)

[STANDART FOR KULLANIMI 33](#_Toc119918940)

[Çift Sayı Bulma Döngüsü 34](#_Toc119918941)

[İç içe döngüler 34](#_Toc119918942)

[çarpım tablosu 34](#_Toc119918943)

[Break continue 35](#_Toc119918944)

[sonsuz döngüler/forever 35](#_Toc119918945)

[For uygulaması kullanıcı girişi 36](#_Toc119918946)

[fizzbuzz 37](#_Toc119918947)

[range for yapısı 37](#_Toc119918948)

[etiket(label)kullanımı ve goto komutu 37](#_Toc119918949)

[for uygulaması hesap makinesi 38](#_Toc119918950)

[FOR UYGULAMASI FAKTORİYEL BULMA 39](#_Toc119918951)

[FOR UYGULAMASI MÜKEMMEL SAYI 39](#_Toc119918952)

[diziler(arrays) 40](#_Toc119918953)

[çok boyutlu diziler 42](#_Toc119918957)

[slices 42](#_Toc119918958)

[SLİCE APPEND MAKE METOTLARI 44](#_Toc119918959)

[Range for yapısı 45](#_Toc119918960)

[maps yapısı 45](#_Toc119918961)

[array 2.örnek (alıştırma) 47](#_Toc119918963)

[slice örnek(alışırma) 47](#_Toc119918964)

[Map örnek(alıştırma) 48](#_Toc119918965)

[hata yönetimi 48](#_Toc119918966)

[FONKSİYONLAR(FUNCTİONS) 49](#_Toc119918968)

[Return Fonksiyonlar 49](#_Toc119918969)

[MultipleReturn Fonksiyonlar 50](#_Toc119918970)

[NAMED RETURN Fonksiyonlar 51](#_Toc119918971)

[variadic Fonksiyonlar 52](#_Toc119918972)

[Recursive (İç-içe) Fonksiyonlar 52](#_Toc119918973)

[anonim fonksiyonlar 53](#_Toc119918974)

[defer 54](#_Toc119918975)

[fibonacci sayı serisi 55](#_Toc119918976)

[STRUCT 56](#_Toc119918977)

[ERİŞİLEBİLEN STRUCTLAR 58](#_Toc119918978)

[İÇ İÇE STRUCT 59](#_Toc119918979)

[ANONİM STRUCT 59](#_Toc119918980)

[METHOD(METOTLAR) 60](#_Toc119918981)

[KENDİ VERİ TİPİMİZ 61](#_Toc119918982)

[interface 62](#_Toc119918983)

[POİNTER(İŞARETÇİLER) 64](#_Toc119918984)

[Declaring pointers 65](#_Toc119918985)

[Bir işleve işaretçi geçirme 66](#_Toc119918987)

[POİNTERLARIN FONKSİYONLAR İLE KULLANIMI 67](#_Toc119918988)

[POİNTER İLE STRUCT’LAR ve metotlar ile kullanımı 67](#_Toc119918989)

[PACKAGE(PACKAGES) 69](#_Toc119918990)

[TİME PAKETİ 69](#_Toc119918991)

[MATH PAKETİ 70](#_Toc119918992)

[STRİNGS PAKETİ 71](#_Toc119918993)

[GİTHUB PAKET 72](#_Toc119918994)

[DOSYA İŞLEMLERİ 72](#_Toc119918995)

[dosya açmak ve içine veri yazdırmak 73](#_Toc119918996)

[DOSYA TAŞINMAK VE İSİM YENİLENDİRMEK 74](#_Toc119918997)

[DOSYA KOPYALAMAK 74](#_Toc119918998)

[DOSYA SİLME 75](#_Toc119918999)

[DOSYA BİLGİLERİ 75](#_Toc119919000)

[DOSYA KONTROL 75](#_Toc119919001)

[DOSYA OKUMA İŞLEMİ 76](#_Toc119919002)

[KLASÖR OLUŞTURMA 76](#_Toc119919003)

[GEÇİCİ KLASÖR VE GEÇİCİ DOSYA 77](#_Toc119919004)

[concurrency 77](#_Toc119919005)

[GOROUTİNE 78](#_Toc119919006)

[TERİMLER 78](#_Toc119919007)

[WAİTGROUP 79](#_Toc119919008)

[KANALLAR(CHANNELS) 80](#_Toc119919009)

[Boyutlu kanal oluşturma 82](#_Toc119919010)

[DEADLOCK 82](#_Toc119919011)

[VERİTABANI(DATABASE) 83](#_Toc119919012)

[database SQL TABLO OLUŞTURMA, VERİ EKLEME 85](#_Toc119919013)

[SQL SORGULARI VERİ ÇEKME 86](#_Toc119919015)

[VERİTABANI BAĞLANTISI VE TABLO OLUŞTURMA 86](#_Toc119919016)

[dbsql VERİ çekme 87](#_Toc119919017)

[dbsql veri güncelleme 88](#_Toc119919018)

[dbsql array ile veri ekleme 88](#_Toc119919019)

[dbsql VERİ SİLME 89](#_Toc119919020)

[dbsql transactions 89](#_Toc119919021)

[veri ekleme 90](#_Toc119919022)

[XML,JSON 90](#_Toc119919023)

[XML NEDİR? (EXTENSİBLE MARKUP LANGUAGE) 90](#_Toc119919024)

[JSON NEDİR? (JAVASCRİPT OBJECT NOTATİON) 91](#_Toc119919025)

[XML DOSYASI OLUŞTURMA 91](#_Toc119919026)

[GO PROGRAMLAMA İLE XML DOSYASI OLUŞTURMA 93](#_Toc119919027)

[json dosyası oluşturma 94](#_Toc119919028)

[JSON İŞLEMLERİ 95](#_Toc119919029)

[GO PROGRAMLAMA İLE JSON DOSYASI OLUŞTURMA 96](#_Toc119919030)

[basit mail gönderme işlemi 97](#_Toc119919031)

[HTML YAPILI MAİL GÖNDERME 98](#_Toc119919032)

[GO MAİL PACKAGE (gopkg.in/gomail.v2) 98](#_Toc119919033)

[“MERHABA DÜNYA” İLE WEB UYGULAMASI YAPMAK 99](#_Toc119919034)

[WEB FUNCTİONS AND HANDLERS 99](#_Toc119919035)

[HATA KONTROLÜ 100](#_Toc119919036)

[HTML ŞABLONLAR(TEMPLATES) 101](#_Toc119919037)

[101](#_Toc119919039)

[html kodu gönderme 102](#_Toc119919040)

[ŞABLONA STRUCT GÖNDERME 102](#_Toc119919041)

[ŞABLON İÇİNDEN DEĞİŞKEN ATAMA 103](#_Toc119919042)

[ŞABLONDA İF-ELSE KULLANIMI 103](#_Toc119919043)

[ŞABLONDA RANGE DÖNGÜSÜ KULLANIMI 104](#_Toc119919044)

[ŞABLON İÇERSİNDE YORUM SATIRI OLUŞTURMA 104](#_Toc119919045)

[NET/http İLE WEB SERVER OLUŞTURMA 104](#_Toc119919046)

[http İSTEKLERİ(REQUESTS) 105](#_Toc119919047)

[GET İSTEĞİ 105](#_Toc119919048)

[SORGU PARAMETRESİ EKLEME 106](#_Toc119919049)

[POST İSTEĞİ 107](#_Toc119919050)

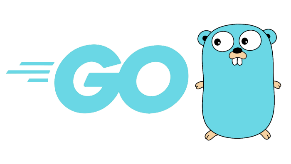
[WaitGroup ile Asenkron İşlemleri Beklemek 109](#_Toc119919051)

[Mutex ile Asenkron İşlem Sırası 110](#_Toc119919052)

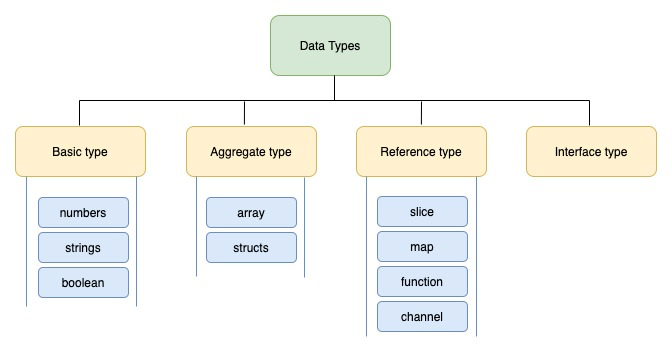
[select 111](#_Toc119919053)

# golang eğitim

### başlangıç deniz korkmazer



# GO TEMEL(DEĞER) VERİ TİPLERİ



## TEMEL(DEĞER) VERİ TİPLERİ

Değer tipleri veriyi tutan değişken türleridir.  
Referans türleri ise veri yerine verinin bellekteki adresini tutan değişken türleridir Go Programlama dilinde kullanılan Temel düzeydeki yani Değer Tipindeki veri tiplerini inceleyelim.

### Golang komutları

BUİLD:Kendisine verilen kaynak kod dosyasını ve bağımlılıklarını derlemeyi sağlar.Eğer paket main paketi ise executable dosyayı pakey dosyasının bulunduğu klasörde oluşturur.

Clean:Object dosyalarını paket kaynak kod dosyasını ve bağımlılıklarını derlemeyi sağlar.

BUG: Default tarayicinin, sistem bilgilerini de içeren yeni bir bug raporuile acilmasini saglar.

. DOC: Belirtilen pakette tanimlanan doktimantasyon yorumlarini yazdirmayı saglar.

• ENV: Calisilan bilgisayardaki go kurulumuna ait çevresel degiskenlerle

ilgili bilgilerin yazdırımlasını sağlar.

FİX: Belirtilen pakette düzeltme islemlerini gerçeklestirir.

• FİT: Belirtilen paketteki kodlari go standartlarina göre formatlamayı saglar.

• Generate: Derleme öncesi kaynak kodu olusturmak veya güncellemek

amaci ile araçlarin calsmasii otomatiklestirmek amaci ile kullanilir.

•GET: Bir go paketi kurulurken kullanilan komuttur. go get bagimhliklart

ile birlikte belirtilen paketleri indirmeyi ve daha sonra yüklemeyi saglar.

• İNSTALL: go install belirtilen paketi derler ve olusturulan executable dos-yayı $GOBIN ile belirtilen yola, derlenen bagimlılıkları da $GOPATH/pkg dizinine yükler.

• LİST: Calsilan projedeki paketleri listeler.

• MOD: Modüllerdeki islemlere erisimi saglar.

• RUN: Belirtilen paketi derleyip çalistirmayı sağlar.

• TEST: Bulunulan dizindeki her bir paketin "\*\_test.go"

dosya ismi deseni ile örtüsen isimlerle birlikte yeniden derlenmesini sağlar.

• TOOL: Kendisine parametre olarak verilen go aracini çalistirmayı sağlar.

Eger parametre verilmezse bilinen araclarin listesini yazdirir.

•VERSİON:Çalışanların go versiyonunu öğrenmeyi sağlar.

•VET:Kaynak kodları inceler ve şüpheli yapıları raporlar.

### 1-STRING

String'ler byte dizinidir.Karakter (Harf) dizinlerini yani kelimeleri,cümleleri tanımlarken kullandığımız veri tipidir.Go'da string şeklinde yazılır. Referans tipteki bir veri tipi olmasına karşın temel düzeyde sıkça kullandıldığı için bu yazıda paylaşmak istedim.

*var mesaj string="Merhaba Dünya"*

GO'da stringlere değer ataması yapılırken "" (çift tırnak) işareti kullanılır.

### 2-INTEGER

Integer, tamsayıları belirtmek için kullandığımız veri tipidir. Integer'ın kısaltması int olarak tanımlanır.  
Integer veri tipleri İşaretli(signed) ve İşaretsiz(unsinged) olmak üzere ikiye ayrılır. İşaret dediğimiz **"- "**değerini almamasıdır. uint olarak işaretsiz integerlar tanımlanır. Bir değişkenin değerinin eksi olmayacağından eminseniz uint tipini kullanabilirsiniz.  
Integerler int8,int16,int32,int64 || uint8,uint16,uint32,uint64 olarak tanımlanabilir. Buradaki sayılar 28 , 216 gibi boyutunu belirtir.  
Örneğin int8 yani 256 bitlik bize -127 ve +127 arasındaki tam sayıları tutmamızı sağlar.  
Buradaki 1'lik bit fark sayının işaretini tutmak için kullanılmıştır.  
İşaretsiz integer'da ise 0-255 arasındaki değerler tutulmaktadır. Aynı mantık int16,int32 içinde geçerlidir.

*var sayi8 int8 = 127  
var sayi16 int16 = 32767  
var sayi32 int32 = 2147483647  
var sayi64 int64 = 9223372036854775807*

Go dilinde yazarken, eğer çok büyük bellek tüketimine sahip bir program yazmıyorsak,sadece **int** yazarak geçiyoruz.

### 3-FLOATING-POINT

Float tipi integer değerlerini yetersiz olduğu yani ondalıklı basamak gösterirken kullanımız bir veri tipidir.Float32 ve Float64 olmak üzere iki türü vardır.Go'da float32 veya float64şeklinde yazılır.

float32: 32bitlik float sayılar  
float64: 64bitlik float sayılar

*var piSayisi float64=3.14285714286*

Türkiye'de küsüratlı yani ondalık gösterirken **, (virgül)**kullanılırız. Ancak Programlama dillerinde "İngiliz-Amerikan" ayrıma biçimi olan **. (nokta)** kullanılır.

### 4-BOOLEAN

Boolean veri tipleri mantıksal sonuç çıkarmada kullanılır.Doğru yanlışı ifade eder. True - False değerlerini tutar.1 bit'lik veri tutar.Go'da diğer dillerden farklı olarak Boolean veri tiplerini 0-1 olarak düşünmemiz gerekiyor. boolean olarak kullanılır.

*var dogruMu boolean=false*

### 5-COMPLEX

Int ve Float kadar çok ihtiyacınız olmasa da Go complex64 ve complex128 adında iki karmaşık sayı tipini de sunuyor. İlki gerçek kısım için bir float32 kullanıp, kalanı sanal kısma ayırırken, ikincisi gerçek kısım için float64 kullanıyor.

*var karmasikSayi complex64  
d = 4 + 5i*

GO'da stringlere değer ataması yapılırken "" (çift tırnak) işareti kullanılır.

### 6-BYTE

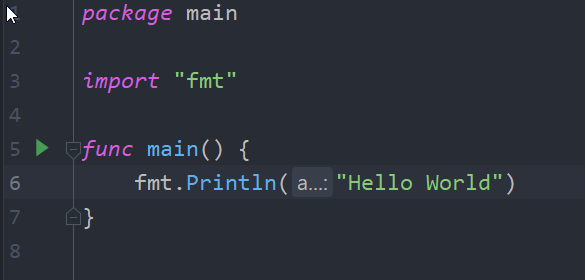
bayt veri türü, işaretsiz bir 8 bit tam sayıyı temsil etmek için kullanılır. Bir bayt veri türü varsayılan olarak sıfırdır.Go'da byte şeklinde tanımlanır.

*var yeniByte byte=12  
var asc2Kod byte='t'*

byte veri tipi,değer olarak verilen harfleri ASCII kodu karşılığındaki sayıyı da verir.



### Hello World





Bu bir “paket bildirimi” olarak bilinir. -Program derleme



import  programımızla kullanmak için diğer paketlerden gelen kodları nasıl eklediğimizdir. -Aktarma

fmt paketi (format için shorthand) giriş ve çıkış için formatlama uygular.

import fmt – fmt paketini alır. Bu paket, I/O  işlevlerini uygular.

-fmt paketinin çift tırnak işareti ile çevrili olduğuna dikkat edelim.

### Yorum satırı

Go iki farklı yorum stilini destekler: // ile satırın sonu arasındaki tüm metnin yorumun bir parçası olduğu yorumlar / \*     \*/ arasındaki her şeyin yorumun bir parçası olduğu yorumlar. (Ve birden fazla satır içerebilir)

Go/**run** ifadesi, …. .go uzantılı bir dosyaya kaydetmek ve sözdizimini kullanarak çalıştırmak anlamına gelir.



* fmt.Println – Bu, ekrandaki metni fmt’nin Println işlevi ile yazdırır.

Println adındaki fmt paketinin içindeki başka bir fonksiyona erişiriz (bu, fmt.Println parçası, Println, Print Line anlamına gelir). Daha sonra, Merhaba Dünya! içeren yeni bir dize yaratır ve ilk ve tek argüman olarak dizeyle çalışan bu dizeyi çağırır (çağrı veya yürütme olarak da bilinir).



func main() – Bu, program yürütmenin başladığı işlevdir. Ana işlev her zaman ana pakete yerleştirilmelidir. main() altına kodu {} içine yazabilirsiniz.

### **Tam Sayı Tipleri**

Go’nun tamsayı türleri: uint8, uint16, uint32, uint64, int8, int16, int32 ve int64. 8, 16, 32 ve 64 bize her bir tipin kaç bit kullandığını anlatıyor.

uint **“işaretsiz tam sayı”**,

int ise**“işaretli tam sayı”** anlamına gelir.

İşaretsiz tamsayılar yalnızca pozitif sayılar (veya sıfır) içerir.

* **int8** – 8 bit işaretli tam sayılar.
* **int16** – 16 bit işaretli tam sayılar.
* **int32** – 32 bit işaretli tam sayılar.
* **int64** – 64 bit işaretli tam sayılar.
* **uint8** – 8 bit işaretsiz tamsayılar.
* **uint16** – 16 bit işaretsiz tamsayılar.
* **uint32** – 32 bit işaretsiz tamsayılar.
* **uint64** – 64 bit işaretsiz tamsayılar.

Ayrıca iki takma ad vardır: **uint8** ile aynı olan bayt ve **int32** ile aynı olan rune.

Go iki kayan nokta tipine sahiptir. **Küsüratlı Ondalık sayıları göstermek için**

* **float32** – 32 bit kayan nokta sayıları.
* **float64** – 64 bit kayan nokta sayıları.

### String Tipler

Bir bayt sırası (karakter) gösterir. Dize bitiştirme, çıkarma alt dize vb. gibi dizelerde çeşitli işlemler yapabilirsiniz.

String Veri tipi(Harf)

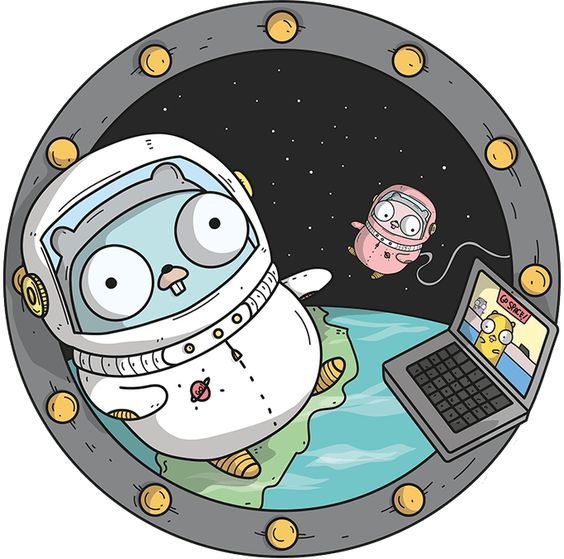
Örnek Var Mesaj string=”Buraya yazınız”

Değişken

### Boolean Tipler

Boolean değeri (George Boole’den sonra adlandırılmış), **true** ve **false** (veya açık ve kapalı) temsil etmek için kullanılan özel bir 1 tam sayı türüdür. Üç mantıksal işleç boolean değerlerle kullanılır.

|  |  |
| --- | --- |
| && | and |
| || | or |
| ! | not |



### Operatörler

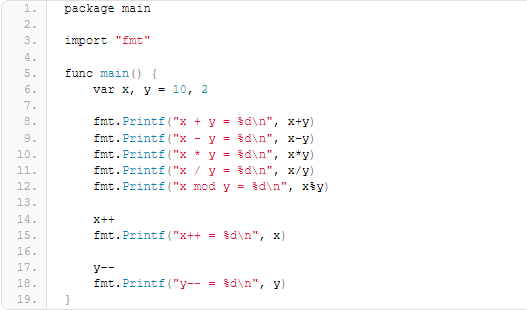
* **Aritmetik Operatörler** (Arithmetic Operators)
* **Atama Operatörleri** (Assignment Operators)
* **Karşılaştırma Operatörleri**(Comparison Operators)
* **Mantıksal Operatörler**(Logical Operators)
* **Bitsel Operatörler**(Bitwise Operators)

## **ARİTMETİK OPERATÖRLER**

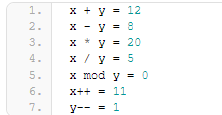
Aritmetik operatörler, toplama, çıkarma, çarpma vb. gibi matematiksel işlemleri gerçekleştirmek için kullanılır.



### **Aritmetik Operatörler Örneği**

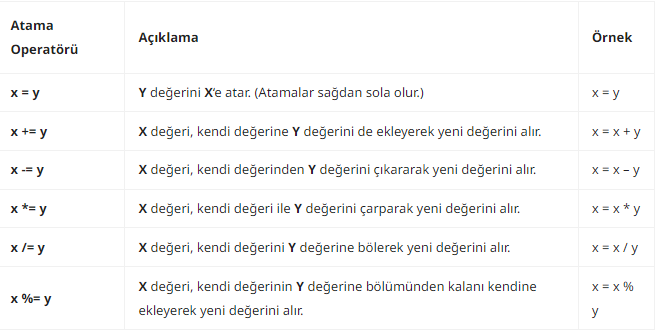


### **Kod Çıktısı:**

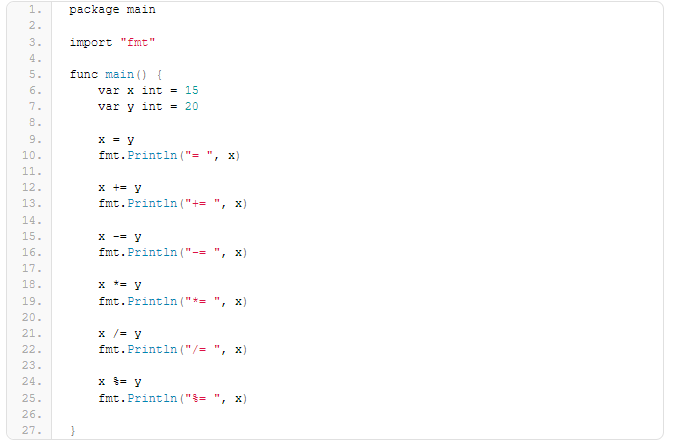


### **ATAMA OPERATÖRLERİ**

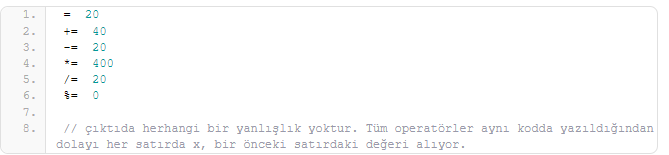
Atama operatörleri, değişkenlere değer atamak için kullanılır.



### **Atama Operatörleri Örneği**

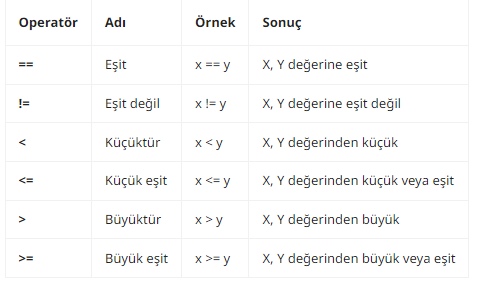


**Kod Çıktısı:**

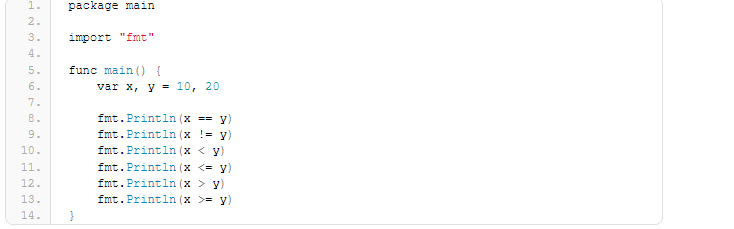


## **KARŞILAŞTIRMA OPERATÖRLERİ**

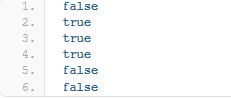
Karşılaştırma operatörleri, iki değer arasında karşılaştırma yapmak için kullanılır.



### **Karşılaştırma Operatörleri Örneği**

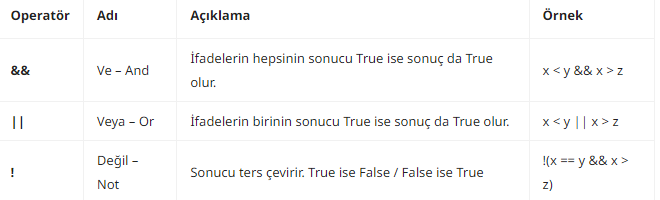


**Kod Çıktısı:**

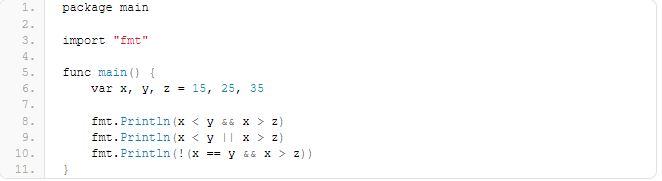


## **MANTIKSAL OPERATÖRLER**

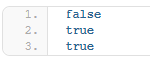
Mantıksal operatörler, değişkenler veya değerler arasındaki True ve False karşılaştırması yapar.



### **Mantıksal Operatörler Örneği**

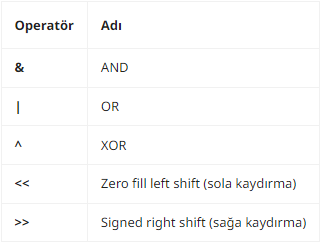


**Kod Çıktısı:**

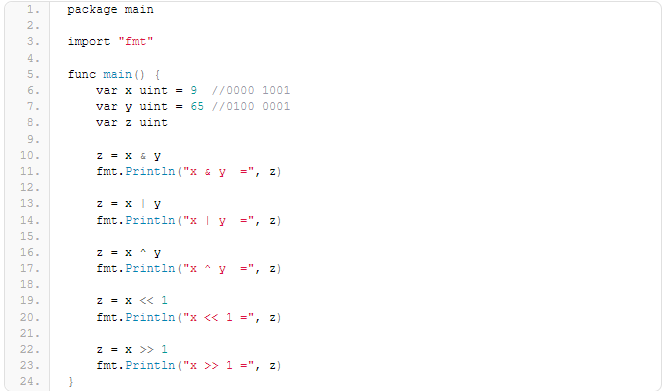


## **BİTSEL OPERATÖRLER**

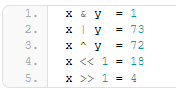
Bitsel operatörler (binary) sayıları karşılaştırmak için kullanılır.



### **Bitsel Operatörler Örneği**



**Kod Çıktısı:**



### Golang development company - FingersBoş tanımlayıcılar

Golang kodlarımızda bazen 2 adet değer döndüren fonksiyonlar kullanırız. Bu değerlerden hangisini kullanmak istemiyorsak, değişken adı yerine **\_ (alt tire)** kullanırız

Örnek:

*package* main  
  
*func* main() {  
  
 *var* \_ = 5  
  
 *// data, \_ =metot()*

### Type casting

*package* main  
  
*import* (  
 "fmt"  
 "strconv"  
)  
  
*func* main() {  
  
 *//TYPE CASTING  
  
  
 Sonucu küsüratlı bulmak halde bulmak için  
 var* toplam int=984  
 *var* sayi int=17  
  
 *var* sonuc=float32(toplam)/float32(sayi)  
  
 *var* s1=int(sonuc) Küsüratsı(düz)bulmak için (int) kullanıyoruz  
  
 fmt.Println(sonuc)  
  
 fmt.Println(s1)  
  
  
   
  
  
 TYPE CONVERSION  
  
  
   
 *var* str="1"  
  
 String Int'e Dönüşüm  
  
 *var* sayi,\_=strconv.Atoi(str)  
  
 fmt.Println(sayi)  
  
 *var* sonuc=sayi+7  
  
 fmt.Println(sonuc)  
  
  
   
  
 Int String'e Dönüşüm  
  
 *var* sayi=1  
  
 *var* str=strconv.Itoa(sayi)  
  
 fmt.Println(str)  
  
 fmt.Println("NASILSIN"+str)

### Konsol çıkış işlemleri print

*package* main  
  
*import* "fmt" (n)“Birleşik.Alt Alta yazacaktır.”  
  
*func* main(){  
  
 fmt.Println("Merhaba\n\n\nDünya") *// \n \t* fmt.Println("Hello\t\t\tWorld") (t)Boşluk bırakır  
  
 fmt.Println("Deniz Korkmazer’in \"Dünyası\" ")  
  
 fmt.Println("Malatya"+"Kayısısı"+"Ünlüdür")  
 fmt.Println("Malatya","Kayısısı","Ünlüdür")  
  
 fmt.Println("Lorem İpsum" +  
 "Merhaba Dünya" +  
 "Hello World")  
  
  
 isim:="Deniz”   
 soyisim:="Korkmazer"  
 sehir:="İzmir" Kaç karekter olduğunu öğrenmek   
 için.  
 fmt.Println(isim,soyisim,sehir)  
  
 *//fmt.Println(len(isim))  
  
 var* stringUzunlugu=len(isim)  
  
 fmt.Println(stringUzunlugu)

### Formatlama

Örnekler:

*package* main  
  
*import* "fmt"  
  
*func* main(){  
 Format gelecek  
 *var* sayi1 int=5  
  
 fmt.Printf("Benim Yaşım: %d",sayi1)  
  
 *var* sayi2 float32=3.1415925  
  
 fmt.Printf("PI SAYISI: %.2f",sayi2)  
 *var* dogruMu bool=true  
 fmt.Printf("%t",dogruMu)

2

*var* isim="Ahmet"  
*var* sehir="Ankara"  
*var* yas = 35  
  
fmt.Printf("Benim adım %s, %d yaşındayım ve %s doğdum",isim,yas,sehir)

### Konsol giriş işlemleri

1-

*package* main  
  
*import*(  
 "os"  
 "bufio"  
 "fmt"  
 "strconv"  
)  
  
  
  
*func* main() {  
  
   
 tarayici:=bufio.NewScanner(os.Stdin)  
  
 fmt.Print("Birşeyler Yazınız:")  
  
 tarayici.Scan()  
  
 verigirisi:=tarayici.Text()  
  
 fmt.Printf("Bunu yazdınız: %s",verigirisi)

2-

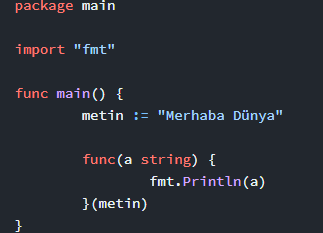
*package* main  
  
*import*(  
 "os"  
 "bufio"  
 "fmt"  
 "strconv"  
)  
*func* main() {

scanner:=bufio.NewScanner(os.Stdin)  
  
fmt.Print("Hangi Yıl Doğdunuz: ")  
  
scanner.Scan()  
verigirisi,\_:=strconv.ParseInt(scanner.Text(),10,64)  
  
fmt.Printf("Şuanda %d yaşındasın.",2021-verigirisi

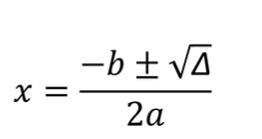
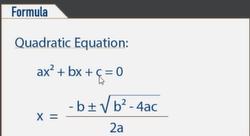
### VKI ÖRNEK (vÜCUT kİTLE eNDEKSİ)

*package* main  
  
  
  
*//Vücut Kitle İndeksi  
  
//Formül : VKI = kg/(boy\*boy)  
  
import*(  
 "os"  
 "bufio"  
 "fmt"  
 "strconv"  
)  
  
  
  
*func* main(){  
  
 scanner:=bufio.NewScanner(os.Stdin)  
  
 fmt.Print("Kilonuzu Giriniz:")  
 scanner.Scan()  
  
 kilo,\_:=strconv.ParseFloat(scanner.Text(),64)  
  
 fmt.Print("Boyunuzu Giriniz: ")  
 scanner.Scan()  
  
 boy,\_:=strconv.ParseFloat(scanner.Text(),64)  
  
 vki:=kilo/((boy/100)\*(boy/100))  
  
 fmt.Printf("Vücut Kitle İndeksiniz: %f",vki)

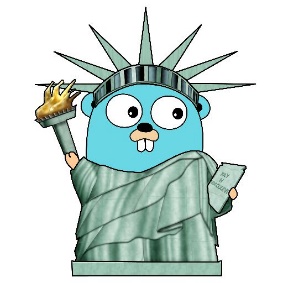
### Anonim foksiyonlar

Anonim fonksiyonların en büyük özelliği isimsiz olmalarıdır. Yazıldıkları yerde direkt olarak çalışırlar. Çalışırken diğer fonksiyonlardaki gibi parametre verilemediği için fonksiyonun sonuna parametre eklenerek çalışıtırılırlar. 

### kök bulma



*package* main  
  
*import* (  
 "bufio"  
 "fmt"  
 "math"  
 "os"  
 "strconv"  
)  
  
*func* main() {  
 scanner := bufio.NewScanner(os.Stdin)  
  
 fmt.Print("A sayisini giriniz")  
 scanner.Scan()  
  
 a, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 fmt.Printf("B sayisini giriniz:")  
  
 scanner.Scan()  
  
 b, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 fmt.Print("C sayisini giriniz:")  
  
 scanner.Scan()  
  
 c, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 delta := (math.Pow(b, 2)) - 4\*a\*c  
  
 kokbir := (-b - (math.Pow(delta, 0, 5))) / 2 \* a  
  
 kokiki := (-b + (math.Pow(delta, 0, 5))) / 2 \* a  
  
 fmt.Printf("1.Kök: %f\n2.Kök: %f \n Delta: %f", kokbir, kokiki, delta)  
  
}



### 2021 Yılında Neden GoLang Öğrenmelisiniz? | Siberci.com

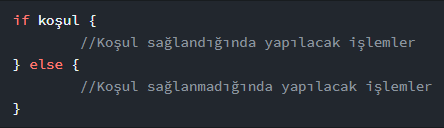
### konsol giriş işlemleri scanf

*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 *var* isim string  
 *var* yas int  
  
 fmt.Print("Adınızı Giriniz:")  
 fmt.Scan(&isim)  
  
 fmt.Print("Yaşınızı Giriniz:")  
 fmt.Scan(&yas)  
  
 fmt.Printf("Adınız:%s Yaşınız:%d", isim, yas)  
  
}

# koşullu durumlar

## If-Else

**If :** Eğer, **Else :** Yoksa anlamına gelir. **If-Else** akışı koşullandırmalar için kullanılır. Diğer dillerin aksine koşul parametresi parantezler içine yazılmaz.





*package* main  
  
*import* "fmt"

*func* main() {

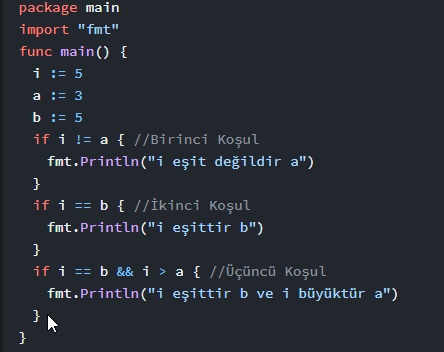
yas := 23  
  
 fmt.Println("18 Yaşından Küçükler Giremez")  
  
 *if* yas >= 18 {  
 fmt.Println("Mekana Hoşgeldiniz")  
 } *else if* yas < 18 && yas > 19 {  
 fmt.Println("Ailenle girebilirsin")  
 } *else* {  
 fmt.Println("GİREMEZSİNİZ")  
 }  
}

### ELSE-IF KULLANIMI

**If-Else** akışında birden fazla koşul kontrolü ekleyebiliriz. Bunu **else if** deyimi ile yapabiliriz.

*package* main  
  
*import* "fmt"  
  
*func* main() { Bu kısma “Deniz1” gibi bir kod yazarsak Else olacaktır  
  
 isim := "Deniz"  
 *if* isim == "Deniz" {  
 fmt.Println("Tekrar Hoş Geldin Deniz")  
 } *else* {  
 fmt.Println("Lütfen çıkış yapınız")  
 }

### **Koşullar İçerisinde Operatör Kullanımı**



### IF ELSE ile Not Uygulaması

*package* main  
  
*import* (  
 "bufio"  
 "fmt"  
 "os"  
 "strconv"  
)  
  
*func* main() {  
  
 scanner := bufio.NewScanner(os.Stdin)  
  
 fmt.Print("Vize notunu lütfen giriniz")  
 scanner.Scan()  
  
 vizenot, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 fmt.Print("Lütfen final notunuzu giriniz")  
 scanner.Scan()  
  
 finalnot, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 ortalama := (vizenot \* 0.4) + (finalnot \* 0.6)  
  
 fmt.Println("Ortalama:", ortalama)  
  
 *if* ortalama >= 85 && ortalama < 100 {  
 fmt.Println("AA")  
 } *else if* ortalama >= 70 && ortalama < 85 {  
 fmt.Println("BB")  
 } *else if* ortalama >= 60 && ortalama < 70 {  
 fmt.Println("CB")  
 } *else if* ortalama >= 50 && ortalama < 60 {  
 fmt.Println("CC")  
 } *else if* ortalama >= 50 && ortalama < 0 {  
 fmt.Println("FF")  
 } *else* {  
 fmt.Println("0-100 Arası bir değer olmalıdır")  
  
 }  
  
}

### 

## IF ELSE ile hesap makinesi

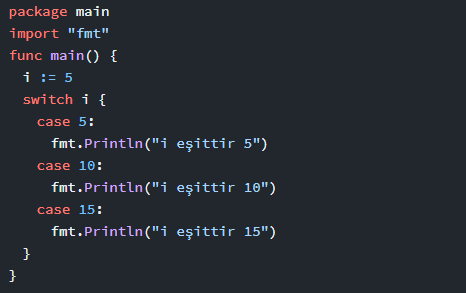
*package* main  
  
*import* (  
 "bufio"  
 "fmt"  
 "os"  
 "strconv"  
)  
  
*func* main() {  
  
 scanner := bufio.NewScanner(os.Stdin)  
  
 fmt.Println("Lütfen seçim yapınız")  
 fmt.Println("1.Toplama\n" +  
 "2-Çıkarma\n" +  
 "3-Çarpma\n" +  
 "4-Bölme\n" +  
 "5-Mod Alma")  
  
 scanner.Scan()  
  
 secim := scanner.Text()  
  
 fmt.Print("1.Sayıyı Giriniz")  
 scanner.Scan()  
 sayi1, \_ := strconv.ParseFloat(scanner.Text(), 64)  
 fmt.Print("2.Sayıyı Giriniz")  
 scanner.Scan()  
 sayi2, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 *if* secim == "1" {  
 fmt.Printf("Toplama:%f", sayi1+sayi2)  
 } *else if* secim == "2" {  
 fmt.Printf("Çıkarma:%f", sayi1-sayi2)  
 } *else if* secim == "3" {  
 fmt.Printf("Çarpma:%f", sayi1\*sayi2)  
 } *else if* secim == "4" {  
 fmt.Printf("Bölme:%f", sayi1/sayi2)  
 } *else if* secim == "5" {  
 modAlma := int(sayi1) % int(sayi2)  
 fmt.Printf("Mod alma:%d", modAlma)  
 } *else* {  
 fmt.Println("Hatalı giriş yaptınız...")  
  
 }  
  
}

### Switch(CaSe)

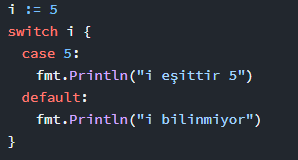
Switch kelimesinin Türkçe’deki anlamı **anahtardır**. Switch deyimi de if-else deyimi gibi koşul üzerine çalışır. **Case** deyimi durumu ifade eder. Koşul sağlandığı zaman işleme devam edilmez.

*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 sayi := 5  
  
 *switch* sayi {  
 *case* 5:  
 fmt.Println("Sayı 5'e eşittir.")  
 *case* 10:  
 fmt.Println("Sayı 10'a eşittir.")  
 *default*:  
 fmt.Println("Tanımsız")  
   
 }  
}

2-



Switch’te koşulların gerçekleşmediği zaman işlem uygulamak istiyorsak bunu **default** terimi ile yaparız. Örnek;



Örnek:

harf:="z"  
  
*switch* harf{  
*case* "a","e","i","o","u":  
 fmt.Println("Sesli Harf")  
*default*:  
 fmt.Println("Sessiz Harf")  
}

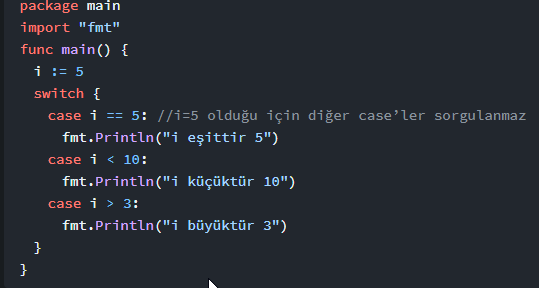
### Koşullu Switch

*switch*{  
*case* sayi==5:  
 fmt.Println("Sayı 5'e eşittir.")  
 *fallthrough 2 case’ye bakıyor   
case* sayi<10:  
 fmt.Println("Sayı 10'dan küçüktür.")  
*default*:  
 fmt.Println("Tanımsız")  
}

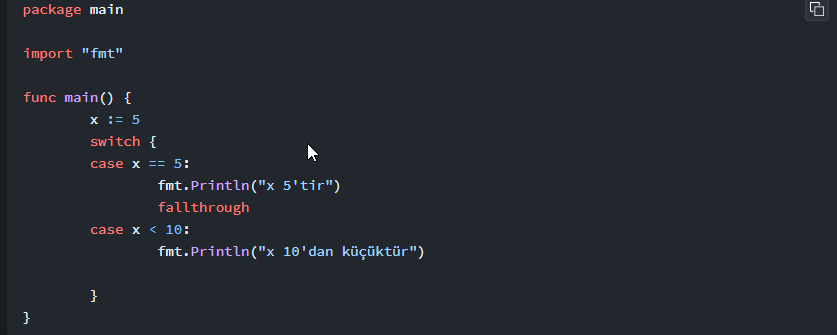
*switch* sayi:=25;{  
*case* sayi>=0 && sayi<=10:  
 fmt.Println("Sayı 0'dan büyük 10'dan küçük yada eşittir.")  
*case* sayi>=11 && sayi<=30:  
 fmt.Println("Sayı 11'den büyük ve eşit yada 30'dan küçük yada eşit")  
*default*:  
 fmt.Println("Aralık tanımsız")  
}

### **Koşulsuz Switch**

Yöntemde switch deyiminin yanına koşul girmek yerine case deyiminin yanına koşul giriyoruz.

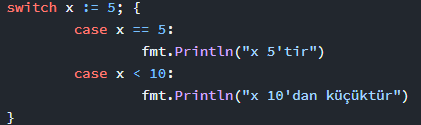


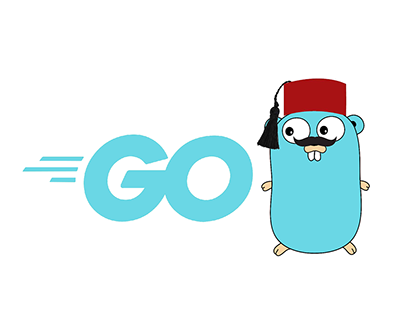
### Sonraki Koşulu Kontrol Ettirme



### Switch'e Özel Değişken Tanımlama

Tıpkı If deyimindeki Switch içerisinde kullanabileceğimiz değişkenler tanımlayabiliriz.





### Switch case hesap makinesi

*package* main  
  
*import* (  
 "bufio"  
 "fmt"  
 "os"  
 "strconv"  
)  
  
*func* main() {  
 scanner := bufio.NewScanner(os.Stdin)  
  
 fmt.Println("1-Toplama\n" +  
 "2-Çıkarma\n" +  
 "3-Çarpma\n" +  
 "4-Bölme\n" +  
 "Lütfen İşlem Seçiniz")  
 scanner.Scan()  
  
 secim := scanner.Text()  
  
 fmt.Print("1.Sayıyı Giriniz")  
 scanner.Scan()  
 sayi1, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 fmt.Print("2.Sayıyı Giriniz:")  
 scanner.Scan()  
 sayi2, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 *switch* secim {  
 *case* "1":  
 fmt.Println("Toplama:", sayi1+sayi2)  
 *case* "2":  
 fmt.Println("Çıkarma:", sayi1-sayi2)  
 *case* "3":  
 fmt.Println("Çarpma:", sayi1\*sayi2)  
 *case* "4":  
 fmt.Println("Bölme:", sayi1/sayi2)  
 *default*:  
  
 fmt.Println("Hatalı Seçim Yaptınız")  
  
 }  
  
}



# Döngüler

Programlama dillerinde **while, do while** ve **for** döngüleri vardır. Bu döngüler ile yapacağımız işlemin belirli koşullarda tekrarlanmasını sağlayabiliriz. Golang’ta ise diğer dillerin aksine sadece **for** döngüsü vardır. Ama bu **while** ve **do while** ile yapılanları yapamayacağımız anlamına gelmiyor. Golang’taki for döngüsü ile hepsini yapabiliriz. Yani dilin yapımcıları tek döngü komutu ile hepsini yapabilmemize olanak sağlamışlar.

Gelelim for döngüsünün kullanımına. Go’da for döngüsü parametreleri parantez içine alınmaz.

### STANDART FOR KULLANIMI

*package* main  
  
*import* "fmt"  
  
*func* main() { Buraya Farklı bir sayı geldiğinde örnek 10 < olarak   
 x := 0 gösterecektir.  
  
 *for* x < 5 {  
 fmt.Println(x)  
 x += 1  
 }  
}

Konsol çıktımız şu şekilde olacaktır;

0

1

2

3

4

2.Örnek:

*package* main  
  
*import* "fmt" ++ veya i yazmak yukardaki örnekle aynı işlemi  
  
*func* main() { yapacaktır daha uzun olarak.  
 *for* i := 0; i < 10; i++ {  
 fmt.Println(i)  
 }  
}  
}

Konsol çıktımız şu şekilde olacaktır;

0 -1-2-3-4-5-6-7-8-9

### Çift Sayı Bulma Döngüsü

*package* main  
  
*import* "fmt"  
  
*func* main() {  
 *//x := 0  
  
 //for x < 5 {  
 fmt.Println(x) “Noktalı virgül kullanımına dikkat edelim”  
 //x += 1  
 //} Eğer buraya bir sayı Örn:1 Verirsek sonsuza kadar   
  
 for* i := 0; i <= 100; i++ { Bizlere çift sayı yazacaktır.  
 *if* i%2 == 0 {  
 fmt.Printf("Çift Sayılar:%d\n", i)  
 }  
 } \n kulannımı alt alta yazmamızı

sağlar.   
}

### İç içe döngüler

*package* main  
  
*import* "fmt"  
  
*func* main() {  
 *for* i := 1; i <= 5; i++ {  
 *for* j := 1; j <= 5; j++ {  
 fmt.Printf("İ:%d J:%d \n", i, j)   
 }  
 }  
}

### çarpım tablosu

*for* i:=1;i<=10;i++{  
 fmt.Println("\*\*\*\*\*\*\*\*\*")  
 *for* j:=1;j<=10;j++{  
 fmt.Printf("%d X %d = %d\n",i,j,i\*j)  
 }  
 }  
  
  
  
  
  
}

### Break continue

*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 *for* i := 1; i <= 5; i++ {  
 *if* i == 3 {  
 *break Break döngüyü tamamen bitiriyor.* }  
 fmt.Println(i)  
 }  
}

//

*for* i := 1; i <= 5; i++ { Kendisinden sonra gelen işlemi sonlandırıyor.  
 *if* i == 3 {  
 *continue Ama döngüye devam ettiriyor!!* }  
 fmt.Println(i)  
 }  
}

### sonsuz döngüler/forever

*package* main  
  
*import* "fmt"  
  
*func* main() {  
 *for* {  
  
 fmt.Println("Merhaba Ermat") Sonsuz yazacaktır.  
 }  
}

*for* i := 1; i <= 100; {  
 fmt.Println(i) Sonsuz yazacaktır.  
}

& while de forever ile aynıdır sadece yazımları farklıdır.

*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 x := 1  
  
 *for* x < 10 {  
 fmt.Println(x)  
 x++  
 }  
}

### For uygulaması kullanıcı girişi

*package* main  
  
*import* (  
 "bufio"  
 "fmt"  
 "os"  
)  
  
*func* main() {  
  
 k\_adi := "Deniz"  
 k\_sifre := "12345"  
  
 giris\_hakki := 3  
  
 fmt.Println("\*\*\*\*LOGIN\*\*\*\*")  
  
 *for* true {  
  
 scanner := bufio.NewScanner(os.Stdin)  
  
 fmt.Print("Kullanıcı Adını Giriniz:")  
  
 scanner.Scan()  
  
 giris\_kadi := scanner.Text()  
  
 fmt.Print("Kullanıcı Şifresini Giriniz:")  
  
 scanner.Scan()  
  
 giris\_ksifre := scanner.Text()  
  
 *if* k\_adi != giris\_kadi && k\_sifre != giris\_ksifre {  
 fmt.Println("Kullanıcı Adınız ve Şifreniz Yanlış")  
 giris\_hakki--  
 } *else if* k\_adi == giris\_kadi && k\_sifre != giris\_ksifre {  
 fmt.Println("Şifreniz Yanlıştır.")  
 giris\_hakki--  
 } *else if* k\_adi != giris\_kadi && k\_sifre == giris\_ksifre {  
 fmt.Println("Kullanıcı Adınız Yanlıştır.")  
 giris\_hakki--  
 } *else* {  
 fmt.Println("Hoşgeldiniz")  
 *break* }  
  
 *if* giris\_hakki == 0 {  
 fmt.Println("Giris Hakkınız Sona Erdi")  
 *break* }  
  
 }  
  
}

### fizzbuzz

*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 *for* i := 1; i <= 100; i++ {  
 *if* i%3 == 0 {  
 fmt.Println("FİZZ")  
 } *else if* i%5 == 0 {  
 fmt.Println("BUZZ")  
 } *else if* i%3 == 0 && i%5 == 0 {  
 fmt.Println("FİZZBUZZ")  
 }  
 }  
}

### range for yapısı

Range yapısı diziler üzerinde yapılacak iteratif işlemler için kullanılmaktadır.

*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 kelime := "Go Prog"  
 *for* ind, c := *range* kelime {  
 fmt.Printf("%d indisindeki karekter = %c \n", ind, c)  
 }  
}

-1. Karekter G 2. Karekter o ….gibi vs.

### etiket(label)kullanımı ve goto komutu

Etiketler(label)programın belirli bir kısmının isimlendirilmesidir break ve continue ifadesinde isteğe bağlı olarak kullanılır.Ancak goto ifadesinde kullanımı zorunludur.

<Label adı>:

*package* main  
  
*import* "fmt"  
 Not: goto ve label blokları arasında yeni değişken  
*func* main() { tanımlaması yapılmaz.   
 fmt.Print("1")   
 *goto* **End** fmt.Print("2")  
**End**:  
 fmt.Print("3")  
}

### for uygulaması hesap makinesi

*package* main  
  
*import* (  
 "bufio"  
 "fmt"  
 "os"  
 "strconv"  
)  
  
*func* main() {  
  
 *for* true {  
  
 scanner := bufio.NewScanner(os.Stdin)  
  
 fmt.Println("\nİŞLEM SEÇİNİZ\n" +  
 "1-Toplama\n" +  
 "2-Çıkarma\n" +  
 "3-Çarpma\n" +  
 "4-Bölme\n" +  
 "ÇIKIŞ İÇİN q'ya basınız..")  
  
 scanner.Scan()  
 secim := scanner.Text()  
  
 *if* secim == "q" || secim == "Q" {  
 fmt.Println("Çıkış yapılıyor...")  
 *break* }  
  
 fmt.Print("1. sayıyı Giriniz:")  
 scanner.Scan()  
 sayi1, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 fmt.Print("2. sayıyı Giriniz:")  
 scanner.Scan()  
 sayi2, \_ := strconv.ParseFloat(scanner.Text(), 64)  
  
 *if* secim == "1" {  
 fmt.Printf("%.2f + %.2f = %.2f", sayi1, sayi2, sayi1+sayi2)  
 } *else if* secim == "2" {  
 fmt.Printf("%.2f - %.2f = %.2f", sayi1, sayi2, sayi1-sayi2)  
 } *else if* secim == "3" {  
 fmt.Printf("%.2f X %.2f = %.2f", sayi1, sayi2, sayi1\*sayi2)  
 } *else if* secim == "4" {  
 fmt.Printf("%.2f / %.2f = %.2f", sayi1, sayi2, sayi1/sayi2)  
 } *else* {  
 fmt.Println("Hatalı Giriş")  
 }  
  
 }  
  
}

İpucu: Go programlama dilinde goto kullanımı tercih edilen bir yapı değildir.

### FOR UYGULAMASI FAKTORİYEL BULMA

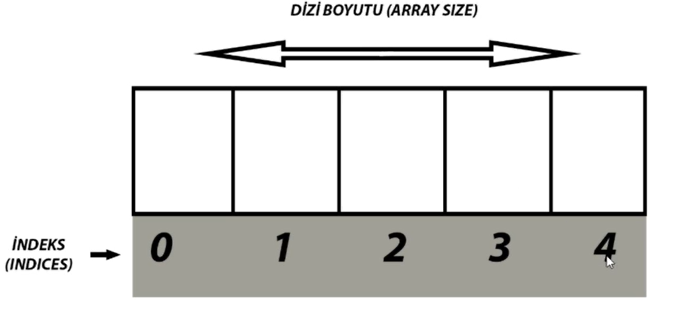
*package* main  
  
*import* "fmt"  
  
*/\* 1'den 10'a kadar olan sayıların Faktoriyel  
  
  
 5!=5x4x3x2x1 = 120  
 n!=n x (n-1)!  
 1!=1  
 0!=1  
  
  
 \*/  
  
  
  
func* main(){  
  
 *for* i:=1;i<=10;i++{  
 faktoriyel:=1  
 *for* j:=1;j<=i;j++{  
 faktoriyel\*=j *//faktoriyel=faktoriyel\*j* }  
 fmt.Println(i,"! =",faktoriyel)  
 }  
  
  
}

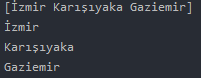
### FOR UYGULAMASI MÜKEMMEL SAYI

*package* main  
  
*import* "fmt"  
  
*/\*  
 MÜKEMMEL SAYI;  
  
 Kendi bölünenlerinin toplamının yine kendisine eşit olduğu sayılardır.  
  
 28 = 14+7+4+2+1  
  
 6 = 3+2+1  
  
 496,8128,33550336  
  
  
 \*/  
  
  
func* main(){  
  
 *var* sayi int  
  
 fmt.Print("Sayı Giriniz:")  
 fmt.Scanf("%d",&sayi)  
  
 toplam:=0  
  
 *for* i:=1;i<sayi;i++{  
 *if* sayi%i==0{  
 toplam+=i  
 }  
 }  
  
 *if* toplam==sayi{  
 fmt.Printf("%d sayısı Mükemmel Sayıdır.",sayi)  
 }*else*{  
 fmt.Printf("%d sayısı Mükemmel Sayı Değildir.",sayi)  
 }  
  
  
}

# diziler(arrays)

Diziler içlerinde bir veya birden fazla değer tutabilen birimlerdir. Dizi, aynı türden öğeler dizisinin adlandırılmış **sabit bir boyut**unu temsil eder.Bir dizideki her değer sırasıyla numaralandırılır. Numaralandırma sıfırdan başlar.



*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 *var* dizi [3]string  
 dizi[0] = "İzmir"  
 dizi[1] = "Karışıyaka"   
 dizi[2] = "Gaziemir"  
  
 fmt.Println(dizi)  
 fmt.Println(dizi[0])  
 fmt.Println(dizi[1])  
 fmt.Println(dizi[2])  
}

dizi2 := [5]int{1, 3, 5, 7, 9}  
  
 fmt.Println(dizi2)  
 fmt.Println(dizi2[0]) [1 3 5 7 9]

1  
}

## Not:Array statik yapılıdır.List Dinamik yapılıdır

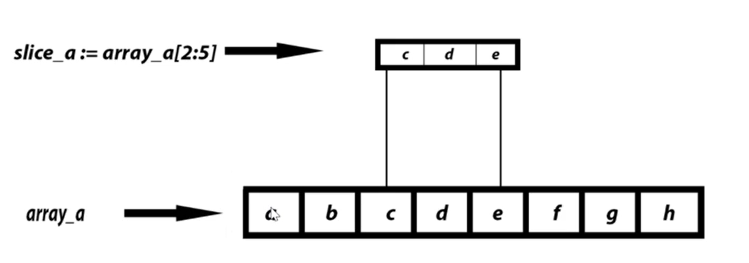
*package* main  
  
*import* "fmt"  
  
*func* main() { Buraya “...” harici sayı yazarsak yazdığımız sayının boyutu kadar olmalıdır /... koyduğumuzda sınırsız yazmaktadır Örnek:[3]int{1,2,3’ten sonrası hata verecektir  
   
 arr:=[...]int{1,2,3,4,5,6} [0 0 0]   
 fmt.Println(arr)  
  
}

fmt.Println(lem(arr)) len komutu=kaç eleman var cevap=6

*for* i:=0;i<=len(arr)-1;i++{  
 fmt.Println(arr[i])

## Not:cap KOMUTU:dizinin kaç eleman alabileceğini söylüyor

## not len komutu=kaç eleman var



### çok boyutlu diziler

*package* main  
  
*import* "fmt"  
  
*func* main() {  
 *var* dizi2D = [5][2]int{{0, 0},  
 {1, 2},  
 {2, 4},  
 {3, 6},  
 {4, 8}}  
  
 fmt.Println(dizi2D)  
  
 fmt.Println(dizi2D[3][0])  
 fmt.Println(dizi2D[4][1])  
  
 *for* i := 0; i < 5; i++ {  
 *for* j := 0; j < 2; j++ {  
 fmt.Printf("Dizi2D[%d][%d] = %d\n", i, j, dizi2D[i][j])  
 }  
 }  
 *//3 Boyutlu Diziler  
 var* dizi3D = [3][3][3]int{{{1, 2, 3}, {1, 2, 3}, {1, 2, 3}}, {{1, 2, 3}, {1, 2, 3}, {1, 2, 3}}, {{1, 2, 3}, {1, 2, 3}, {1, 2, 3}}}  
  
 fmt.Println(dizi3D[0][2][1])  
  
}

### slices

*package* main  
  
*import* "fmt"  
  
*func* main() {  
  
 dizi := [5]int{1, 3, 5, 7, 9}  
  
 *var* dilim []int = dizi[0:5]  
 */\*  
 dizi[0:10]  
 dizi[:10]  
  
 dizi[0:]  
  
 dizi[:]  
 \*/* fmt.Println(dilim)  
  
 a := [5]string{"Go", "Python", "Kotlin", "PHP", "JavaScript"}  
 b := a[1:3]  
 fmt.Println("Slice:", b)  
 fmt.Println("Slice Kapasitesi:", cap(b))  
 fmt.Println("Slice Uzunluğu:", len(b))  
 fmt.Println("Array:", a)  
  
}

**Boş Dilimler (Nil Slices)** Boş bir dilimin varsayılan (sıfır) değeri **nil**’dir.

*package* main  
*import* "fmt"  
*func* main() {  
  *var* a []int  
  *if* a == nil {  
   fmt.Println("Boş")  
  }  
}

**Dilim Uzunluğu ve Kapasitesi** Bir dilimin **uzunluk** ve **kapasite** değeri vardır. Dilimin uzunluğunu **len()** fonksiyonu ile, kapasitesini ise **cap()** fonksiyonu ile hesaplarız.

*package* main  
*import* "fmt"  
*func* main() {  
  a := [6]int{2, 3, 5, 6, 7, 9}  
  b := a[2:4]  
  fmt.Println("a uzunluk", len(a))  
  fmt.Println("a kapasite", cap(a))  
  fmt.Println("a'nın içeriği", a)  
  fmt.Println("b uzunluk", len(b))  
  fmt.Println("b kapasite", cap(b))  
  fmt.Println("b'nin içeriği", b)  
}

**Make ile Dilim Oluşturma** Dilimler **make** fonksiyonu ile de oluşturulabilir. Dinamik büyüklükte diziler oluşturabiliriz.

a := make([]int, 5)

-

a := make([]int, 0, 5)

**Dilime Ekleme Yapma** Bir dilime ekleme yapmak için append fonksiyonu kullanılır.

*package* main  
*import* "fmt"  
*func* main() {  
  *var* a []string  
  fmt.Println(a) *//[ ]*  a = append(a, "Ali")  
  a = append(a, "Veli")  
  fmt.Println(a) *//[Ali Veli]*}



### SLİCE APPEND MAKE METOTLARI

*package* main  
  
*import* "fmt"  
  
*func* main(){  
  
 *//yeni bir oluşturma  
 var* a []int=[]int{11,22,33,44,55}  
  
 *//fmt.Printf("%T",a)* fmt.Println(a)  
  
 *//append  
  
 /\*  
 b:=append(a,66)  
 fmt.Println(b)  
  
 \*/* a=append(a,66)  
  
 fmt.Println(a)  
  
 a=append(a,77)  
  
 fmt.Println(a)  
  
 *var* sehirler=[]string{"İstanbul","Ankara","Manisa"}  
 sehirler=append(sehirler,"Antalya")  
 sehirler=append(sehirler,"İzmir")  
  
 fmt.Println(sehirler)  
  
 *//make* sayilar:=make([]int,5)  
  
 fmt.Println(sayilar)  
  
 sayilar[0]=3  
 sayilar[1]=6  
 sayilar[2]=9  
 sayilar[3]=12  
 sayilar[4]=15  
  
 fmt.Println(sayilar)  
  
 sayilar=append(sayilar,18)  
  
 fmt.Println(sayilar)  
  
 fmt.Println("Kapasite:",cap(sayilar))  
 fmt.Println("Uzunluk:",len(sayilar))  
  
}

### Range for yapısı

Range yapısı diziler üzerinde yapılacak iteratif işlemler için kullanılmaktadır.

*package* main  
  
*import* "fmt"  
  
*func* main() {  
 *var* a []int = []int{1, 5, 6, 7, 8, 44}  
 */\*  
 for i:=0;i<len(a);i++{  
 fmt.Println(a[i])  
 }  
 \*/  
  
 for* index, eleman := *range* a {  
 fmt.Printf("%d: %d \n", index, eleman)  
 }  
  
 isim := "Ferhat Korkmaz"  
  
 *for* i, harf := *range* isim {  
 fmt.Printf("%d = %c\n", i, harf)  
 }  
}

### maps yapısı

Map bir anahtar(key)-deger(value) ilişkisine bağlı olan veri türüdür.Esasen array ve slice yapısına benzemekle birlikte,map türünde öğeler sırasız bir şekilde ve bir anahtara bağlı olarak yer alırlar.

*package* main  
  
*import* "fmt"  
  
*// MAPS  
func* main() {  
  
 *//Standart oluşturma yöntemi  
  
 var* ogrenci *map*[int]string = *map*[int]string{  
 10: "Deniz",  
 21: "Ferhat",  
 39: "Sercan",  
 89: "Merve",  
 }  
  
 fmt.Println(ogrenci)  
  
 fmt.Println(ogrenci[39])  
  
 ogrenci[21] = "Ferhat"  
  
 fmt.Println(ogrenci)  
  
 *//key:int value:string* plaka := make(*map*[int]string)  
  
 plaka[44] = "İstanbul"  
 plaka[35] = "İzmir"  
 plaka[01] = "Sivas"  
  
 fmt.Println(plaka)  
  
 fmt.Println(plaka[35])  
  
 *//key:string value:string* unvanlar := make(*map*[string]string)  
  
 unvanlar["DR"] = "DOKTOR"  
 unvanlar["PRF"] = "PROFESOR"  
 unvanlar["DOC"] = "DOCENT"  
  
 fmt.Println(unvanlar["PRF"])  
  
 a := unvanlar["DR"]  
  
 fmt.Println(a)  
  
 *//silme* delete(unvanlar, "DR")  
  
 fmt.Println(unvanlar)  
  
 fmt.Println(len(unvanlar))  
  
 *for* key, value := *range* unvanlar {  
 fmt.Printf("Kısaltılması:%v Tam Hali:%v\n", key, value)  
 }  
}

### **Bölgesel Silme İşlemi** **delete** fonksiyonu ile silme işlemimizi yapabiliriz.

*package* main  
*import* "fmt"  
*func* main() {  
  m := make(*map*[string]int) *//m isminde string bölge isimli int değer taşıyan dizi*  m["sayi"] = 25 *//sayi bölgesine 25 değerini yerleştirdik*  fmt.Println(m["sayi"]) *//Çıktımız: 25*  delete(m, "sayi") *//sayi bölgesindeki değeri sildik*  fmt.Println(m["sayi"]) *//Çıktımız: 0 (sıfır)*}



### array 2.örnek (alıştırma)

*package* main  
  
*import* "fmt"  
*//Dizi Elemanlarını Çarpma  
func* main(){  
 *var* dizi1 [2][3]int  
 *var* dizi2 [2][3]int  
 *var* dizisonuc [2][3]int  
 fmt.Print("1.Dizinin Elemanlarını Giriniz:\n")  
 *for* k,r:=*range* dizi1{  
 *for* l:=*range* r{  
 fmt.Scan(&dizi1[k][l])  
 *//dizi1[0][0] [0][1]* }  
 }  
 fmt.Println("2.Dizinin Elemanlarını Giriniz:\n")  
  
 *for* m,rr:=*range* dizi2{  
 *for* n:=*range* rr{  
 fmt.Scan(&dizi2[m][n])  
 }  
 }  
 fmt.Println("Dizilerin Çarpımı Sonucu:\n")  
  
 *for* i,satir:=*range* dizisonuc{  
 *for* j:=*range* satir{  
 dizisonuc[i][j]=dizi1[i][j]\*dizi2[i][j]  
  
 fmt.Print(dizisonuc[i][j],"\t")  
 }  
 fmt.Println()  
 }  
}

### slice örnek(alışırma)

*package* main  
*import* (  
 "bufio"  
 "fmt"  
 "os"  
)  
*func* main(){  
 ulkeler:=make([]string,5)  
 *for* i:=0;i<6;i++{  
 scanner:=bufio.NewScanner(os.Stdin)  
 fmt.Print("Ülke Giriniz:")  
 scanner.Scan()  
 ulkegiris:=scanner.Text()  
 ulkeler=append(ulkeler,ulkegiris)  
 }  
 *for* index,eleman:=*range* ulkeler{  
 *if* index<=5{  
 *continue* }  
 fmt.Printf("Girdiğiniz %d. Ülke:%s\n",index-5,eleman)  
 }

### Map örnek(alıştırma)

*import* "fmt"  
  
*func* main(){  
 sozluk:=make(*map*[string]string)  
  
 *var* eng string  
 *var* tr string  
  
 *for* i:=1;i<=5;i++{  
  
 fmt.Print("İngilizce Kelime Giriniz:")  
 fmt.Scan(&eng)  
  
 fmt.Print("Türkçe Karşılığını Giriniz:")  
 fmt.Scan(&tr)  
  
 sozluk[eng]=tr  
 }  
 fmt.Println(sozluk)  
 fmt.Print("Silmek İstediğiniz İngilizce Kelimeyi Giriniz:")  
 fmt.Scan(&eng)  
 delete(sozluk,eng)  
 *for* k,v:=*range* sozluk{  
 fmt.Printf("ENG:%s \t TR:%s \n",k,v)  
 }  
}

# hata yönetimi

Error interface’i sadece bir metot imzasına sahiptir.Herhangi bir tip bu interface’i implement edebilir.

## Not:ctrl-a veya belirli bir metine kadar kodları seçip ctrl/ işareti yaptığımızda seçtiğimiz kodları yorum satırı haline getirecektir.

*package* main  
*import* (  
 "bufio"  
 "errors"  
 "fmt"  
 "log"  
 "os"  
 "strconv"  
)  
*func* main() {  
 2 paket vardır 1-errors 2-fmt  
 nill => Null  
 scanner := bufio.NewScanner(os.Stdin)  
   
 fmt.Print("Sayı Giriniz")  
   
 a, err := strconv.ParseInt(scanner.Text(), 10, 64)  
 *if* err != nil {  
 *//fmt.Println(err.Error())* log.Fatal("HATA", err)  
 }  
   
 fmt.Println(a)

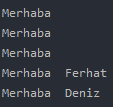
2-

file, err := os.Open("/main.go")  
 *if* err != nil {  
 fmt.Println(errors.New("DOSYA YOK"))  
 }  
 fmt.Println(file.Name(), "BAŞARILI BİR ŞEKİLDE AÇILDI")  
}

# FONKSİYONLAR(FUNCTİONS)

Fonksiyonlar içlerine parametre girilebilen ve işlemler yapabilen birimlerdir. Bir işlev, belirli bir görevi gerçekleştiren ifadeler bloğunu temsil eder. Bir fonksiyon bildirimi bize fonksiyon adını, dönüş tipini ve giriş parametrelerini bildirir. İşlev tanımı, işlevde bulunan kodu temsil eder.

*package* main  
  
*import* "fmt"  
  
*//built-in func:gömülü fonk.  
//user-defined func  
  
func* merhaba(){ *//Parametresiz Func.* fmt.Println("Merhaba")  
}  
  
*func* merhabaPar(mesaj string){ *//Parametreli Func* fmt.Println("Merhaba ",mesaj)

*func* main(){   
   
 merhaba()  
 merhaba()  
 merhaba()  
  
 merhabaPar("Ferhat")  
 merhabaPar("Deniz")  
}

## Return Fonksiyonlar

*package* main  
  
*import* "fmt"  
  
*func* toplama(x int,y int)int{  
 *return* x+y  
}  
*func* onlacarp(t int)int{  
 *return* t\*10  
}  
*func* bolme(a,b float32,c string)float32{  
 *return* a/b  
}  
*func* ikiylecarp(g int)int{  
 *return* g\*2  
}  
  
*func* main(){  
  
 fmt.Println(toplama(7,8))  
  
 a:=onlacarp(8)  
 fmt.Println(a)  
  
 fmt.Println(bolme(22.7,13.3,"AHKS"))  
  
 *for* i:=0;i<=10;i++{  
 fmt.Println(ikiylecarp(i))  
 }  
  
}

15,80,1.706767,0,2,4,6,8,10,12,14,16,18,20

### MultipleReturn Fonksiyonlar

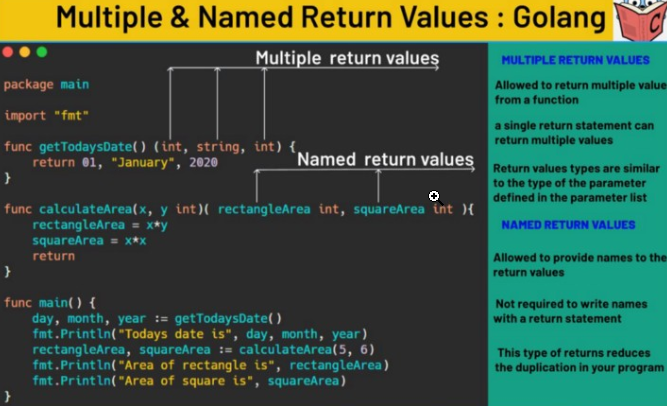
*package* main  
  
*import* "fmt"  
*/\* snytax  
fun fonksiyon\_adı(parametre1,parametre2,...)(dönüş veri tipi1,dönüş veri tipi2){  
// işlemler..  
\*/  
func* ucislem(x, y int) (int, int, int) {  
 *return* x - y, x + y, x \* y  
}  
*func* main() {  
 sayi1, sayi2, sayi3 := ucislem(5, 9)  
  
 fmt.Printf("Sonuc:%d \n", sayi1)  
 fmt.Printf("Sonuc:%d \n", sayi2)  
 fmt.Printf("Sonuc:%d \n", sayi3)  
} Sonuc:-4

Sonuc:14

Sonuc:45

2-

*package* main  
  
*import* "fmt"  
 *func* ucislem(x, y int) (int, int, int) {  
 *return* x - y, x + y, x \* y  
}  
*func* maxmin(a, b int) (int, int) {  
 *if* a > b {  
 *return* a, b  
 } *else* {  
 *return* b, a  
 }  
}  
*func* main() {max, min := maxmin(30, 60)  
  
 fmt.Printf("Max=%d Min=%d", max, min) //=Max=60 Min=30  
}



### NAMED RETURN Fonksiyonlar

*package* main  
  
*import* (  
 "fmt"  
 "math"  
)  
*const* PI=3.14  
*func* hesaplama(a,b int)(carp int,bol int){  
 carp=a\*b  
 bol=a/b  
 *return*}  
*func* daire(yaricap float64)(alan float64,cevre float64){  
 alan=PI\*(math.Pow(yaricap,2.0))  
 cevre=PI\*2\*yaricap  
 *return*}  
*func* main(){  
 */\*  
 carpim,bolme:=hesaplama(66,6)  
  
 fmt.Println("66 X 6 =",carpim)  
 fmt.Println("66 / 6 =",bolme)  
 \*/* dairealan,dairecevre:=daire(5.0)  
  
 fmt.Println("Dairenin Alanı:",dairealan)  
 fmt.Println("Dairenin Çevresi:",dairecevre)  
} 66x6= 396

66x6= 11

Dairenin Alanı: 78.5

Dairenin Çevresi: 31.400000000000002

### variadic Fonksiyonlar

Rasgele sayıda argüman alabilecek bir fonksiyondur. Bunun anlamı şudur ki, fonksiyon bir kere çağrıldığında verilen parametre sayısı değişkeni kadar işleme değişken dahil edecektir.

*package* main  
  
*import* "fmt"  
  
*func* merhaba1(hello ...string) {  
 fmt.Println(hello)  
 fmt.Println(hello[2])  
}  
*func* sehirler(param ...string) {  
 *for* \_, sehir := *range* param {  
 fmt.Println(sehir)  
 }  
}  
*func* toplam(sayilar ...int) int {  
 sonuc := 0  
 *for* \_, sayi := *range* sayilar {  
 sonuc += sayi *//sonuc=sonuc+sayi* }  
 *return* sonuc  
}  
*func* main() {  
 */\*  
 merhaba1("Ahmet","Merve","Cem","Aslı","Ali")  
 \*/  
 /\*  
 sehirler("İstanbul","İzmir","Antalya","Aydın")  
 \*/* son := toplam(5, 6, 4, 3, 6, 7, 4, 3, 5, 6, 87, 98, 7)  
 fmt.Println(son)  
}

### Recursive (İç-içe) Fonksiyonlar

Recursive fonksiyonlar yazdığımız fonksiyonun içinde aynı fonksiyonu kullanmamız demektir. Fonksiyonumun tüm işlemler bittiğinde return olur.

*package* main  
  
*import* "fmt"  
  
*func* main() {  
 fmt.Println(faktoriyel(4))  
}  
  
*func* faktoriyel(a uint) uint {  
 *if* a == 0 {  
 *return* 1  
 }  
 *return* a \* faktoriyel(a-1)  
} Sonuç 24’tür.

2-

*package* main  
  
*import* "fmt"  
  
*func* recursive(){  
 fmt.Println("Merhaba")  
 recursive()  
}  
*func* faktoriyel(sayi int)int{  
 *if* sayi<=1{  
 *return* 1  
 }  
 *return* sayi\*faktoriyel(sayi-1)  
}  
*func* main(){  
 */\*  
 recursive() CVP:12 sayısının Faktoriyeli: 479001600  
 \*/* sayi:=12 *// 12x11x10....1* fmt.Printf("%d sayısının Faktoriyeli: %d",sayi,faktoriyel(sayi))  
}

### anonim fonksiyonlar

*package* main  
  
*import* "fmt"  
  
*func* abc(t int)int{  
 *return* t\*10  
}  
*func* main(){  
 any:=*func*(toplam int)int{  
 *return* toplam/3\*4\*5  
 }(10)  
 */\* diğer fonksiyonlardaki gibi  
 parametre verilmediği için fonksiyonun sonuna  
 parametre verilir.\*/* fmt.Println(any)  
 */\*Bizim anonim fonksiyonları kullanma amacımız  
 genellikle başka bir fonksiyonun içerisine argüman  
 olarak göndermek içindir.  
 \*/* fmt.Println(abc(any))  
} 60 - 600

2-

*package* main  
  
*import* "fmt"  
  
*func* main() {  
 metin := "Merhaba Dünya"  
 *func*(a string) {  
 fmt.Println(a)  
 }(metin)  
}

### defer

Defer kelimesinin Türkçe’deki karşılığı **ertelemektir**. Bu deyimi yapacağımız işlemin başına eklersek o işlemi içerisinde bulunduğu fonksiyondaki işlemlerden sonra çalıştırır. “Ertelenmiş İşlev”

*package* main  
  
*import* "fmt"  
  
*func* start(){  
 *defer* fmt.Println("PC AÇILDI")  
 fmt.Println("PC AÇILIYOR...")  
}  
*func* stop(){  
 *defer* fmt.Println("PC KAPANDI.")  
 fmt.Println("PC KAPANIYOR...")  
}  
*func* sira(){  
  
 *defer* fmt.Println("Birinci")  
 *defer* fmt.Println("İkinci")  
 *defer* fmt.Println("Üçüncü")  
  
 fmt.Println("Fonksiyon Başlıyor...")  
}  
*func* main(){  
 */\*  
 defer fmt.Println("Merhaba")  
 fmt.Println("Dünya")  
 \*/  
 /\*  
 start()  
 stop()  
 \*/* sira()

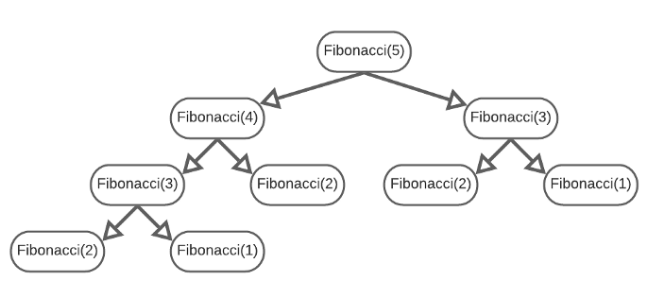
2-

*package* main  
*import* "fmt"  
*func* main() {  
  *defer* fmt.Println("İlk Cümle")  
  fmt.Println("İkinci Cümle")  
}

Çıktımız şu şekilde olacaktır;İkinci Cümle - İlk Cümle



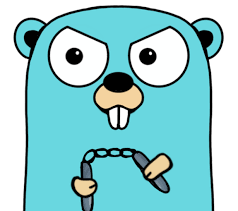
### fibonacci sayı serisi



Tanım olarak, Fibonacci dizisindeki ilk iki sayı, dizinin seçilen başlangıç ​​noktasına bağlı olarak ya 1 ve 1 ya da 0 ve 1'dir ve sonraki her sayı, önceki ikisinin toplamıdır.

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 …

*package* main  
  
*import* "fmt"  
  
*//Fibonnacci  
//0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, …  
//a b a b  
//f(n)=f(n-1)+f(n-2)  
func* fibo(n int) int {  
 a := 0  
 b := 1  
 *for* i := 0; i < n; i++ {  
 gecici := a  
 a = b  
 b = gecici + a  
 }  
 *return* a  
}  
*func* main() {  
  
 *for* n := 0; n <= 20; n++ {  
 sonuc := fibo(n)  
 fmt.Printf("Fibonacci:%d =%d \n", n, sonuc)  
 }  
}  
}



*package* main  
  
*import* "fmt"  
  
*//Fibonnacci  
  
//0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, …  
//a b a b  
//f(n)=f(n-1)+f(n-2)  
  
func* fibo(n int) int {  
 a := 0  
 b := 1  
 *for* i := 0; i < n; i++ {  
 gecici := a  
 a = b  
 b = gecici + a  
 }  
 *return* a  
}  
*func* main() {  
 *for* n := 0; n <= 20; n++ {  
 sonuc := fibo(n)  
 fmt.Printf("Fibonacci:%d =%d \n", n, sonuc)  
 }  
}

# STRUCT

Go programlama dilinde sınıflar yoktur. Sınıflar yerine struct'lar (yapılar) vardır. Yapılar sayesinde bir nesne oluşturabilir ve bu nesneye ait özellikler oluşturabiliriz.

Örnek:

*package* main  
  
*import* "fmt"  
  
*type* kişi *struct* {  
 isim string  
 soyİsim string  
 yaş int  
}  
*func* main() {  
 kişi1 := kişi{"Murat", "Boz", 23}  
 fmt.Println(kişi1)   
}

{Murat Boz 23}

kişi1 := kişi{}  
kişi1.isim, kişi1.soyİsim = "M. K.", "ATATÜRK"  
kişi1.yaş = 57  
  
fmt.Println(kişi1) *//{M. K. ATATÜRK 57}*

*package* main  
  
*import* "fmt"  
  
*//struct structure  
  
type* ogrenci *struct* {  
 *//field fieldtype* isim string  
 soyisim string  
 yas int  
 notlar []int  
 isMezunMu bool  
}  
  
*func* main() {  
  
 *var* o1 ogrenci *//var sayi int* o1.isim = "Deniz"  
 o1.soyisim = "Korkmazer"  
 o1.yas = 29  
 o1.notlar = []int{55, 66, 77}  
 o1.isMezunMu = true  
  
 fmt.Println(o1)  
  
 *//2. gösteriliş şekli* o2 := ogrenci{  
 isim: "Sercan",  
 *//soyisim:"Gürbüz",  
 //yas:28,* notlar: []int{80, 90, 95},  
 isMezunMu: false,  
 }  
  
 fmt.Println(o2)  
  
 o3 := ogrenci{"Ferhat", "Korkmaz", 49, []int{5, 6, 7}, false}  
  
 fmt.Printf("%v", o3)  
  
}



### ERİŞİLEBİLEN STRUCTLAR

package main  
  
import "fmt"  
  
type calisan struct{  
 isim string  
 yas int  
 maas int  
 isHaveKid bool //isMarried  
  
}  
  
type mudur struct{  
 calisan  
 unvan string  
  
}  
  
  
func main(){  
  
 a1:=calisan{  
 isim:"Ahmet",  
 yas:46,  
 maas:4250,  
 isHaveKid:true,  
 }  
  
 fmt.Println(a1)  
  
 y1:=mudur{  
 calisan:calisan{  
 isim:"Volkan",  
 yas:36,  
 maas:9750,  
 isHaveKid:false,  
  
 },  
 unvan:"Personel Muduru",  
 }  
  
 fmt.Println(y1)  
  
 kurucu:=struct{  
 isim string  
 sermaye int  
 }{isim:"Tahsin",sermaye:750000}  
  
 fmt.Println(kurucu)  
  
 fmt.Println(kurucu.sermaye)  
  
  
}

### İÇ İÇE STRUCT

GO dilinde tanımlanan bir struct başka bir struct içersinde kullanılabilimektedir.

package main  
  
import "fmt"  
  
type adres struct{  
 sehir string  
 ulke string  
}  
type kisi struct{  
 isim string  
 yas int  
 kisiadres adres  
}  
func main(){  
  
 k1:=kisi{  
 isim:"Ferhat Korkmaz",  
 yas:35,  
 kisiadres:adres{  
 sehir:"İzmir",  
 ulke:"Türkiye",  
 },  
 }  
 fmt.Println("İsim:",k1.isim)  
 fmt.Println("Yaş:",k1.yas)  
 fmt.Println("Şehir:",k1.kisiadres.sehir)  
 fmt.Println("Ülke",k1.kisiadres.ulke)  
}

### ANONİM STRUCT

GO dilinde struct’ların isim kullanılmadan tanımlandığı durumlar vardır.Bu tip tanımlamaya Anonymous(anonim)struct adı verilmektedir.

package main  
import "fmt"  
func main() {  
 kişi := struct {  
 ad, soyad string  
 }{"Kemal", "Atatürk"}  
 fmt.Println(kişi)  
}

Yukarıda struct’ı bir değişken olarak tanımladık.Bunu normal struct method olarak yazmaya kalksaydık aşağıdaki gibi yazardık.

package main  
import "fmt"  
type insan struct {  
 ad, soyad string  
}  
func main() {  
 kişi := insan{"Kemal", "Atatürk"}  
 fmt.Println(kişi)  
}

### METHOD(METOTLAR)

package main  
  
import "fmt"  
  
/\*  
  
func(alıcı\_adı Türü) fonksiyon\_adı(parametre\_listesi)(dönüş\_türü){  
 Kod  
}  
  
\*/  
  
type Ogrenci struct {  
 isim string  
 yas int  
 notlar []int  
 gectiMi bool  
}  
  
func (o Ogrenci) AdGoster() string {  
 return o.isim  
}  
  
func (ogr Ogrenci) BilgileriGoster() {  
 fmt.Println("Öğrenci Adı:", ogr.isim)  
 fmt.Println("Öğrenci Yas:", ogr.yas)  
 fmt.Println("Öğrenci Notlar:", ogr.notlar)  
 fmt.Println("Öğrenci Geçti Mi:", ogr.gectiMi)  
}

func (not Ogrenci) Ortalama() float64 {  
 toplam := 0  
 for \_, i := range not.notlar {  
 toplam += i  
 }  
  
 return float64(toplam) / float64(len(not.notlar))  
  
}  
  
func (gec Ogrenci) GectiMiKaldiMi() bool {  
 if gec.Ortalama() <= 50 {  
 return false  
 }  
 return true  
}  
func main() {  
 o1 := Ogrenci{"AHKS", 29, []int{1, 6, 7, 84, 95}, false}  
  
 fmt.Println(o1.AdGoster())  
  
 o1.BilgileriGoster()  
  
 fmt.Println(o1.Ortalama())  
  
 fmt.Println(o1.GectiMiKaldiMi())  
}

### KENDİ VERİ TİPİMİZ

package main  
  
import (  
 "fmt"  
 "time"  
)  
  
type ustunf float64  
  
func (y ustunf) CarpBes() ustunf {  
 return y \* 5.2  
}  
  
type Aylar int  
  
const (  
 Ocak Aylar = 1 + iota  
 Subat  
 Mart  
 Nisan  
 Mayis  
 Haziran  
 Temmuz  
 Agustos  
 Eylul  
 Ekim  
 Kasim  
 Aralik  
)  
  
// abdulhamid kaan bdlhmd kn  
  
func main() {  
  
 var u1 ustunf = 5.4  
  
 fmt.Printf("%T,%v\n", u1, u1)  
  
 var u2 float64 = 6.77  
  
 fmt.Println(u1 + ustunf(u2))  
  
 y1 := ustunf(50.5)  
  
 fmt.Println(y1.CarpBes())  
  
 \_, ay, \_ := time.Now().Date()  
 fmt.Println(ay)  
  
 fmt.Println(Aylar(ay))  
  
 fmt.Println(Aylar(Nisan))  
  
 fmt.Printf("%T,%v", Aylar(ay), Aylar(ay))  
  
}

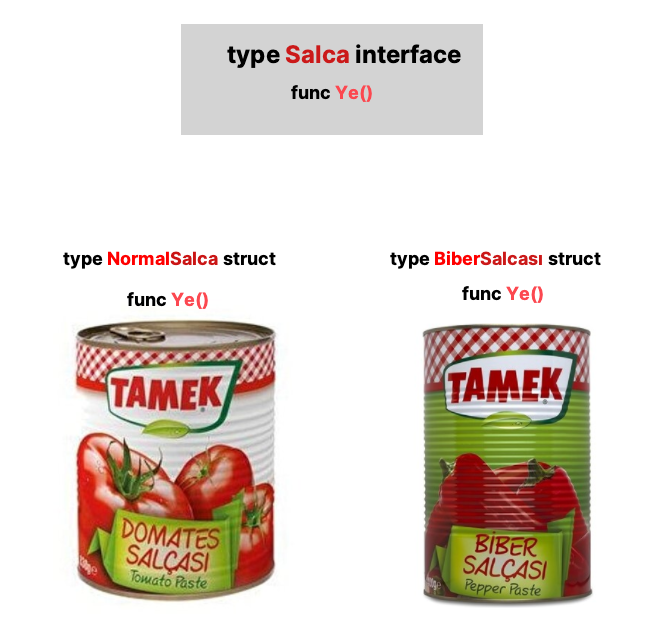
# interface

GO programlama dilinde arayüz;davranışları tanımlamak için kullanılacak metot imzaları kümesinden oluşan bir şablon olarak tanımlanabilir.

package main  
  
import "fmt"  
/\*  
type interface\_adi interface{  
 // Method imzaları  
 func1() int  
 func2() float64  
}  
 \*/  
/\*  
type A struct{  
}  
func(x A)Read(){  
}  
type B struct{  
}  
func(y B)Read(){  
}  
  
func callRead(a A){  
 a.Read()  
}  
  
func callRead2(b B){  
 b.Read()  
}  
 \*/  
type yuruyen interface{  
 Yuru() string  
}  
type Insan string  
type Kedi string  
  
func(i Insan) Yuru() string{  
 return "Ben bir insanım ve yürüyorum"  
}  
func(k Kedi) Yuru() string{  
 return "Ben bir kediyim ve yürüyorum"  
}  
//Interface'den oluşturulan Func  
func yurume(y yuruyen){  
 fmt.Println(y.Yuru())  
}  
func main(){  
 var insan Insan  
 var kedi Kedi  
  
 yurume(insan)  
 yurume(kedi)  
  
}

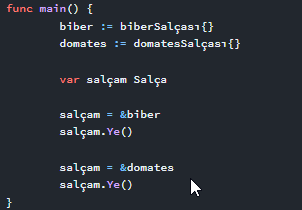
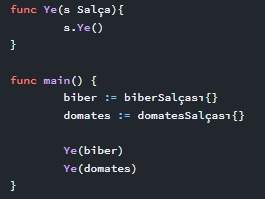
2-

package main  
  
import "fmt"  
  
const PI=3.14  
  
type hesapla interface{  
 alan() float64  
 cevre() float64  
}  
  
type daire struct{  
 yaricap float64  
}  
  
type dikdortgen struct{  
 kisakenar float64  
 uzunkenar float64  
}  
  
func(d daire) alan() float64{  
 return PI\*d.yaricap\*d.yaricap  
}  
  
func(g dikdortgen) alan() float64{  
 return g.kisakenar\*g.uzunkenar  
}  
  
func(d daire) cevre()float64{  
 return d.yaricap\*PI\*2.0  
}  
  
func(g dikdortgen) cevre()float64{  
 return (g.kisakenar+g.uzunkenar)\*2  
}  
  
  
//Interface'den oluşturduğum func  
func hesapYap(h hesapla){  
 fmt.Println(h)  
 fmt.Println(h.alan())  
 fmt.Println(h.cevre())  
  
 fmt.Printf("Veri Tipi:%T\n",h)  
}  
  
  
  
func main(){  
  
 daire1:=daire{5.0}  
 dikdortgen1:=dikdortgen{6.0,9.0}  
  
  
 hesapYap(daire1)  
 hesapYap(dikdortgen1)  
}

Yandaki ilginç şemadan çıkarmamız gereken ders, altta bulunana konserve salçaların ikisinde de ye() isminde fonksiyonları var. Bunları struct fonkisyon olarak eklenmiş gibi düşünün.

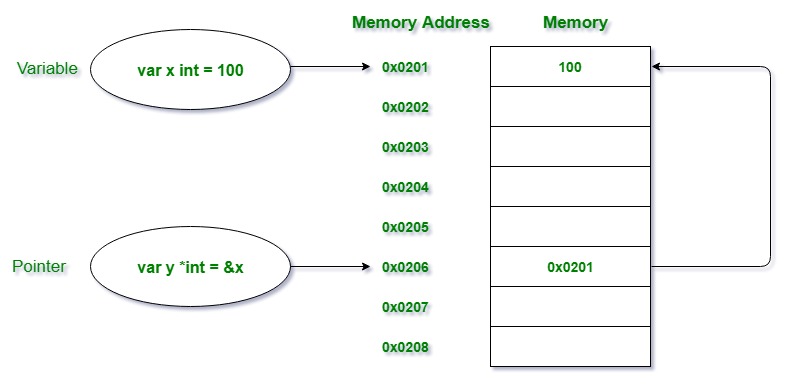
Yukarıdaki Salça interface'imizin ise ye() isminde bir fonksiyon tanımlaması var. Aslında burada struct fonksiyonlar gibi bir fonksiyon tanımlamıyoruz. Burada anlatılmak istenen olay, salça interface'i içerisinde bir salçanın salça olabilmesi için ye() fonksiyonunun olması gerekir diyoruz. Yani interface'imiz bir bakıma Türk Standartları Enstitüsü gibi birşey

Yani biz bir salçalı ekmek yiyeceğimiz zaman Salça interface'imize bakıyoruz ve görüyoruz ki TSE tarafından belirlenmiş olarak ye() fonksiyonu bulunan bir salça almamız gerekiyor. Şemadaki 2 çeşit salçanın da ye() fonksiyonu olduğu için ikisi de salça interface'inde belirlenmiş standart(lar)a uygun oluyor.


# POİNTER(İŞARETÇİLER)

GO programlama dilinde pointer(işaretçi) değişkenler bulunmaktadır . İşaretçiler bir başka değişkenin hafızada tutulduğu adres bilgisini içermektedir.İşaret ettikleri değişkenin içeriğine bakılmaksızın 32 bit makinelerde 4 byte,64 bit makinelerde 8 byte yer tutarlar.





Yukarıdaki çizimde, b değişkeni 156 değerine sahiptir ve 0x1040a124 bellek adresinde depolanmıştır. a değişkeni b'nin adresini tutar. Şimdi a'nın b'yi işaret ettiği söyleniyor.

### Declaring pointers

### **T , T** türünde bir değere işaret eden işaretçi değişkeninin türüdür .

package main  
  
import (  
 "fmt"  
)  
  
func main() {  
 b := 255  
 var a \*int = &b  
 fmt.Printf("Type of a is %T\n", a)  
 fmt.Println("address of b is", a)  
}

package main  
  
import (  
 "fmt"  
)  
  
func main() {  
 b := 255  
 var a \*int = &b  
 fmt.Printf("Type of a is %T\n", a)  
 fmt.Println("address of b is", a)  
}

address of b is 0x1040a124

value of b is 255

\*\*İŞARETÇİLERİN DEFAULT(VARSAYILAN)DEĞERLERİ\*\*

İşaretçiler başlangıç değeri olmadan da tanımlayabilmektedir. Bu durumda “nil” değerini alırlar.

package main  
  
import "fmt"  
  
func main(){  
 //var x int=10  
 x:=10  
 fmt.Println(x)  
 fmt.Println(&x)  
 p:=&x  
 //var p \*int=&x  
 fmt.Println(p)  
 // \*  
 fmt.Println(\*p)  
  
 \*p=44  
  
 fmt.Println(x)  
  
 fmt.Printf("%T,%v\n",x,x)  
 fmt.Printf("%T,%v\n",p,p)  
  
 var z string="Merhaba"  
 var pz \*string=&z  
  
 fmt.Println(z,pz)  
 //Google Etkisi  
 //v2 rocket  
 //2.dünya savaşı,alman rocket  
}-Pointer-

Type of a is \*int

address of b is 0xc000018098

### Bir işleve işaretçi geçirme

package main  
  
import (  
 "fmt"  
)  
  
func change(val \*int) {  
 \*val = 55  
}  
func main() {  
 a := 58  
 fmt.Println("value of a before function call is",a)  
 b := &a  
 change(b)  
 fmt.Println("value of a after function call is", a)  
}

value of a before function call is 58

value of a after function call is 55

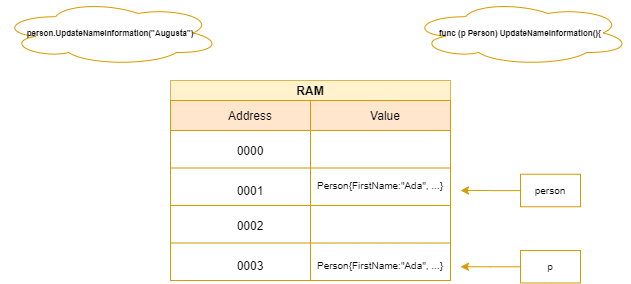
### POİNTERLARIN FONKSİYONLAR İLE KULLANIMI

GO dilinde kullanılan fonksiyonlarda parametre olarak pointer’lar kullanılabilmektedir.

package main  
  
import "fmt"  
  
func isaret(ptr \*int){  
 \*ptr=55 //dereferencing  
}  
  
func onlacarp(sayi \*int) int{  
 //\*sayi\*=10  
 //fmt.Println(\*sayi)  
  
 return \*sayi\*10  
}  
func main(){  
  
 x:=100  
  
 isaret(&x)  
  
 fmt.Println(x)  
  
 //onlacarp(&x)  
  
 fmt.Println(onlacarp(&x))

### POİNTER İLE STRUCT’LAR ve metotlar ile kullanımı

Struct tanımlamak için, isminin önüne *type*ve sonuna da *struct* keyword’lerini eklememiz gerekir. Bir struct int, string türündeki alanlardan oluşabileceği gibi içerisinde farklı türde struct alanlar da bulundurabilir.



package main  
  
import "fmt"  
  
type Vatandas struct{  
 isim string  
 tcno int  
}  
  
func (v Vatandas) AyarlaValue(){ //method on value  
 v.isim="Meral"  
 v.tcno=444555  
}  
  
func (v \*Vatandas) AyarlaPointer(){//method on pointer  
 v.isim="Recep"  
 v.tcno=999999  
}  
  
  
func main(){  
  
 vat1:=Vatandas{"Cem Uzan",123456}  
  
 var ptr \*Vatandas=&vat1  
  
 fmt.Println(ptr)  
  
 fmt.Println(&(ptr.isim))  
 fmt.Println(&(ptr.tcno))  
  
 fmt.Println(\*ptr)  
  
 ptr.isim="Ahmet"  
  
 fmt.Println(vat1)  
  
 fmt.Println()  
  
 ptr.AyarlaValue()  
 fmt.Println(vat1)  
  
 ptr.AyarlaPointer()  
 fmt.Println(vat1)



# PACKAGE(PACKAGES)

### TİME PAKETİ

package main  
  
import (  
 "fmt"  
 "time"  
)  
func main(){  
 //Şuanki Zaman  
 zaman:=time.Now()  
 fmt.Println(zaman)  
 fmt.Println(zaman.Day())  
 fmt.Println(zaman.Month())  
 fmt.Println(zaman.Year()) Tek tek ve alt alta yazacaktır sırası ile   
 fmt.Println()  
 fmt.Println(zaman.Hour())  
 fmt.Println(zaman.Minute())  
 fmt.Println(zaman.Second())  
  
 zamanstring:=zaman.String() Bir satırda tüm verileri yazacaktır detaylı olarak.  
 fmt.Println(zamanstring)  
 //sleep  
 //fmt.Println("Merhaba")  
 //time.Sleep(time.Second\*5) İlk merhaba yazdıracak ve 5 saniye sonra   
 //fmt.Println("Merhaba") tekrar merhaba yazdıracaktır   
 //UTC  
 zamanUTC:=time.Now().UTC() Farklı bir ülkenin zaman dilimini yazdıracaktır  
 fmt.Println(zamanUTC) seçtiğimiz yere göre Örn(UTC)  
  
 //Date Kendi Tarihimizi Oluşturmak  
 dogumgunu:=time.Date(1992,11,30,12,30,0,0,time.Local)  
  
 fmt.Println(dogumgunu) 1 tanesine %v yerine %d yazarsak bize 30-11-1992 yazacaktır(int)  
 fmt.Printf("%v-%v-%v\n",dogumgunu.Day(),dogumgunu.Month(),dogumgunu.Year())  
 //2 tarih arasındaki fark  
 baslangic:=time.Date(1915,3,18,0,0,0,0,time.UTC)  
 bugun:=time.Now().UTC()  
  
 fark:=bugun.Sub(baslangic).Hours()/24  
  
 fmt.Println(fark)  
  
 kacgunoldu:=bugun.Sub(dogumgunu).Hours()/24  
  
 fmt.Println(kacgunoldu)  
  
 var d time.Duration=1000000000  
  
 fmt.Println(d.Hours())  
 //Tarih Ekleme  
 suankizaman:=time.Now()  
 yarin:=suankizaman.AddDate(15,3,1) 2 gün sonrasının verilerini bizlere yazacaktır.  
 fmt.Println(yarin)  
 ikigunsonra:=suankizaman.Add(time.Hour\*48)  
  
 fmt.Println(ikigunsonra)  
 //Kendimize Özel Zaman Biçimi  
 hafta:=time.Hour\*7\*24  
  
 fmt.Println(hafta)  
}

### MATH PAKETİ

Bu paket temel matematiksel sabitleri ve fonksiyonları sağlamaktadır.Bu başlık altında çok kullanılan sabitler ve fonksiyonlar hakkında bilgi verilmeye çalışılacaktır <> En çok kullanılanlar aşağıda belirtilmiştir.

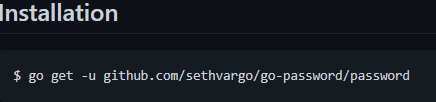
package main  
  
import (  
 "fmt"  
 "math"  
 "math/rand"  
 "time"  
)  
func main(){  
  
 Pisayisi:=math.Pi Pisayisi yazılan değişkeni math.Pi(paketi) besliyor  
  
 fmt.Println(Pisayisi)  
  
 fmt.Println(math.Pow(2,10))  
  
 fmt.Println(math.Sqrt(64))  
  
 fmt.Println(math.Pow(64,0.5))  
  
 //Mutlak Değer Absolute Value-Float  
 fmt.Println(math.Abs(-55.4))  
  
 fmt.Println(math.Sin(37))  
 fmt.Println(math.Cos(57))  
  
  
 fmt.Println(math.Mod(44,3))  
  
 //Yuvarlama  
 fmt.Println(math.Round(22.5))  
  
 //Tavan-- Yukarı Doğru Yuvarlama  
 fmt.Println(math.Ceil(22.1))  
  
 //Taban--Aşağı Doğru Yuvarlama  
 fmt.Println(math.Floor(22.9))  
  
 //Max-Min  
 fmt.Println(math.Max(55,2))  
 fmt.Println(math.Min(67,43))  
  
 //Sayı Kontrol   
 fmt.Println(math.IsNaN(400)) /IsNaN=Sayimi demektir eğer sayı ise false   
 yazacaktır sayi değil ise true  
 fmt.Println(math.Log(100))  
 fmt.Println(math.Log10(100))  
  
 rand.Seed(time.Now().UnixNano())  
 Herhangi bir sayı verecektir(rastgele sayı)  
 rastgele:=rand.Intn(100)  
 fmt.Println(rastgele)  
}

### STRİNGS PAKETİ

package main  
  
import (  
 "fmt"  
 "strings"  
)   
func main() { İşlemi küçük harfe çevirecektir  
  
 fmt.Println(strings.ToLower("DENİZ KORKMAZER"))  
 fmt.Println(strings.ToUpper("deniz korkmazer"))  
 İşlemi büyük harfe çevirecektir  
 //Index  
 fmt.Println(strings.Index("Turkiye", "r"))  
  
 //Count  
 fmt.Println(strings.Count("İzmir", "a"))  
 fmt.Println(strings.Count("İzmir", ""))  
 //Contains – Yanlış yazılan örn:mail adresini bizi true false olarak söyler  
 fmt.Println(strings.Contains("info@gmail.com", "@"))  
 //Compare/a=b return=0 if a<b return=-1 > ise +1 olarak verecektir  
 k1 := "Merhaba" Yazılanlar eşitmi diye bakmaktadır  
 k2 := "Merhaba Dunya"  
 k3 := "Merhaba"  
  
 fmt.Println(strings.Compare(k1, k2))  
 fmt.Println(strings.Compare(k2, k3))  
 fmt.Println(strings.Compare(k3, k1))  
 //Return 0, if str1 == str2  
 //Return 1, if str1 > str2  
 //Return -1, if str1 < str2  
  
 //Replace / Yazılan bir cümlenin harflerini değiştirme(yazım yanlışı düzeltmek   
 için kullanılabilir.  
 fmt.Println(strings.Replace("Olmak yadak Olmamak", "k", "z", 2))  
  
 //ReplaceAll  
  
 fmt.Println(strings.ReplaceAll("Hey Hey Hey", "Hey", "Mey"))  
  
 //Fields Alan şekili almış olur (Boşluk düzeltme)  
  
 data := " username password email date"  
 fields := strings.Fields(data)  
 fmt.Printf("%q\n", fields)  
 //Split  
 fmt.Println(strings.Split("a,b,c,d", ",")) 🡪Yazdığımıza göre arada ki“virgülü”  
 fmt.Println(strings.Split("19-11-1987", "-")) parçalayarak ayrı ayrı yazacak  
 //Split Slice  
 tarih := "19-11-1879"  
 ayir := strings.Split(tarih, "-")  
 fmt.Println(ayir[2])

//SplitAfter  
 sira := "a,b,c,d"  
 ayri := strings.SplitAfter(sira, ",") 🡪 Virgül ile beraber bölmekte  
 fmt.Println(ayri)  
 fmt.Println(ayri[1])  
}

### GİTHUB PAKET

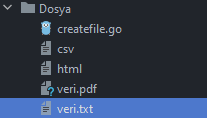


CMD KONSOLUNA YAZDIKTAN SONRA İNDİYOR/!!Dikkatli olunması gereken yer GO’nun nereye kurulu olduğudur.Paketi kullanmak için ise “"github.com/sethvargo/go-password/password"ı kopyala yapıştır yapmamız yeterlidir daha sonra işlemimizi yapmaya devam edebiliriz.

package main  
  
import (  
 "log"  
  
 "github.com/sethvargo/go-password/password"  
)  
  
func main() {  
 // Generate a password that is 64 characters long with 10 digits, 10 symbols,  
 // allowing upper and lower case letters, disallowing repeat characters.  
 res, err := password.Generate(21, 10, 10, false, false)  
 if err != nil {  
 log.Fatal(err)  
 }  
 log.Printf(res)  
}

# DOSYA İŞLEMLERİ

package main  
  
import (  
 "log"  
 "os"  
)  
  
func main() {   
 “.../veri.”pdf,txt,html,csv” farklı klosöre atar  
 dosya, err := os.Create("csv")🡪”pdf,txt,html” gibi formatları değiştirebiliriz  
 if err != nil {  
 log.Fatal(err)  
 }  
 dosya.Close()  
}



### dosya açmak ve içine veri yazdırmak

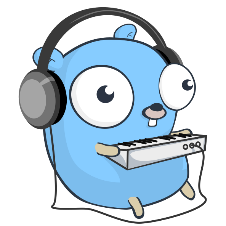
package main  
  
import (  
 "log"  
 "os"  
)  
  
//O\_RDONLY SADECE OKUMA  
//O\_WRONLY SADECE YAZMA  
//O\_RDWR HEM OKUMA HEM YAZMA  
//O\_APPEND DOSYA SONUNA EKLEME  
//O\_CREATE DOSYA OLUŞTURMA  
//O\_EXCL DOSYA YOKSA OLUŞTUR O\_CREATE İLE KULLANILIR  
//O\_SYNC I/O İşlemleri için Senkron açılır  
//O\_TRUNC DOSYA SONUNU KESER  
  
//0000 İzin yok  
//0700 Yönetici yazabilir,okuyabilir ve oluşturabilir  
//0770 Yönetici ve Grup yazabilir,okuyabilir ve oluşturabilir  
//0777 Herkes yazabilir,okuyabilir ve oluşturabilir  
//0111 Oluşturma  
//0222 Yazma  
//0333 Yazma ve Oluşturma  
//0444 Okuma  
//0555 Okuma ve Oluşturma  
//0666 Yazma ve Okuma (!)  
//0740 Yönetici yazabilir,okuyabilir ve oluşturabilir; Grup Sadece Okuyabilir; Diğerlerinin İzni Yok  
  
//Birden fazla şekilde de kullanılabilir O\_WRONLY|O\_RDONLY  
func main(){  
  
 dosya,err:=os.OpenFile("veri.txt",os.O\_WRONLY,0000)  
 defer dosya.Close()  
 if err !=nil{  
 log.Fatal(err)  
 }  
 //fmt.Println("Dosya Açıldı")  
  
 //dosya.WriteString("Abdulhamid Kaan")  
  
 //yaziSlice:=[]byte("Merhabalar Udemy ")  
 //  
 //dosyaYazma,err:=dosya.Write(yaziSlice)  
 //if err !=nil{  
 // log.Fatal(err)  
 //}  
 //  
 //fmt.Printf("Dosyaya %d Byte Boyutunda Yazı Yazıldı",dosyaYazma)  
 sehirler:=[]string{"Malatya","Ankara","Çanakkale","İzmir","Kütahya"}  
  
 for \_,sehir:=range sehirler{  
 \_,err:=dosya.WriteString(sehir +"\n")  
 if err !=nil{  
 log.Fatal(err)  
 }  
 }  
} 🡪veri.txt dosyasına yazdırıyor

### DOSYA TAŞINMAK VE İSİM YENİLENDİRMEK

package main  
  
import (  
 "log"  
 "os"  
)  
  
func main() {  
  
 //eskiAd:="veri.txt"   
 //yeniAd:="data.txt" Bir dosyanın ismini değiştirmek için  
 //  
 //os.Rename(eskiAd,yeniAd)  
  
 dosyaKonum:="data.txt"  
 yeniKonum:="/veri.txt"  
 Bir dosyayı bir klasörün içine   
 err:=os.Rename(dosyaKonum,yeniKonum) taşımak için  
 if err !=nil{  
 log.Fatal(err)  
 }

### DOSYA KOPYALAMAK

package main  
  
import (  
 "fmt"  
 "io/ioutil"  
 "log"  
 "os"  
)  
  
func main() {  
  
 kaynak,err:=ioutil.ReadFile("veri.txt")  
 if err !=nil{  
 log.Fatal(err)  
 }  
  
 yeniDosya,err2:=os.Create("yenidosya.txt")  
 if err2 !=nil{  
 log.Fatal(err2)  
 }  
  
 err3:=ioutil.WriteFile("yenidosya.txt",kaynak,0666)  
 if err3 !=nil{  
 fmt.Println("Oluştururken hata oluştu",err3)  
 }  
  
 defer yeniDosya.Close()  
}



### DOSYA SİLME

package main  
  
import (  
 "log"  
 "os"  
)  
func main(){  
 //os.Remove("veri.txt")🡪 Silme işlemi için sadece bu yeterlidir  
 err:=os.Remove("Dosya/veri.html")  
 if err !=nil{  
 log.Fatal(err)  
 } 🡪 Bu farklı bir dosyayı silmek içindir  
}

### DOSYA BİLGİLERİ

package main  
import (  
 "fmt"  
 "log"  
 "os"  
)  
func main() {  
 dosyaBilgi,err:=os.Stat("veri.txt")  
 if err !=nil{  
 log.Fatal(err)  
 }  
 fmt.Println("Dosya Adı:",dosyaBilgi.Name())  
 fmt.Println("Dosya İzinleri:",dosyaBilgi.Mode())  
 fmt.Println("Dosya Boyutu:",dosyaBilgi.Size())  
 fmt.Println("Değiştirme Tarihi:",dosyaBilgi.ModTime())  
 fmt.Println("Klasör Mü?",dosyaBilgi.IsDir())  
 fmt.Println("Sistem Interface",dosyaBilgi.Sys())  
  
 if dosyaBilgi.Size()<90{  
 os.Remove("veri.txt")  
 }  
}

### DOSYA KONTROL

package main  
import (  
 "fmt"  
 "os"  
)  
func main(){  
 \_,err:=os.Stat("yeni.txt")  
 if err !=nil{  
 fmt.Println("HATA:",err)  
 } Mevcut değil mi ?   
 dosyaKontrol:=os.IsNotExist(err)  
 fmt.Println(dosyaKontrol)  
 if dosyaKontrol==true{  
 os.Create("veri.txt")  
 } Yeni klasör oluştur(“Yeni isim”)  
}

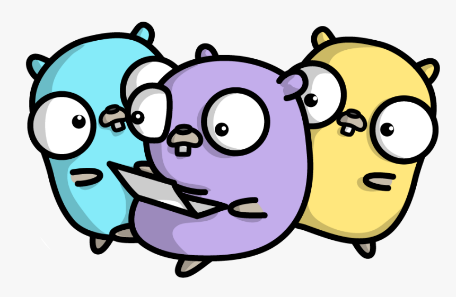
### DOSYA OKUMA İŞLEMİ

package main  
  
import (  
 "fmt"  
 "io/ioutil"  
 "log"  
)  
func main(){  
  
 dosyaOkuma,err:=ioutil.ReadFile("ulkeler.txt")  
 if err !=nil{  
 log.Fatal(err)  
 }  
 //fmt.Println(string(dosyaOkuma)) 🡪Ülke ülke alt alta yazacaktır

for i,\_:=range dosyaOkuma{ 🡪Harf harf alt alta yazacaktır  
 fmt.Println(string(dosyaOkuma[i])) 🡪  
 }  
 //dosya,err:=os.Open("ulkeler.txt")  
 //defer dosya.Close()  
 //if err !=nil{  
 // log.Fatal(err)  
 //}  
 //  
 //scanner:=bufio.NewScanner(dosya)  
 //  
 //for scanner.Scan(){  
 // satir:=scanner.Text()  
 // time.Sleep(2\*time.Second) 🡪Her ülke 2 saniye sonra ekrana düşecektir  
 // fmt.Println(satir)  
 //  
 //}  
}

### KLASÖR OLUŞTURMA

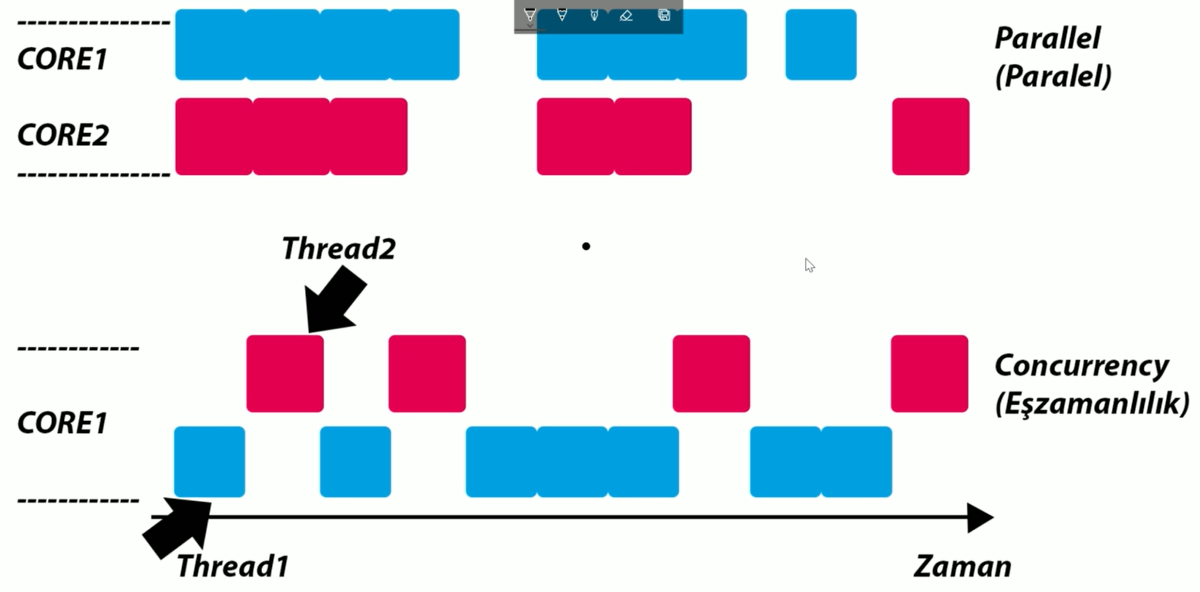
package main  
  
import (  
 "log"  
 "os"  
)  
  
func main(){  
 (name:”b”,os.ModePerm)🡪B klasörü oluştur.  
 err:=os.Mkdir("../xml",os.ModePerm)  
 🡪Geriye git ve xml adlı bir klasör oluştur  
 if err !=nil{  
 log.Fatal(err)  
 }



### GEÇİCİ KLASÖR VE GEÇİCİ DOSYA

package main  
  
import (  
 "fmt"  
 "log"  
 "os"  
)  
  
func main(){  
  
 geciciKlasor,err:=os.MkdirTemp("","udemy")  
 if err !=nil{  
 log.Fatal(err)  
 }  
  
 fmt.Println("Gecici Klasor:",geciciKlasor)  
  
 defer os.RemoveAll(geciciKlasor)  
  
 geciciDosya,err:=os.CreateTemp(geciciKlasor,"udemykursu")  
 if err !=nil{  
 log.Fatal(err)  
 }  
  
 fmt.Println("Gecici Dosya:",geciciDosya.Name())  
  
 defer os.Remove(geciciDosya.Name())  
}

# concurrency



# GOROUTİNE

**Goroutine**’ler **Go Runtime** tarafından yönetilen hafif bir sistemdir. Bir işlemi eşzamanlı olarak yapmak istiyorsak, Goroutine'den faydalanabiliriz. Bu sayede aynı çalışma-zamanı içerisinde birden fazla iş parçacığı oluşturabiliriz.

### TERİMLER

ANA İŞ PARÇACIĞI

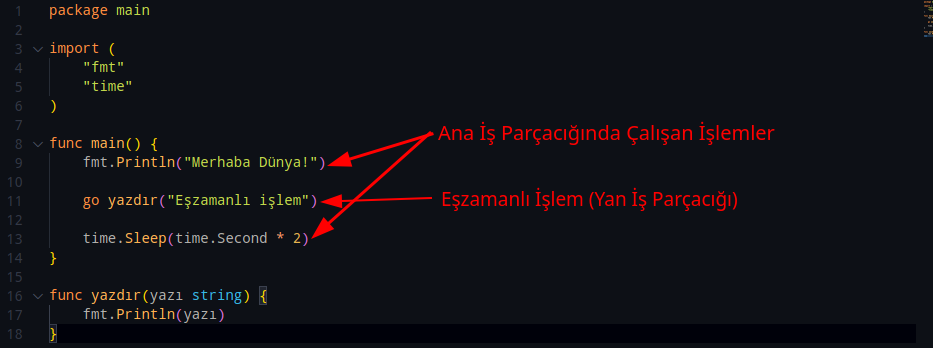
Main() fonksiyonu içerisine yazdığımız, asenkron olmayan kodlardır. Varsayılan olarak Go Runtime bu iş parçacığını izler. Programımız asenkron işlemlerin tamamlanmasını beklemiyorsa, ana iş parçacığı tamamlandığında program sona erer.

EŞZAMANLILIK

Eşzamanlılık, programlamada bir işlem gerçekleşirken, aynı zamanda başka işlemlerin de gerçekleşmesidir.

EŞ ZAMANLI BİR İŞLEM OLUŞTURALIM

Eşzamanlı bir işlem oluşturmak için go anahtar kelimesinden faydalanabiliriz. Bunun için eşzamanlı çalışacak işlemin başına go yazmamız yeterli olacaktır.



package main  
  
import (  
 "fmt"  
 "time"  
)  
func yazdir(s string){  
 for i:=0;i<5;i++{  
 fmt.Println(s)  
 }  
}  
func main(){  
 //Thread 1 MB Goroutine 2KB 🡪 Gorutinlerde çakışma hatasını çözmek için time   
 eklemek gereklidir böylelikle sorun çözlr  
 go yazdir("Merhaba")  
 time.Sleep(1\*time.Second)  
 yazdir("Dunya")  
  
}

### WAİTGROUP

Goroutine’leri Asenkron programlama yaparken kullanırız. Böylece aynı anda birden fazla işlem gerçekleştirebiliriz. Peki programımızın belirttiğimiz asenkron işlemleri bekleme gibi bir ihtiyacı olsaydı, ne yapmamız gerekirdi? Bu durumlarda WaitGroup'lardan faydalanabiliriz. Örneğin projemizde 3 adet asenkron işlem bulunuyorsa, WaitGroup'a 3 değerini ekleriz. Her asenkron işlem tamamlandığında WaitGroup -1 azalır ve sıfıra geldiğinde WaitGroup tamamlanmış olur. WaitGroup'u kullanmak için ise "sync" paketini projemize dahil ediyoruz. Kodlar üzerinde açıklamasını görelim.

package main  
  
import (  
 "fmt"  
 "sync" //WaitGroup'u kullammak için  
 "time" //bekleme işlemleri için  
)  
/\*  
\* waitgroup nesnesini işaretçi olarak parametre veriyoruz.  
\* işaretçi olarak vermemizin sebebi, programın bekleme işlemi için  
\* asıl waitgroup nesnesini kontrol etmesidir.  
 \*/  
func fonksiyon1(wg \*sync.WaitGroup) {  
 //fonksiyonun 2 sn beklemesini istiyoruz.  
 time.Sleep(2 \* time.Second)  
 fmt.Println("Fonk1 tamamlandı")  
 //wg.Done() fonksiyonu ile waitgroup nesnesini -1 azalttık.  
 wg.Done()  
}  
//bu fonksiyonumuza da wg nesnesini işaretçi ile parametre olarak verdik.  
func fonksiyon2(wg \*sync.WaitGroup) {  
 //fonksiyonu 3 sn uyuttuk.  
 time.Sleep(3 \* time.Second)  
 fmt.Println("Fonk2 tamamlandı")  
 //-1 daha eksilttik.  
 wg.Done()  
}  
func main() {  
 /\*  
 \* Öncelikle waitgroup'u kullanabilmek için bir waitgroup  
 \* nesnesi oluşturuyoruz.  
 \*/  
 var wg sync.WaitGroup  
 \* waitgroup'a 2 ekliyoruz. Yani 2 tane işlemden yanıt gelmesini  
 \* beklemesini istiyoruz. Aslında burada beklemeyecek. Sadece  
 \* işlem sayısını belirttik.  
 wg.Add(2)  
 \* fonksiyon1 ve fonksiyon2'ye oluşturduğumuz wg örneğinin  
 \* bellekteki adresinin veriyoruz.  
 go fonksiyon1(&wg)  
 go fonksiyon2(&wg)  
 fmt.Println("Merhaba Dünya!")  
 \* Burada wg.Wait() fonksiyonu ile asenkron işlemleri beklemesini  
 \* sağlıyoruz. yani waitgroup'un 0'a düşmesini bekliyoruz.  
 \* Eğer waitgroup olmadan yapsaydık. asenkron fonksiyonlarımızın tamamlanmasını  
 \* beklemeden program kendini sonlandırırdı.  
 wg.Wait()  
 //waitgroup tamamlandığında ekrana yazı bastıralım.  
 fmt.Println("WaitGroup'lar tamamlandı.")  
}

##### WaitGroup2

package main  
  
import (  
 "fmt"  
 "sync"  
)  
func main() {  
 wg := sync.WaitGroup{}  
 wg.Add(1)  
  
 go func() {  
 for i := 0; i < 5; i++ {  
 fmt.Println("Goroutine")  
 }  
 wg.Done()  
 }()  
 wg.Wait()  
  
 fmt.Println("Ana Thread")  
}

### KANALLAR(CHANNELS)

**Kanallar**, Go dilinde asenkron programlama yaparken değer aktarımı yapabileceğimiz hatlardır. Kanala değer atanması iş parçacığı tarafından bekleneceği için asenkron işlemler arasındaki senkronizasyonu ayarlayabiliriz. Kanallar make() fonksiyonu ile oluşturulur.

package main  
  
import (  
 "time"  
)  
  
func main() {  
 //bir kanal oluşturalım  
 k := make(chan bool)  
 //bu kanalımız bool değer taşıyacak  
 //asenkron bir iş parçacığı oluşturalım  
 go func() {  
 //bu iş parçacığı 5 sn beklesin  
 time.Sleep(time.Second \* 5)  
 //k kanalına bool bir değer gönderelim  
 k <- true  
 }()  
 //ana iş parçacığı k kanalına değer gelene kadar bekleyecek  
 <-k  
 //değer geldiğinde program sonlanacaktır.  
}

package main  
  
import (  
 "fmt"  
 "time"  
)  
func main(){  
 //k:=make(chan string)  
 //  
 //go func(){  
 //  
 // k<-"kanaldan gelen veri"  
 //}()  
 //  
 //mesaj:=<-k  
 //  
 //fmt.Println(mesaj)  
 //birden fazla veri taşıyan kanal oluşturma  
 //superK:=make(chan bool,3)  
 //  
 //go func(){  
 // superK<-true  
 //  
 // time.Sleep(1\*time.Second)  
 //  
 // superK<-false  
 //  
 // time.Sleep(1\*time.Second)  
 //  
 // superK<-true  
 //}()  
 //  
 //fmt.Println(<-superK,<-superK,<-superK)

sehirler:=[]string{"Malatya","Kayseri","Ordu","Rize"}  
  
 sehirKanal:=make(chan string)  
  
  
 //Gönderen Yapı  
 go func(dizi []string){  
 for \_,sehir:=range dizi{  
 sehirKanal<-sehir  
 }  
 }(sehirler)  
  
  
 go func(){  
 for i:=0;i<len(sehirler);i++{  
 alinan:=<-sehirKanal  
 fmt.Println("Kanaldan Gelenler",alinan)  
  
 }  
 }()  
  
 <-time.After(time.Second\*3)  
}

### Boyutlu kanal oluşturma

Oluşturduğumuz kanala boyut vermek de mümkün. Yani kanalımıza birden fazla değer yollayabiliyoruz. Bunun için kanalı oluştururken make() fonksiyonunda boyutu da belirtelim.

Örnek:

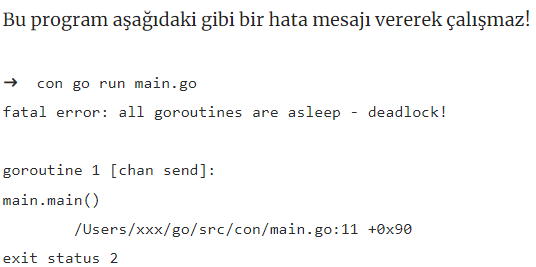
package main  
  
import (  
 "fmt"  
 "time"  
)  
  
func main() {  
  
 //2 adet bool değer taşıyan bir kanal oluşturalım  
 k := make(chan bool, 2)  
  
  
 //asenkron bir iş parçacığı oluşturalım  
 go func() {  
  
 //5 sn beklesin  
 time.Sleep(time.Second \* 5)  
  
 //k kanalına bool bir değer gönderelim  
 k <- true  
  
 //tekrardan 2 sn beklesin  
 time.Sleep(time.Second \* 2)  
  
 //ve k kanalına 2. değer de gönderilsin.  
 k <- false  
 }()  
  
 //ana iş parçacığı k kanalına 2 değer gelene kadar bekleyecek  
 fmt.Println(<-k, <-k) //çıktı: true false  
 //iki bool değeri de baştırmak için k kanalını 2 defa yazdık  
}

### DEADLOCK

Go channels denilince akla gelen en yaygın problem deadlock hatasıdır. Özetlemek gerekirse bir kanala gönderen (sender) kadar okuyucu (reader) atanmaz ise, diğer bir deyişle kapasitesinden fazla veri gönderilirse programımız deadlock problemi ile runtime hatası vererek derlenmez. Tabi bir go rout Örnek bir deadlock hatası şöyle bir programda ortaya çıkabilir.

func main() {  
kanalim := make(chan string, 2)  
kanalim <- "Beşiktaş"  
kanalim <- "Galatasaray"  
kanalim <- "Fenerbahçe"  
fmt.Println(<-kanalim)  
fmt.Println(<-kanalim)  
}

package main  
  
import "fmt"  
  
func main() {  
  
 //ch:=make(chan int,2)  
 //  
 //ch<-1  
 //  
 //fmt.Println(<-ch)  
  
 kanal := make(chan string, 5)  
  
 kanal <- "RENAULT"  
 kanal <- "VOLKSWAGEN"  
 kanal <- "TOYOTA"  
 kanal <- "DACİA"  
  
 fmt.Printf("Kanal Kapasitesi:%d\n", cap(kanal))  
 fmt.Printf("Gelen Veri Boyutu:%d\n", len(kanal))  
 fmt.Printf("Alinan Veri:%s\n", <-kanal)  
 fmt.Printf("Yeni Gelen Veri Boyutu:%d\n", len(kanal))  
  
}

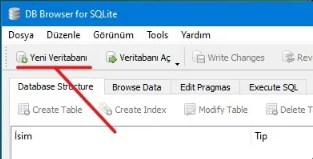


# VERİTABANI(DATABASE)

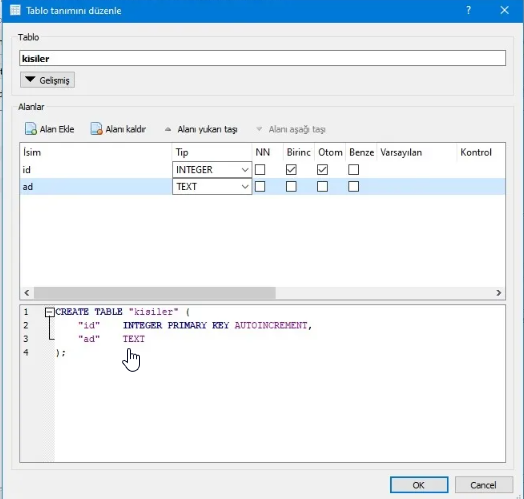
Veritabanı,herhangi bir konuda birbiriyle ilişkili olan verilerin düzenli bir biçimde depolamamızı ve kullanmamızı sağlayan depolama ortamıdır.

*VERİ TABANI YÖNETİM SİSTEMLERİ:*Yeni bir veritabanı oluşturmak,oluşturulan veritabanlarını düzenlemek,yönetmek,geliştirmek,belirli amaçlar için kullanmak ve bu veritabanlarının bakımlarını yapmak için kullanılan yazılıma veritabanı yönetim sistemi denir.

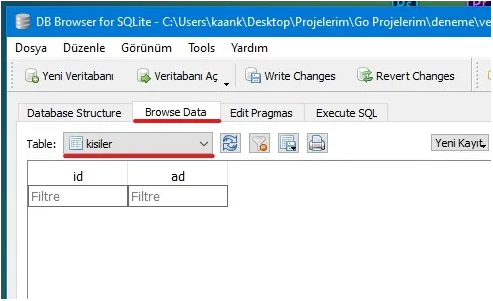
MySQL,PostreSQL,Microsoft SQL Server,SQLite,Oracle Database



Veritabanının kayıt yerini, programımızın kodlarının bulunacağı **main.go** dosyası ile aynı yeri seçelim ve ismini **veritabanı.db** olarak kaydedelim. İstediğiniz ismi de verebilirsiniz.



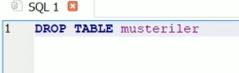
Tablomuzun ismini **kisiler** olarak ayarlayalım. **Alan Ekle**’ye tıklayarak yukarıdaki gibi **id** ve **ad** isminde alanlar oluşturalım. **id** alanının tipini **INTEGER** yaparak, sayısal verileri saklayabilmesini sağlıyoruz. **Birincil Anahtar** ve **Otomatik Arttırma** bölümlerini seçiyoruz. Otomatik Arttırma özelliği sayesinde tabloya veri eklendiğinde **id** içindeki değer her eklemede artacaktır. Bu da her satır için ayırıcı bir özellik olacaktır. **ad** alanının tipini **TEXT** yapıyoruz. **OK** butonuna basarak tabloyu oluşturuyoruz.

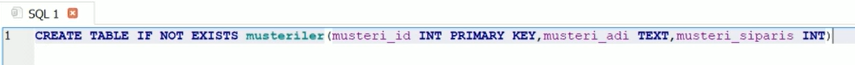
Tablomuz içindeki kayıtları görmek için **Browse Data** sekmesine tıklayalım. **Table** kısmının yanında tablo oluştururken yazdığımız **kisiler** seçeneğini seçelim. Şuanlık tablomuz boş. Çünkü içine bir kayıtta bulunmadık **DB Browser** programına bize yardımcı olduğu için teşekkür ederek artık Golang e geçebilmekteyiz.Veya SQL den yazabiliriz.

### database SQL TABLO OLUŞTURMA, VERİ EKLEME

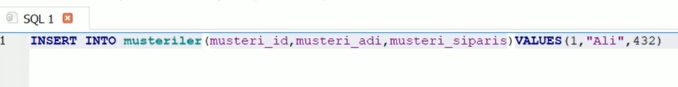
### 

Kod ile yeni bir tablo oluşturabiliriz “CREATE TABLE” Verisi ile.

 “DROP TABLE” ile tablo silme işlemi yapılabilir.

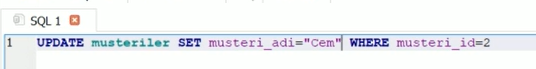


“İF NOT EXISTS” Aynı isimden tablo oluşturmayı engeller.

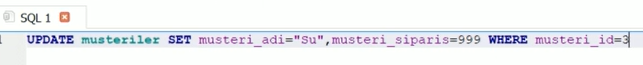


“İNSERT İNTO” =İçine veri koymamıza yarar. “VALUES” değerleri atamamıza yarar.

Çalıştırdıktan sonra buraya tek tek yeni veri girip çalıştır yaptığımızda liste uzamaya devam edecektir.



🡪Tablodaki verilerin ismini veya diğer verilen bilgileri değiştirmek için bu komutları eklememiz gerekmektedir.



🡪Örnekte olduğu gibi bu komutlar ile isim,sipariş no sunu değiştirebiliyoruz.

 Bir veriyi silmek için bu komutu kullanabiliriz “DELETE”

### SQL SORGULARI VERİ ÇEKME



\*🡪Tüm tablodaki verilerin gösterilmesi / Buraya filmad,bütçe yazsaydık sadece o veriler çıkıcaktı.

Yapimyili>1992 den büyük olanları sadece göstericektir



“LİKE” 🡪 “Buraya aramak istediğimiz verinin baş harfini bile yazsak onları gösterecektir

Örnek 🡨



“ORDER BY” (Yapımyili) yazan tablonun altı .Örneğin yıllara göre sıralanacaktır.

### VERİTABANI BAĞLANTISI VE TABLO OLUŞTURMA

package main  
  
import (  
 "database/sql"  
 \_ "github.com/mattn/go-sqlite3" //init  
 "log"  
)  
  
func checkErr(err error) { //OVERLOADING  
 if err != nil {  
 log.Fatal(err)  
 }  
}  
func main() {  
 //\_, err := os.Create("data1.db") KULLANMANIZA GEREK YOK  
 //checkErr(err)  
 db, err := sql.Open("sqlite3", "filmler.db")🡪Buraya farklı bir isimde dosya   
 defer db.Close() açtığımızda yeni .db oluşturacaktır.  
 checkErr(err)  
 \_, err = db.Exec("CREATE TABLE notlar(id INT PRIMARY KEY,notdurumu TEXT,isim TEXT)") 🡪Buradaki veriler SQL a yazığımız kodlar ile aynıdır.  
 checkErr(err)  
}

### dbsql VERİ çekme

package main  
  
import (  
 "database/sql"  
 "fmt"  
 \_ "github.com/mattn/go-sqlite3"  
 "log"  
)  
  
func checkErr6(err error) {  
 if err != nil {  
 log.Fatal(err)  
 }  
}  
  
func main() {  
  
 db, err := sql.Open("sqlite3", "filmler.db")  
 defer db.Close()  
 checkErr6(err)  
  
 satirlar, err := db.Query("SELECT \* FROM filmler")  
 checkErr6(err)  
  
 var id int  
 var filmAd string  
 var filmKategori string  
 var yonetmenAd string  
 var yapimyili int  
 var butce int  
  
 for satirlar.Next() {  
 err = satirlar.Scan(&id, &filmAd, &filmKategori, &yonetmenAd, &yapimyili, &butce)  
 checkErr6(err)  
  
 //fmt.Println(id)  
 //fmt.Println(filmAd)  
 //fmt.Println(butce)  
  
 fmt.Println("ID:", id, "Film Adı:", filmAd, "Film Kategori:", filmKategori, "Yönetmen:", yonetmenAd, "Yapım Yılı:", yapimyili, "Bütçe", butce/2)  
  
 }



### dbsql veri güncelleme

package main  
  
import (  
 "database/sql"  
 "fmt"  
 \_ "github.com/mattn/go-sqlite3"  
 "log"  
)  
func checkErr4(err error) {  
 if err != nil {  
 log.Fatal(err)  
 }  
}  
func main() {  
 db, err := sql.Open("sqlite3", "filmler.db")  
 defer db.Close()  
 checkErr4(err)  
 //GUNCELEME  
 guncelle, err := db.Prepare("UPDATE oyuncular SET yas=? WHERE ad=?") //stmt  
 checkErr4(err)  
 res, err := guncelle.Exec(99, "Orhan")  
  
 affect, err := res.RowsAffected()  
 //affect, err := res.LastInsertId()  
 checkErr4(err)  
 fmt.Println("İşlem Yapılan Satır Sayısı", affect)  
}

### dbsql array ile veri ekleme

package main  
  
import (  
 "database/sql"  
 \_ "github.com/mattn/go-sqlite3"  
 "log"  
)  
func checkErr3(err error) {  
 if err != nil {  
 log.Fatal(err)  
 }  
}  
func main() {  
  
 db, err := sql.Open("sqlite3", "filmler.db")  
 defer db.Close()  
 checkErr3(err)  
  
 oyuncuAd := []string{"Şener Şen", "Kemal Sunal"}  
 oyuncuId := []int{4, 5}  
 oyuncuYas := []int{74, 68}  
 for i := 0; i < len(oyuncuId); i++ {  
 veriekle, err := db.Prepare("INSERT INTO oyuncular(id,ad,yas)VALUES(?,?,?)")  
 checkErr3(err)  
 veriekle.Exec(oyuncuId[i], oyuncuAd[i], oyuncuYas[i])  
 }  
}

### dbsql VERİ SİLME

package main  
import (  
 "database/sql"  
 "fmt"  
 \_ "github.com/mattn/go-sqlite3"  
 "log"  
)  
func checkErr5(err error) {  
 if err != nil {  
 log.Fatal(err)  
 }  
}  
func main() {  
 db, err := sql.Open("sqlite3", "filmler.db")  
 defer db.Close()  
 checkErr5(err)  
 silme, err := db.Prepare("DELETE FROM oyuncular WHERE id=?")  
 checkErr5(err)  
 res, err := silme.Exec(1)  
 checkErr5(err)  
 affect, err := res.RowsAffected()  
 checkErr5(err)  
 fmt.Println("Silinen Satır Sayısı:", affect)  
}

### dbsql transactions

package main  
import (  
 "database/sql"  
 "fmt"  
 \_ "github.com/mattn/go-sqlite3"  
 "log"  
)  
func main() {  
 db, err := sql.Open("sqlite3", "filmler.db")  
 defer db.Close()  
 if err != nil {  
 log.Fatal(err)  
 }  
 guncelle, err := db.Prepare("UPDATE filmler SET butce=?")  
 if err != nil {  
 log.Fatal(err)  
 }  
 tx, BGerr := db.Begin()  
 if BGerr != nil {  
 log.Fatal(BGerr)  
 }  
 \_, txerr := tx.Stmt(guncelle).Exec(500)  
 if txerr != nil {  
 fmt.Println("Geri alınıyor...ROLLBACK")  
 tx.Rollback()  
 } else {  
 fmt.Println("Commit Edildi")  
 tx.Commit()  
 }  
}

### veri ekleme

package main  
import (  
 "database/sql"  
 \_ "github.com/mattn/go-sqlite3"  
 "log"  
)  
func checkErr2(err error) {  
 if err != nil {  
 log.Fatal(err)  
 }  
}  
func main() {  
 db, err := sql.Open("sqlite3", "filmler.db")  
 defer db.Close()  
 checkErr2(err)  
 //db.Exec("DROP TABLE notlar")  
 db.Exec("CREATE TABLE IF NOT EXISTS oyuncular(id INT PRIMARY KEY,ad TEXT,yas INT )")  
 oyuncuAd := "Brad Pitt"  
 oyuncuYas := 60  
 oyuncuId := 2  
  
 veriekle, err := db.Prepare("INSERT INTO oyuncular(id,ad,yas) VALUES(?,?,?)")  
 checkErr2(err)  
 veriekle.Exec(oyuncuId, oyuncuAd, oyuncuYas)  
}

# Xml Clipart Transparent PNG Hd, Xml Vector Icon, Xml Icons, Document Icon, File Icon PNG Image For Free DownloadXML,JSON

### XML NEDİR? (EXTENSİBLE MARKUP LANGUAGE)

XML dili,internet tabanlı veri alışverişini gerçekleştirmeyi sağlayan platformlar arasındaki iletişimi standardize eden bir işaretleme dilidir.

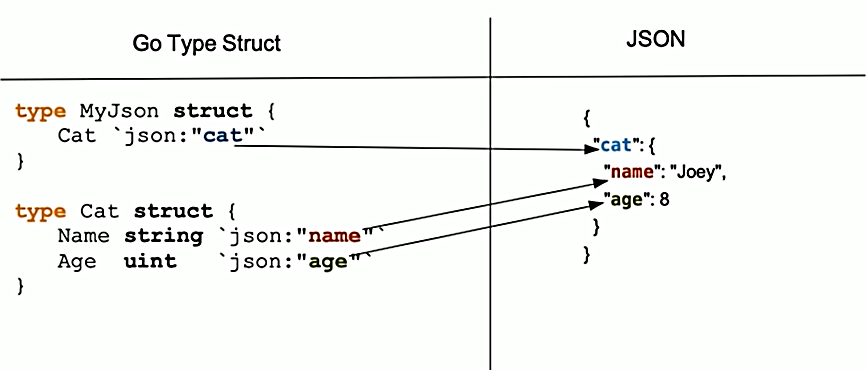
NEDEN KULLANIYORUZ?

* Farklı Platformlar arasındaki veri alış-verişi için
* Veriyi kategorize etmek için ve saklamak için
* Güvenlik için  
  

### Json file - Free interface iconsJSON NEDİR? (JAVASCRİPT OBJECT NOTATİON)

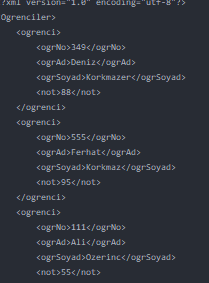
Basit veri yapılarını temsil etmek için tasarlanmış bağımsız bir veri değişim formatıdır.Esas olarak iki sistem arasındaki veri alışverişi için kullanılır.

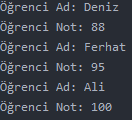




### XML DOSYASI OLUŞTURMA

<?xml version="1.0" encoding="utf-8"?> 🡪 BU VERİ XML DOSYASININ BAŞINDA OLMASI GEREKMEKTEDİR  
<Ogrenciler>  
 <ogrenci>  
 <ogrNo>349</ogrNo>  
 <ogrAd>Deniz</ogrAd>  
 <ogrSoyad>Korkmazer</ogrSoyad>  
 <not>88</not>  
 </ogrenci>  
 <ogrenci>  
 <ogrNo>555</ogrNo>  
 <ogrAd>Ferhat</ogrAd>  
 <ogrSoyad>Korkmaz</ogrSoyad>  
 <not>95</not>  
 </ogrenci>  
 <ogrenci>  
 <ogrNo>111</ogrNo>  
 <ogrAd>Ali</ogrAd>  
 <ogrSoyad>Ozerinc</ogrSoyad>  
 <not>100</not>  
 </ogrenci>  
</Ogrenciler>

package main  
  
import (  
 "encoding/xml"  
 "fmt"  
 "io/ioutil"  
 "log"  
 "os"  
)  
  
type Students struct {  
 XMLNAME xml.Name `xml:"Ogrenciler"`  
 OgrenciListesi []Ogr `xml:"ogrenci"`  
}  
type Ogr struct {  
 //ALT 96 YADA CTRL VE ALT , TUŞUNA BASILCAK  
 XMLNAME xml.Name `xml:"ogrenci"`  
 OgrenciNo int `xml:"ogrNo"`  
 OgrenciAd string `xml:"ogrAd"`  
 OgrenciSoyad string `xml:"ogrSoyad"`  
 OgrenciNot int `xml:"not"`  
}  
  
func main() {  
  
 dosya, err := os.Open("ogrenciler.xml")  
 if err != nil {  
 log.Fatal(err)  
 }  
 defer dosya.Close()  
  
 arr, err := ioutil.ReadAll(dosya)  
 if err != nil {  
 log.Fatal(err)  
 }  
  
 //var ogrenci Ogr  
 //xml.Unmarshal(arr, ogrenci)  
 //fmt.Println(string(arr))  
  
 Ogrenci := &Students{}  
 xml.Unmarshal(arr, Ogrenci)  
  
 for \_, degerler := range Ogrenci.OgrenciListesi {  
 fmt.Println("Öğrenci Ad:", degerler.OgrenciAd)  
 fmt.Println("Öğrenci Not:", degerler.OgrenciNot)  
 }  
  
}



### GO PROGRAMLAMA İLE XML DOSYASI OLUŞTURMA

package main  
  
import (  
 "encoding/xml"  
 "io"  
 "log"  
 "os"  
)  
  
type Sirket struct {  
 XMLName xml.Name `xml:"sirket"`  
 Personeller []Personel `xml:"personel"`  
}  
type Personel struct {  
 XMLName xml.Name `xml:"personel"`  
 TCNO int `xml:"tcno"` 🡪 T.C KİMLİK  
 PerAd string `xml:"personelAd"`  
 PerSoyad string `xml:"personelSoyad"`  
 PerMevki string `xml:"mevki"`  
}  
func main() {  
  
 dosya, err := os.Create("personel.xml") 🡪 (?).xml istediğimiz isimde burdan  
 if err != nil { Kayıt edebiliriz  
 log.Fatal(err)  
 }  
 str := "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n"  
 dosya.WriteString(str)  
  
 s := &Sirket{}  
  
 s.Personeller = append(s.Personeller, Personel{TCNO: 23296, PerAd: "Deniz", PerSoyad: "Korkmazer", PerMevki: "Yazılımcı"})  
 s.Personeller = append(s.Personeller, Personel{TCNO: 38954, PerAd: "Ali", PerSoyad: "Özerinç", PerMevki: "Yönetici"})  
  
 xmlWriter := io.Writer(dosya)  
  
 enc := xml.NewEncoder(xmlWriter)  
 enc.Indent(" ", " ") //Girinti oluşturma  
 enc.Encode(s)  
  
 //XML TO JSON  
  
 //jsonVeri, err := json.Marshal(s)  
 //if err != nil {  
 // log.Fatal(err)  
 //}  
 //  
 //jsonDosya, err := os.Create("yeni.json")  
 //if err != nil {  
 // log.Fatal(err)  
 //}  
 //  
 //defer jsonDosya.Close()  
 //  
 //jsonDosya.Write(jsonVeri)  
}

Oluşturduğunuz xml dosyası böyle görünmektedir

<?xml version="1.0" encoding="utf-8"?>  
 <sirket>  
 <personel>  
 <tcno>23296</tcno>  
 <personelAd>Deniz</personelAd>  
 <personelSoyad>Korkmazer</personelSoyad>  
 <mevki>Yazılımcı</mevki>  
 </personel>  
 <personel>  
 <tcno>38954</tcno>  
 <personelAd>Ali</personelAd>  
 <personelSoyad>Özerinç</personelSoyad>  
 <mevki>Yönetici</mevki>  
 </personel>  
 </sirket>

### json dosyası oluşturma

{ 🡪Örnek1:  
 "marka": "Renault",  
 "fiyat": 750000,  
 "model": "Megan Sedan",  
 "paket": ["FULL","COMFORT","İCON"],  
 "özellik": {  
 "klima": true,  
 "jant": 18  
 },  
 "bayisatis": true  
}

{ 🡪Örnek2:  
 "universite": "Harvard Üniversitesi",  
 "bolum": "Software Engineer",  
 "ogrenciler": [  
 {  
 "ogrNo": 349,  
 "ogrAd": "Deniz",  
 "ogrSoyad": "Korkmazer",  
 "hobiler": "DJ"  
 },  
 {  
 "ogrNo": 552,  
 "ogrAd": "Ferhat",  
 "ogrSoyad": "Korkmaz",  
 "arkadası": "Sercan"  
 }  
 ],  
 "dersler": [  
 {  
 "dersNo": 101,  
 "dersAdı": "Programlama"  
 },  
 {  
 "dersNo": 404,  
 "dersAdı": "Veri Yapıları"  
 }  
 ]  
}

### JSON İŞLEMLERİ

package main  
  
import (  
 "encoding/json"  
 "fmt"  
 "io/ioutil"  
 "log"  
 "os"  
)  
  
type Students struct {  
 Ogrenciler []Ogrenci `json:"ogrenciler"`  
}  
  
type Ogrenci struct {  
 OgrenciNo int `json:"ogrNo"`  
 OgrenciAd string `json:"ogrAd"`  
 OgrenciSoyad string `json:"ogrSoyad"`  
 OgrenciNot int `json:"ogrNot"`  
}  
  
func main() {  
  
 dosya, err := os.Open("ogrenciler.json")  
 if err != nil {  
 log.Fatal(err)  
 }  
 defer dosya.Close()  
  
 arr, err := ioutil.ReadAll(dosya)  
 if err != nil {  
 log.Fatal(err)  
 }  
  
 var Ogr Students  
  
 json.Unmarshal(arr, &Ogr)  
  
 //fmt.Println(string(arr))  
  
 for i := 0; i < len(Ogr.Ogrenciler); i++ {  
 fmt.Println("Öğrenci No:", Ogr.Ogrenciler[i].OgrenciNo)  
 fmt.Println("Öğrenci Ad:", Ogr.Ogrenciler[i].OgrenciAd)  
 fmt.Println("Öğrenci Not:", Ogr.Ogrenciler[i].OgrenciNot)  
 }  
  
}



{  
 "ogrenciler": [  
 {  
 "ogrNo": 349,  
 "ogrAd": "Deniz",  
 "ogrSoyad": "Korkmazer",  
 "ogrNot": 46  
  
 },  
 {  
 "ogrNo": 552,  
 "ogrAd": "Ferhat",  
 "ogrSoyad": "Korkmaz",  
 "ogrNot": 96  
 },  
 {  
 "ogrNo": 321,  
 "ogrAd": "Sercan",  
 "ogrSoyad": "Yalirgav",  
 "ogrNot": 75  
 },  
 {  
 "ogrNo": 911,  
 "ogrAd": "Recep",  
 "ogrSoyad": "İvedik",  
 "ogrNot": 88  
 }  
 ]  
  
  
  
}

### GO PROGRAMLAMA İLE JSON DOSYASI OLUŞTURMA

{"CListe":[{"ID":543,"Isim":"VOC","KurulusTarihi":1602,"DevamEdiyorMu":false},{"ID":16,"Isim":"TOLON","KurulusTarihi":1950,"DevamEdiyorMu":false}]}

package main  
  
import (  
 "encoding/json"  
 "fmt"  
 "io"  
 "log"  
 "os"  
)  
type Companies struct {  
 CListe []Sirket  
}  
  
type Sirket struct {  
 ID int  
 Isim string  
 KurulusTarihi int  
 DevamEdiyorMu bool  
}  
func main() { 🡪func main’den sonrası bir sonraki sayafadadır.  
 Sirket1 := Sirket{}  
 Sirket1.ID = 1  
 Sirket1.Isim = "ERMAT"  
 Sirket1.KurulusTarihi = 1989  
 Sirket1.DevamEdiyorMu = true  
 sirket1Byte, err := json.Marshal(Sirket1)  
 if err != nil {  
 log.Fatal(err)  
 }  
 fmt.Println(string(sirket1Byte))  
 sirket2 := Sirket{55, "Mais", 1969, true}  
 sirket2Byte, \_ := json.Marshal(sirket2)  
 fmt.Println(string(sirket2Byte))  
 dosya, err := os.Create("sirketlerim.json")  
 if err != nil {  
 log.Fatal(err)  
 }  
 c1 := Companies{}  
 c1.CListe = append(c1.CListe, Sirket{ID: 543, Isim: "VOC", KurulusTarihi: 1602, DevamEdiyorMu: false})  
 c1.CListe = append(c1.CListe, Sirket{ID: 16, Isim: "TOLON", KurulusTarihi: 1950, DevamEdiyorMu: false})  
 jsonWriter := io.Writer(dosya)  
 enc := json.NewEncoder(jsonWriter)  
 json.MarshalIndent(c1, " ", "\t")  
 enc.Encode(c1)  
}

### basit mail gönderme işlemi

package main  
//SMTP -- Simple Mail Transfer Protocol  
import (  
 "fmt"  
 "log"  
 "net/smtp"  
)  
//SMTP mail göndermek  
//POP IMAP mail almak  
func main() {  
 // Gönderici Bilgileri  
 gonderen := "ermat.denizkorkmazer@gmail.com"  
 sifre := "deneme123"  
 // Alıcı Bilgiler  
 alici := []string{  
 "ermat@renault.com",  
 }  
 // STMP Ayarları  
 smtpHost := "smtp.gmail.com"  
 smtpPort := "587"   
 // Mesaj. 🡪Başlık ekleme baslik:=”BAŞLIK”   
 baslik := "Subject:Bu bir başlıktır\n" / içerik:=”Bu bir mail içeriğidir”  
 icerik := "Bu mail icerigidir"  
 mesaj := []byte(baslik + icerik)  
 //Authentication İşlemleri  
 auth := smtp.PlainAuth("", gonderen, sifre, smtpHost)  
 // Email Gönderme  
 err := smtp.SendMail(smtpHost+":"+smtpPort, auth, gonderen, alici, mesaj)  
 if err != nil {  
 log.Fatal(err)  
 }  
 fmt.Println("E-Posta başarıyla gönderildi.")  
}

### HTML YAPILI MAİL GÖNDERME

package main  
  
import (  
 "fmt"  
 "log"  
 "net/smtp"  
)  
func main() {  
 gonderen := "ermat.denizkorkmazer@gmail.com"  
 sifre := "ermat123"  
  
 alici := []string{  
 "ermat@renault.com",  
 }  
 smtpHost := "smtp.gmail.com"  
 smtpPort := "587"  
 //MIME -- Multipurpose Internet Mail Extensions  
 baslik := "Subject:Merhaba Ermat\n"  
 mime := "MIME-version:1.0\nContent-Type:text/html; charset=\"UTF-8\";\n\n"  
 icerik := "<html><body><h1 style=\"color:blue;\">Merhaba Bu bir HTML göstergesidir.</h1></body></html>"  
 mesaj := []byte(baslik + mime + icerik)  
  
 auth := smtp.PlainAuth("", gonderen, sifre, smtpHost)  
  
 err := smtp.SendMail(smtpHost+":"+smtpPort, auth, gonderen, alici, mesaj)  
 if err != nil {  
 log.Fatal(err)  
 }  
 fmt.Println("Mail gönderildi")  
}

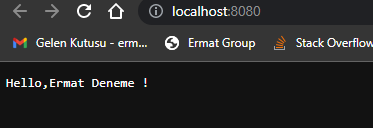
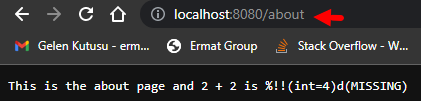
### GO MAİL PACKAGE (gopkg.in/gomail.v2)

package main  
  
import (  
 "gopkg.in/gomail.v2"  
 "log"  
)  
func main() {  
 mesaj := gomail.NewMessage()  
  
 mesaj.SetHeader("From", "ermat.denizkorkmazer@gmail.com")  
 mesaj.SetHeader("To", "renault")  
 mesaj.SetHeader("Subject", "Önemli!!!")  
  
 mesaj.Attach("udemy.jpg")  
  
 mesaj.SetBody("text/html", "<p>Bu bir paragraf</p><h2 style=\"text-align:center\">Bunu Gördün mü?</h2>")  
 dia := gomail.NewDialer("smtp.gmail.com", 587, "ermat.denizkorkmazer@gmail.com", "ermat123")  
 err := dia.DialAndSend(mesaj)  
 if err != nil {  
 log.Fatal(err)  
 }  
}

# “MERHABA DÜNYA” İLE WEB UYGULAMASI YAPMAK

YABANCI KAYNAK:

package main  
  
import (  
 "fmt"  
 "net/http"  
)  
  
func main() {  
 http.HandleFunc("/", func(w http.ResponseWriter, r \*http.Request) {  
 n, err := fmt.Fprintf(w, "Hello,Ermat Deneme !")  
 if err != nil {  
 fmt.Println(err)  
 }  
 fmt.Println(fmt.Sprintf("Number of ytes written: %d", n))  
 })  
 \_ = http.ListenAndServe(":8080", nil) 🡪 http://localhost:8080/  
}

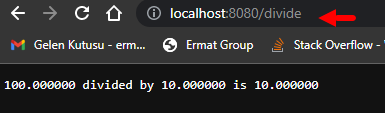
 

### WEB FUNCTİONS AND HANDLERS

package main  
  
import (  
 "fmt"  
 "net/http"  
)  
const portNumber = ":8080" 🡪Portal numarası  
// Home is the handler for the home page  
func Home(w http.ResponseWriter, r \*http.Request) {  
 fmt.Fprintf(w, "This is the home page") 🡪Ana sayfa   
}  
// About is the handler for the about page  
func About(w http.ResponseWriter, r \*http.Request) {  
 sum := addValues(2, 2)  
 \_, \_ = fmt.Fprintf(w, fmt.Sprintf("This is the about page and 2 + 2 is %d"), sum)  
}  
// addValues adds two ints x and y, and returns the sum  
func addValues(x, y int) int {  
 return x + y  
}  
// main is the main function  
func main() {  
 http.HandleFunc("/", Home)  
 http.HandleFunc("/about", About) 🡪Sayfa isimleri “/..”  
  
 fmt.Println(fmt.Sprintf("Staring application on port %s", portNumber))  
 \_ = http.ListenAndServe(portNumber, nil)  
}

### HATA KONTROLÜ

package main  
  
import (  
 "errors"  
 "fmt"  
 "net/http"  
)  
const portNumber = ":8080"  
  
// Home is the handler for the home page  
func Home(w http.ResponseWriter, r \*http.Request) {  
 fmt.Fprintf(w, "This is the home page")  
}  
// About is the handler for the about page  
func About(w http.ResponseWriter, r \*http.Request) {  
 sum := addValues(2, 2)  
 \_, \_ = fmt.Fprintf(w, fmt.Sprintf("This is the about page and 2 + 2 is %d", sum))  
}  
// Divide divides one value into another and returns message with result  
func Divide(w http.ResponseWriter, r \*http.Request) {  
 f, err := divideValues(100.0, 10.0)  
 if err != nil {  
 \_, \_ = fmt.Fprintf(w, "Cannot divide by 0")  
 return  
 }  
 \_, \_ = fmt.Fprintf(w, fmt.Sprintf("%f divided by %f is %f", 100.0, 10.0, f))  
}  
func divideValues(x, y float32) (float32, error) {  
 if y <= 0 {  
 err := errors.New("Cannot divide by zero")  
 return 0, err  
 }  
 result := x / y  
 return result, nil  
}  
// addValues adds two ints x and y, and returns the sum  
func addValues(x, y int) int {  
 return x + y  
}  
// main is the main function  
func main() {  
 http.HandleFunc("/", Home)  
 http.HandleFunc("/about", About)  
 http.HandleFunc("/divide", Divide)  
  
 fmt.Println(fmt.Sprintf("Staring application on port %s", portNumber))  
 \_ = http.ListenAndServe(portNumber, nil)  
}



### HTML ŞABLONLAR(TEMPLATES)

HTML Şablonlar, Golang üzerinde web sayfalarının dinamikliği için kullanılır. Yani şablonlar kullanarak web sayfalarımızın belirlediğimiz bölümlerini Go üzerinden değişikliğe uğratabiliriz.

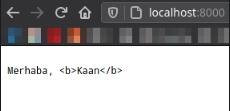
## ÖRNEK KULLANIM:

Merhaba, <b>{{ . }}</b>

Yukarıdaki örnekte {{ }} süslü parantezler içerisinde . (nokta) yazıyor. Bu da Go şablon işlenirken bu kısma Go tarafından vereceğimiz değerin geleceği anlamına geliyor.

Şimdi yukarıdaki örneğimizi sablon.html adı ile kaydedelim

Sayfamıza bakmak için <http://localhost:8000> adresine gittiğimizde, şöyle bir sonuç ile karşılaşacağız.



package main  
  
import (  
 "fmt"  
 "html/template"  
 "net/http"  
)  
// yakalayıcı fonksiyonumuz  
func anasayfa(w http.ResponseWriter, r \*http.Request) {  
 //isim değişkenimiz  
 isim := "Kaan"  
 //burada şablon oluşturuyoruz  
 şablon, \_ := template.ParseFiles("sablon.html")  
 //Burada da şablonu çalıştırmasını ve isim  
 //değişkenini kullanmasını istiyoruz.  
 şablon.Execute(w, isim)  
}  
func main() {  
 fmt.Println("Sunucu Başladı")  
 //ana dizini anasayfa fonksiyonu ile yakalayalım  
 http.HandleFunc("/", anasayfa)  
 //portu 8000 yapalım ve sunucuyu başlatalım  
 http.ListenAndServe(":8000", nil)  
}

### Golang Package Archive · GitHub

### html kodu gönderme

Bu örneğimizde sablon.html dosyamız aşağıdaki gibi olsun.

<!DOCTYPE html>  
{{ . }}

package main  
import (  
 "fmt"  
 "html/template"  
 "net/http"  
)  
func anasayfa(w http.ResponseWriter, r \*http.Request) {  
 //html kodumuz  
 htmlKodu := "<h1>Merhaba</h1>"  
  
 şablon, \_ := template.ParseFiles("sablon.html")  
  
 şablon.Execute(w, htmlKodu)  
}  
func main() {  
 fmt.Println("Sunucu Başladı")  
  
 http.HandleFunc("/", anasayfa)  
  
 http.ListenAndServe(":8000", nil)  
}

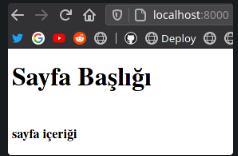
Çözüm olarak da Go tarafındaki değişkenimizin tipini template.HTML yapacağız. main.go dosyamızda htmlKodu değişkenimizin tipini değiştirelim.

var htmlKodu template.HTML = "<h1>Merhaba</h1>"

### ŞABLONA STRUCT GÖNDERME

package main  
  
import (  
 "fmt"  
 "html/template"  
 "net/http"  
)  
type bilgi struct {  
 Başlık string  
 İçerik template.HTML  
}  
func anasayfa(w http.ResponseWriter, r \*http.Request) {  
 sayfaBilgi := bilgi{  
 Başlık: "Sayfa Başlığı",  
 İçerik: "<b>sayfa içeriği</b>",  
 }  
 şablon, \_ := template.ParseFiles("sablon.html")  
 şablon.Execute(w, sayfaBilgi)  
}  
func main() {  
 fmt.Println("Sunucu Başladı")  
 http.HandleFunc("/", anasayfa)  
 http.ListenAndServe(":8000", nil)  
}

<!DOCTYPE html>  
<h1>{{ .Başlık }}</h1><br>  
{{.İçerik}}



### ŞABLON İÇİNDEN DEĞİŞKEN ATAMA

Şablon içerisinde oluşturduğumuz değişkenleri tıpkı PHP'deki gibi $ işareti ile kullanırız.

package main  
  
import (  
 "fmt"  
 "html/template"  
 "net/http"  
)  
  
func anasayfa(w http.ResponseWriter, r \*http.Request) {  
 şablon, \_ := template.ParseFiles("sablon.html")  
  
 //şablona değer göndermeyeceğimiz için burası nil olsun.  
 şablon.Execute(w, nil)  
}  
func main() {  
 fmt.Println("Sunucu Başladı")  
 http.HandleFunc("/", anasayfa)  
 http.ListenAndServe(":8000", nil)  
}

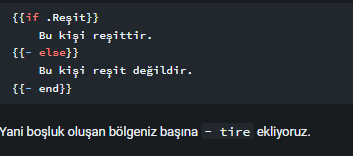
sablon.html dosyamız da aşağıdaki gibi olsun.

sablon.html

<!DOCTYPE html>  
  
{{ $isim := "kaan"}}  
{{ $isim }}

### ŞABLONDA İF-ELSE KULLANIMI

{{if .Reşit}}  
Bu kişi reşittir.  
{{else}}  
Bu kişi reşit değildir.  
{{end}}

Yukarıda dikkat etmemiz gereken şey if-else'in sonuna end eklememiz gerekiyor. Sadece if olsaydı bile end eklememiz gerekir.

### ŞABLONDA RANGE DÖNGÜSÜ KULLANIMI

Range döngüsü ile web sayfamızın içerisinde bir listenin sıralanmasını sağlayabiliriz.

<ul>  
{{range .Liste}}  
{{.}}  
{{end}}  
</ul>

### ŞABLON İÇERSİNDE YORUM SATIRI OLUŞTURMA

Biliyorsunuz ki, HTML kodu içerisindeki yorumlar sayfa kaynağını göstere tıklayınca gözüküyor. Eğer yorumların gözükmesini istemezsek Şablon yorumu olarak yazabiliriz.

{{/\* Yorumu buraya yazabilirsiniz \*/}}

# NET/http İLE WEB SERVER OLUŞTURMA

**localhost:5555** üzerinde çalışacak olan bir web sunucusu oluşturacağız.

package main  
import (  
 "fmt"  
 "net/http"  
)  
  
func handler(w http.ResponseWriter, r \*http.Request) {  
 fmt.Fprintf(w, "Merhaba %s", r.URL.Path[1:])  
}  
  
func main() {  
 http.HandleFunc("/", handler)  
 http.ListenAndServe(":5555", nil)  
  
 fmt.Println("Web Sunucu")  
}

**Peki bu olayın açıklaması nedir?** **main()** fonksiyonunun içerisinde 2 temel fonksiyon bulunuyor. **HandleFunc()** fonksiyonu belirlediğimiz adrese girildiğinde hangi fonksiyonun çalıştırılacağınız belirliyor. **ListenAndServe()** fonksiyonu ise sunucunun ayağa kalkmasını ve istediğimiz bir porttan ulaşılmasını sağlıyor. Eğer sunucuya dosya verme yoluyla işlem yapmasını istiyorsak aşağıdaki yönteme başvurmalıyız. **index.html** adında bir dosya oluşturuyoruz. İçine aşağıdakileri yazıyoruz ve kaydediyoruz.

<!DOCTYPE html>  
<html lang="tr">  
<head>  
<title>Sayfa Başlığı</title>  
</head>  
<body>  
Merhaba Dünya  
</body>  
</html>

Şimde de sunucu işlemlerini gerçekleştireceğimiz **main.go** dosyamızı oluşturalım.

package main  
import (  
 "fmt"  
 "io/ioutil"  
 "net/http"  
)  
func loadFile(fileName string) (string, error) {  
 bytes, err := ioutil.ReadFile(fileName)  
 if err != nil {  
 return "", err  
 }  
 return string(bytes), nil  
}  
func handler(w http.ResponseWriter, r \*http.Request) {  
 var body, \_ = loadFile("index.html")  
 fmt.Fprintf(w, body)  
}  
func main() {  
 http.HandleFunc("/", handler)  
 http.ListenAndServe(":5555", nil)  
}

Tarayıcıdan **localhost:5555** adresine girdiğimiz zaman oluşturmuş olduğumuz **index.html** dosyasının görüntülendiğini göreceksiniz. Açıklayacak olursak eğer; **loadFile()** fonksiyonumuz **index.html** programa aktarıldığında **byte** türünde olduğu için onu okuyabileceğimiz **string** türüne çevirdi. Bu özellik programımıza **“io/ioutil”** paketi sayesinde eklendi.

# http İSTEKLERİ(REQUESTS)

### GET İSTEĞİ

package main  
import (  
 "fmt"  
 "io"  
 "net/http"  
)  
func main() {  
 //Google anasayfasına GET isteğinde bulunalım  
 //ve dönen cevabı cevap değişkenine atayalım  
 cevap, hata := http.Get("https://www.google.com.tr")  
 if hata != nil {  
 //Ya sev ya terket dedik ve paniği bastık.  
 panic(hata)  
 }  
 //Body'i kapatmayı unutmayalım  
 defer cevap.Body.Close()  
 //Bu esnada Body'den cevabı çekip sayfa değişkenine alıyoruz.  
 //Çünkü bunu yapmadan okunabilir bir sonuç alamayız.  
 sayfa, hata := io.ReadAll(cevap.Body)  
 if hata != nil {  
 panic(hata)  
 }  
 //sayfa değişkeni byte dizisi tipinde olduğu için  
 //string tipine cevirip okuyalım  
 fmt.Println(string(sayfa))  
 //Çıktımız Google TR anasayfasının kaynak kodları olacaktır.  
}

### SORGU PARAMETRESİ EKLEME

Örneğin, yaptığımız istekte search?q=golang&hl=de gibi bir sorgu kısmı olmasını istiyoruz. URL'yi girdiğimiz kısmı bunu el ile ekleyebiliriz tabiki. Ama onunda kolay bir yöntemi var. Bunun için net/url paketini içe aktaralım.

func main() {  
params := url.Values{  
"q": {"golang"}, //aranacak metin  
"hl": {"tr"}, //örnek olarak Türkçe sonuç vermesi için  
}  
fmt.Println("https://www.google.com.tr/search?" + params.Encode())  
}

Çıktımız: https://www.google.com.tr/search?q=golang&hl=tr olacaktır.

package main  
import (  
 "fmt"  
 "io"  
 "net/http"  
 "net/url"  
)  
  
func main() {  
 params := url.Values{  
 "q": {"golang"},  
 "hl": {"tr"},  
 }  
  
 //URL'imizi aşağıdaki gibi yazalım.  
 cevap, hata := http.Get("https://www.google.com.tr/search?" + params.Encode())  
 if hata != nil {  
 panic(hata)  
 }  
 defer cevap.Body.Close()  
  
 sayfa, hata := io.ReadAll(cevap.Body)  
  
 if hata != nil {  
 panic(hata)  
 }  
  
 fmt.Println(string(sayfa))  
  
}

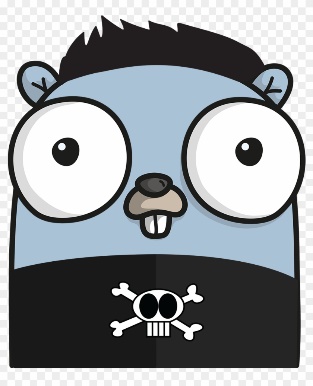


### POST İSTEĞİ

Post isteğini yaparken örnek bir json verisi göndermeyi görelim.

package main  
  
import (  
 "bytes"  
 "encoding/json"  
 "fmt"  
 "io"  
 "net/http"  
)  
  
func main() {  
 address := "https://orneksite.com/kisiler"  
 hamVeri := map[string]string{  
 "isim": "Deniz",  
 "soyisim": "Korkmazer",  
 }  
 jsonVeri, hata := json.Marshal(hamVeri)  
 //jsonVeri string tipine çevirildiğinde:  
 //{"isim":"Kaan","soyisim":"Kuşcu"}  
  
 if hata != nil {  
 panic(hata)  
 }  
  
 //Aşağıdaki gönderdiğimiz verinin json tipinde olduğunu bildiriyoruz.  
 sorgu, hata := http.Post(address, "application/json", bytes.NewBuffer(jsonVeri))  
 if hata != nil {  
 panic(hata)  
 }  
  
 //sonucu sonuç değişkenine atayalım  
 sonuç, hata := io.ReadAll(sorgu.Body)  
 if hata != nil {  
 panic(hata)  
 }  
  
 //Son olarak ekrana bastıralım.  
 fmt.Println(string(sonuç))  
}

Yukarıdaki örnekte json verimizi map ile oluşturarak post isteğinde kullandık. Map ile oluşturduğumuz json verisi esnek olmadığı için, yani yukarıdaki örnekte sadece string tipinde veri oluşturabildiğimiz için, bir de struct ile oluşturmayı görelim. Anonim bir struct oluşturup işimizi görebiliriz.



package main  
  
import (  
 "bytes"  
 "encoding/json"  
 "fmt"  
 "io"  
 "net/http"  
)  
  
func main() {  
 address := "https://orneksite.com/kisiler"  
  
 //struct içindeki değişken isimlerini büyük harfle  
 //yazmazsak json'a çevirirken hata alırız.  
 structVeri := struct {  
 İsim string `json:"isim"` //json'da isim olacak  
 Soyisim string `json:"soyisim"` //json'da soyisim olacak  
 Yaş int `json:"yas"` //json'da yas olacak  
 }{"Deniz", "Korkmazer", 23}  
  
 //struct'ımızı json'a çevirelim.  
 jsonVeri, hata := json.Marshal(structVeri)  
 //jsonVeri string tipine çevirildiğinde:  
 //{"isim":"Deniz","soyisim":"Korkmazer","yas":23}  
  
 if hata != nil {  
 panic(hata)  
 }  
  
 //Aşağıdaki gönderdiğimiz verinin json tipinde olduğunu bildiriyoruz.  
 sorgu, hata := http.Post(address, "application/json", bytes.NewBuffer(jsonVeri))  
 if hata != nil {  
 panic(hata)  
 }  
  
 //sonucu sonuç değişkenine atayalım  
 sonuç, hata := io.ReadAll(sorgu.Body)  
 if hata != nil {  
 panic(hata)  
 }  
  
 //Son olarak ekrana bastıralım.  
 fmt.Println(string(sonuç))  
}

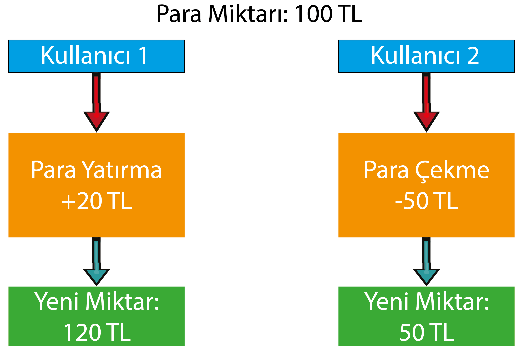


# WaitGroup ile Asenkron İşlemleri Beklemek

Goroutine’leri Asenkron programlama yaparken kullanırız. Böylece aynı anda birden fazla işlem gerçekleştirebiliriz. Peki programımızın belirttiğimiz asenkron işlemleri bekleme gibi bir ihtiyacı olsaydı, ne yapmamız gerekirdi? Bu durumlarda WaitGroup'lardan faydalanabiliriz. Örneğin projemizde 3 adet asenkron işlem bulunuyorsa, WaitGroup'a 3 değerini ekleriz. Her asenkron işlem tamamlandığında WaitGroup -1 azalır ve sıfıra geldiğinde WaitGroup tamamlanmış olur. WaitGroup'u kullanmak için ise "sync" paketini projemize dahil ediyoruz. Kodlar üzerinde açıklamasını görelim.

package main  
  
import (  
 "fmt"  
 "sync" //WaitGroup'u kullammak için  
 "time" //bekleme işlemleri için  
)  
/\*  
\* waitgroup nesnesini işaretçi olarak parametre veriyoruz.  
\* işaretçi olarak vermemizin sebebi, programın bekleme işlemi için  
\* asıl waitgroup nesnesini kontrol etmesidir.  
func fonksiyon1(wg \*sync.WaitGroup) {  
 //fonksiyonun 2 sn beklemesini istiyoruz.  
 time.Sleep(2 \* time.Second)  
 fmt.Println("Fonk1 tamamlandı")  
 //wg.Done() fonksiyonu ile waitgroup nesnesini -1 azalttık.  
 wg.Done()  
}  
//bu fonksiyonumuza da wg nesnesini işaretçi ile parametre olarak verdik.  
func fonksiyon2(wg \*sync.WaitGroup) {  
 //fonksiyonu 3 sn uyuttuk.  
 time.Sleep(3 \* time.Second)  
 fmt.Println("Fonk2 tamamlandı")  
 //-1 daha eksilttik.  
 wg.Done()  
}  
func main() {  
 \* Öncelikle waitgroup'u kullanabilmek için bir waitgroup  
 \* nesnesi oluşturuyoruz.  
 var wg sync.WaitGroup  
 /\*  
 \* waitgroup'a 2 ekliyoruz. Yani 2 tane işlemden yanıt gelmesini  
 \* beklemesini istiyoruz. Aslında burada beklemeyecek. Sadece  
 \* işlem sayısını belirttik.  
 wg.Add(2)  
 \* fonksiyon1 ve fonksiyon2'ye oluşturduğumuz wg örneğinin  
 \* bellekteki adresinin veriyoruz.  
 go fonksiyon1(&wg)  
 go fonksiyon2(&wg)  
 fmt.Println("Merhaba Dünya!")  
 \* Burada wg.Wait() fonksiyonu ile asenkron işlemleri beklemesini  
 \* sağlıyoruz. yani waitgroup'un 0'a düşmesini bekliyoruz.  
 \* Eğer waitgroup olmadan yapsaydık. asenkron fonksiyonlarımızın tamamlanmasını  
 \* beklemeden program kendini sonlandırırdı.  
 \*/  
 wg.Wait()  
 //waitgroup tamamlandığında ekrana yazı bastıralım.  
 fmt.Println("WaitGroup'lar tamamlandı.")  
}

# Mutex ile Asenkron İşlem Sırası



package main  
import (  
 "fmt"  
 "sync" // mutex'i kullanmak için  
)  
//global olarak mutex nesnesi oluşturalım.  
var mt sync.Mutex  
func paraÇek(bakiye \*float64, çekilecekMiktar float64, wg \*sync.WaitGroup) {  
 \* mt isimli mutex'i bu işlem yapılırken kilitliyoruz.  
 \* bu sayede mt mutex'ini başka işlemler kullanamıyor.  
 mt.Lock()  
 bu kısımda asenkron olmasını istemediğimiz işlemi yapalım.  
 \*bakiye -= 15  
 fmt.Printf("Yeni Bakiye: %.2f\n", \*bakiye)  
 \* diğer işlemlerinde kullanabilmesi için mutex'i tekrardan açalım.  
 \* mt mutex açılınca diğer asenkron işlemdeki mt mutex'i çalışmaya başlar.  
 mt.Unlock()  
 fmt.Println("Çekme işlemi tamamlandı.")  
 \* waitgroup ile işlemin tamamlandığını belirttik.  
 \* böylece wg havuzu 2'den 1'e düştü  
 wg.Done()  
}  
//bu fonksiyonda yukarıdaki ile aynı mantıkta  
func paraYatır(bakiye \*float64, yatırılacakMiktar float64, wg \*sync.WaitGroup) {  
 mt.Lock()  
 \*bakiye += 65  
 fmt.Printf("Yeni Bakiye: %.2f\n", \*bakiye)  
 mt.Unlock()  
 fmt.Println("Yatırma işlemi tamamlandı.")  
 wg.Done()  
}  
func main() {  
 \* asenkron işlemlerimizin, ana iş parçacığında tamamlanmasını  
 \* beklemek için waitgroup nesnesi oluşturalım  
 var wg sync.WaitGroup  
 //2 fonksiyonu da bekleyeceğimiz için Add'e 2 yazalım  
 wg.Add(2)  
 //fonksiyonlarımızın kullancağı bakiye değişkenimiz  
 var bakiye float64 = 100  
 fmt.Printf("İlk Bakiye: %.2f\n", bakiye)  
 \* paraÇek ve paraYatır fonksiyonlarımızı aynı anda başlatıyoruz.  
 \* hangisi daha önce başlarsa mutex sırasına ilk o girer. bu esnada diğer  
 \* fonksiyon mutex'in açılmasını bekler.  
 go paraÇek(&bakiye, 25, &wg)  
 go paraYatır(&bakiye, 65, &wg)  
 \* ana iş parçacığı tamamlandığında asenkron çalışan fonksiyonları beklemez.  
 \* beklemediğinde de asenkron fonksiyonlar çalışmadan program sonlanır.  
 \* ana iş parçacığının asenkron işlemleri beklemesi için waitgroup sonucunun 0 olmasını bekleriz.  
 \* wg.Add(2) yazarak 2 adet wg.Done() fonksiyonu çalıştığında wg.Add(0) olur ve  
 \* wg.Wait() tamamlanır ve program başka işlemler yapılmıyor ise sonlanır.  
 wg.Wait()  
}

# select

**Select** ile çoklu goroutine işlemlerinin iletişimini bekleyebiliriz.

package main  
import (  
 "fmt"  
 "time"  
)  
func main() {  
 k1 := make(chan string)  
 k2 := make(chan string)  
 go func() {  
 time.Sleep(time.Second \* 1)  
 k1 <- "video"  
 }()  
 go func() {  
 time.Sleep(time.Second \* 3)  
 k2 <- "ses"  
 }()  
 for i := 0; i < 2; i++ {  
 select {  
 case mesaj1 := <-k1:  
 fmt.Println("Mesaj 1:", mesaj1)  
 case mesaj2 := <-k2:  
 fmt.Println("Mesaj 2:", mesaj2)  
 }  
 }  
}

# En Çok Kullanılan HTML Kodları

HTML kodları, bir web sayfasının yapısını oluşturan parçacıklardır. Kullanılan kod ne olursa olsun tümü “**<**” ile başlar ve “**>**” ile bitmek zorundadır. Bazı basit HTML kodlarını daima bilmeniz gerekir. Hazırladığınız web sayfalarında bu kodları birden fazla yerde kullanmanız gerekecektir. Ancak öncesinde temel bir HTML belge yapısının nasıl olması gerektiğini bilmeniz gerekir. Buna göre;

**<!DOCTYPE html>:** Tüm HTML belgeleri bu kod ile başlamak zorundadır. Bu kod aslında sayfanın HTML5 formatına göre kodlandığını gösterir.

**<html>:** Kök etiket olarak da bilinen bu temel kullanımda tüm HTML kodları, bu iki etiket arasında tanımlanmak zorundadır.

**<body>:** Görünebilir tüm HTML kod parçalarının eklendiği etiket alanıdır.

**<head>:** Web sayfasının dili, başlığı ve diğer düzenlemelerinin yer aldığı etikettir.

**<title>:** Web sitesinin başlık alanını tanımlamak için kullanılır. <title> etiketi, <head> etiketleri arasında olmak zorundadır.

**<meta charset=”utf-8″>:** **HTML kodunun Türkçe karakter desteği dahil olmak üzere pek çok dil seti içerdiğini belirtir.**

* **Başlık Kodu (Hx)**

Web sayfanızdaki içeriği okunabilir parçalara bölmenizi sağlayan unsur başlıklardır. HTML temelde 6 başlık ögesini destekler. Bunlar **H1, H2, H3, H4, H5** ve **H6**‘dan oluşur. Örnek kullanım şekli şu şekilde gösterilir:

* **<h1>**Birinci başlık**</h1>**
* **<h2>**İkinci başlık**</h2>**
* **<h3>**Üçüncü başlık**</h3>**
* **<h4>**Dördüncü başlık**</h4>**
* **<h5>**Beşinci başlık**</h5>**
* **<h6>**Altıncı başlık**</h6>**
* **Paragraf Kodu (<p>)**

Metin içerisindeki paragrafları oluşturmanızı sağlayan koddur. İçerik bölümlerini ayırırken aralarında **<br>**kodu eklenerek metin paragrafları oluşturulabilse de **<p>** kodunun kullanılması tavsiye edilir. Arama motorlarını ve tarayıcıları sayfa yapısı hakkında bilgilendirir.

* **Satır Sonu Kodu (<br>)**

**Web sayfasındaki metni yatay olarak bölmeye yarayan HTML kodudur.** Kullanıldığı yerde metin bir alt satırdan devam eder. Diğer bir deyişle **<br/>**kullanıldığında, takip eden tüm kelimeler alt satırdan devam eder.

* **Resim Kodu (<img>)**

**<img> kodu, görüntüleri web sayfasına eklemek için kullanılır.** Web sayfasına resim eklemek istediğinizde önce web sunucusuna yükleyin. Daha sonra yüklenen resmin dosya adına başvuru yapmak için bir <img> etiketi kullanabilirsiniz.

* **Buton Kodu (<button>)**

**HTML buton kodu**, sayfalarınızı ilgi çekici yapmak ve etkileşimli içerikler oluşturmak için kullanılır. JavaScript ve **HTML renk kodları**ile daha güçlü hale gelen bu kod, içeriklerinizde kullanıcılarla etkileşime girmenizi sağlar.

* **Bağlantı Kodu (<a>)**

Önemli **HTML link verme** kodlarından biri olan <a> kodu ile diğer içerikleriniz arasında da bağlantılar oluşturabilirsiniz. Bağlantı oluşturmak için, bağlantı kurmak istediğiniz içeriğin çevresine **<a>**ve **</a>** etiketleri sarmanız gerekir.

* **Yatay Çizgi Kodu (<hr>)**

Sayfa bölümlerini görsel olarak ayırmada yatay çizgi kodları kullanılır. Bu kod, sayfadaki geçerli konum üzerinden sağ kenar boşluğu arasında bir çizgi oluşturur. Böylece kelime veya cümle grupları yatay çizgiyle ayrılır.

<a>   köprü

<abbr>   kısaltma

<adres>   adres eleman

<area>   resim haritası içinde belirtir  alan

<article>   makale

<aside> sayfa içeriği kenara içerik

<audio> ses içeriği

<b> kalın yazı

<base> sayfadaki tüm bağlantılar için belirtir  taban URL

<bdo> metin ekranı belirtir yönü

<blockquote> uzun tırnak

<body> body elementi

<br> ekler tek satır sonu

<button>   düğme

<canvas> grafik tanımlayın

<caption>   tablo başlığı

<cite>   alıntı

<code> bilgisayar kodu metin

<col> tablo sütunları için nitelikler

<colgroup> tablo sütunlarında belirtir grupları

<komut>   komut

<datalist>   “otomatik tamamlama” dropdown list

<dd>   tanım açıklama

<del> metin silindi

<details>   elemanın belirtir ayrıntıları

<dfn> Tanımlayan  terim anlamı

<div> belgede belirtir  bölüm

<dl>   tanım listesi

<dt>   terim anlamı

<em> metin vurguladı

<embed> dış uygulama veya etkileşimli içerik

<eventsource>   sunucu tarafından gönderilen etkinlikler için  hedef

<fieldset>   fieldset

<figcaption> için başlık belirtir şekil unsuru.

<figure> medya içeriği  grup ve onların başlık

<footer>   bölüm veya sayfa belirtir altbilgi

<form>   form

<h1>   başlık seviye 1

<h2>   başlık seviye 2

<h3>   başlık seviye 3

<h4>   başlık seviyesi 4

<h5>   başlık seviyesi 5

<h6>   başlık düzeyini 6

<head> belge hakkında bilgileri belirtir

<header> seyir yardımcıları veya  grubun tanıtım dahil group elemanları

<hgroup>   bölüm veya sayfa belirtir  başlık

<hr> yatay  kural

<html>   html belgesi

<i> italik yazı

<\*\*\*\*\*\*>   satır alt pencere (çerçeve)

<img>   görüntü

<input>   giriş alanı

<ins> metin takılı

<kbd> klavye metin

<keygen> üreten  anahtar çifti

<label> form kontrolü için belirtir  etiket

<legend>   fieldset belirtir  başlık

<li>   liste öğesi

<link>   kaynak referans

<mark> metin işaretli

<map>   resim haritası basın  menü listesi

<\*\*\*\*> \*\*\*\* bilgileri

<meter> Önceden tanımlanmış  kapsama alanı içinde belirtir ölçüm

<nav> gezinti bağlantıları

<noscript>   noscript bölüm

<object>   nesne

<ol>   sipariş listesi

<optgroup>   seçenek grubu

<secenek> açılan listede belirtir  seçenek

<output> çıkış belirtir bazı türleri

<p>   paragraf

<param>   nesneyi belirtir  parametre

<pre> metin biçimlendirilmiş

<progress> her türlü görevi belirtir ilerleme

<q> kısa  alıntı

<ruby>   yakut açıklama (Doğu asya tipografi kullanılan)

<rp> yakut açıklamaları desteklemeyen tarayıcılarda yararına kullanılır

<rt>   yakut emrin yakut metin bileşeni.

<samp> örnek bilgisayar kodu

# **Web Sitesi İçerik Girişinde Kullanılan HTML 4 5 Kodları ve Anlamları**

<!–…–> Açıklama yazmak için kullanılır  
<!DOCTYPE> Dosya tipini belirtir<a> Link belirtir  
<abbr> Yazının bir kısaltma olduğunu belirtir  
<acronym> Kelimenin başharfleri olduğunu belirtir  
<address> iletişim adresleri için kullanılır<applet> Gömülü aplet eklemekte kullanılır(html 4 ile bu özellik iptal edilmiştir)  
<area /> Resmin parçalarına ait link vermeyi sağlar  
<b> Koyu harfli yazı  
<base /> Tüm linkler için kaynak kök adresi belirtir<basefont /> Fon tipi belirtir (html 4 ile bu özellik iptal edilmiştir)  
<bdo> Yazının yönünü belirler  
<big> Büyük yazı tanımlar  
<blockquote> Uzun alıntı metin olduğunu gösterir

<body> Dosyanın ana kısmını gösterir  
<br /> Alt satıra geçiş işaretidir  
<button> Gönderme butonu üretir  
<caption> Tablo başlığı yazar  
<center> Elemanı sayfada ortalar (html 4 ile bu özellik iptal edilmiştir)  
<cite> Alıntı belirtir  
<code> Bilgisayar kodları belirtir  
<col /> Tablo sütunları için özellik verir  
<colgroup> TAblodaki bir grup sütuna özellik verir  
<dd> Tanım Listesi  
<del> Silinmiş yazı  
<dir> Klasör listesi belirtir (html 4 ile bu özellik iptal edilmiştir)  
<div> Dökümanda bir bölümü temsil eder  
<dfn> Tanım kelimesini gösterir  
<dl> Tanım listesi elemanı  
<dt> Tanım listesindeki elemanlardan biridir  
<em> Vurgulanmış yazı  
<fieldset> Form için çerçeve üretir  
<font> Yazı özellikleri belirtir (html 4 ile bu özellik iptal edilmiştir)  
<form> Bilgi yayıt formu<frame /> Sayfa içinde gömülü pencere açar  
<frameset> Sayfada bölme yapar  
<h1> ile <h6> arası Başlık yazısı belirler  
<head> Dökümanın temel başlık elemanlarını içerir  
<hr /> Yatay çizgi  
<html> sayfa türünü belirtir  
<i> Eğik yazı  
<iframe> Başka bir sayfayı dökümanda gösterir  
<img /> Resim  
<input /> Form için bilgi alanı  
<ins> Eklenmiş yazı  
<isindex> Dökümana ait arama kelimesi (html 4 ile bu özellik iptal edilmiştir)  
<kbd> Klavye yazısı  
<label> Form için metin etiketleri  
<legend> Form çerçevesi için açıklama  
<li> Liste elemanı  
<link /> Dökümana dışardan bir kaynak bağlar  
<map> resim için parçaları tanımlar  
<menu> Menü listesi (html 4 ile bu özellik iptal edilmiştir)  
<meta /> Dökümana ait önemli tanımlamaları içerir  
<noframes> Frame özelliği kapalı ise gösterilecek yazı  
<noscript> Javascript kapalı isse gösterilecek yazı  
<object> Gömülü uygulamalar  
<ol> Düzenli Liste  
<optgroup> Seçim listesindeki alkalı bileşenler  
<option> Seçim listesindeki elemanlar  
<p> Paragraf başı  
<param /> Gömülü uygulamalar için ayar kısmı  
<pre> Düz yazı biçimi  
<q> Kısa alıntı  
<s> Üseri çizili yazı (html 4 ile bu özellik iptal edilmiştir)  
<samp> Örnek bilgisayar kodu gösterir  
<script> Kullanıcı taraflı uygulama  
<select> Seçim listesi  
<small> Küçük yazı  
<span> Dökümanda bir bölümü gösterir  
<strike> Üzeri çizili ayzı (html 4 ile bu özellik iptal edilmiştir)  
<strong> Kalın yazı  
<style> Elemanların görünümünü belirten şekiller  
<sub> Alt yazı  
<sup> Üst yazı

<table> Tablo  
<tbody> Tablo grubu  
<td> Tablonun bir gözü  
<textarea> Form için uzun yazı alanı  
<tfoot> Tablonun dip kısmı  
<th> Tablodaki başlık alanı  
<thead> Tablo başlık grubu  
<title> Dökümanın ana başlığı  
<tr> Tablonun bir satırı  
<tt> özel metin  
<u> Altı çizili yazı (html 4 ile bu özellik iptal edilmiştir)  
<ul> Düzensiz liste  
<var> Yazı içinde değişken tanımı  
<xmp> Ön tanımlı yazı (html 4 ile bu özellik iptal edilmiştir)

