**Rbenv Nedir?**

[rbenv](https://github.com/rbenv/rbenv#readme), Ruby’nin birden çok kurulumunu yönetmenize imkan sağlar. Varsayılan olarak Ruby’yi kuramasa da [ruby-build](https://www.ruby-lang.org/tr/documentation/installation/#ruby-build) eklentisi bunu yapabilir. Her iki araç da macOS, Linux ve diğer UNIX benzeri işletim sistemlerinde kullanılabilir.

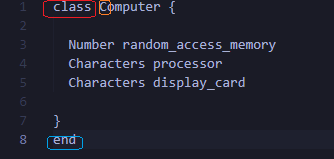
### RVM (“Ruby Version Manager”) Nedir?

[RVM](http://rvm.io/), sisteminizde Ruby’nin birden çok kurulumunu kurup yönetmenize imkân sağlar. Ayrıca farklı gem kümelerini yönetmek için de kullanılabilir. macOS, Linux, ve diğer UNIX benzeri işletim sistemlerinde kullanılabilir.

**Ruby Class- Object**

Ruby nesne yönelimli bir programdır ve nesne yönelimli bir program sınıf, nesne içerir.

Sınıf tek tek nesnelerin oluşturulduğu yerdir. Yani mesela bilgisayarları düşünelim. RAM, işlemci, ekran kartı, SSD kapasitesinden oluşur. Bunlar Bilgisayar sınıfının verileridir. Bunlar sayesinde bilgisayarlar arasında seçim yapabiliriz.



(Burada bilgisayar adında bir sınıf oluşturdum ve sınıfın içerisine ayırt edilebilecek özelikleri ekledim.)

Rubyde bir sınıf oluştururken önce Class yazmamız gerekiyor. Sonrasında sınıfa vereceğimiz ismi yazarız. Ve bu isim her zaman büyük harfle başlar, yukarıdaki örnekte gösterdiğim gibi. End kelimesi ile de sınıfı sonlandırabiliyoruz. Yani tüm veriler class ile end arasında kalır.

Nesne sınıfın örnekleridir. Nesneleri new yöntemi kullanarak oluştururuz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada stud1 ve stud2 nesnelerin adıdır. Eşittir işaretinden sonrası sınıf adıdır.

**Ruby Değişkenler**

Herhangi bir veriyi tutan bellek konumlarına değişken denir. Ruby de ise 5 tür değişken vardır.

*1- Ruby Global Değişkenleri:* Global değişkenler $ ile başlar. Başlatılmamışsa nil değerine sahip olur ve -w seçeneğiyle uyarılar üretir. Kullanılması önerilmez. Bir örnekle inceleyelim.



Şekil

Çıktısı bu şekilde olur.

metin, iç mekan, portakal, karanlık içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil

Kırmızı kutu içerisindeki (3.satırdaki kod) ‘$global\_veriable’ değişkendir. Bu şekil2 deki sonucu üretecektir.

Sarı kutu içerisindeki (5.satırdaki) def ile metot tanımlaması yapıyoruz. Yani Ruby’ye adı print\_global olan bir metot tanıtımı başlattığımızı bildirir.

Mavi kutu içerisindeki (6.satırdaki) puts ile de yazdırma işlemi yapıyoruz daha sonra end ile metot tanımlamasını bitirdiğimizi söylüyoruz.

*2-Ruby Örnek Değişkenleri: Bu* değişkenler @ ile başlar.



Mavi kutu içerisindeki @stud\_id, @stud\_name ve @stud\_addr örnek değişkenlerdir.

*3-Ruby Sınıfı Değişkenleri:* Bu değişkenler @@ ile başlar. Başlatılmamış bir sınıf değişkenine başvurmak bir hata üretir. Bu yüzden yöntem tanımlarında kullanılmadan hemen önce başlatılmalıdır. Sınıf değişkenleri, sınıf değişkenlerinin tanımlandığı sınıfın veya modülün alt öğeleri arasında paylaşılır.



Sınıf değişkenine bir örnek.

*4-Ruby Yerel Değişkenler:* Yerel değişkenler küçük harfle veya \_ ile başlar. Kapsamı, sınıf, modül, def veya do’dan karşılık gelen sona veya bloğun açılış ayracından kapanış ayracına {} kadar değişir.



Buradaki id, name, addr yerel değişkene örnektir.

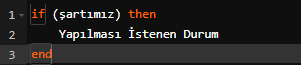
*Yakut sabitleri:* Bu sabitler büyük harfle başlar. Bir sınıf veya modül dışında tanımlanan sabitlere global olarak erişilebilir.

metin içeren bir resim

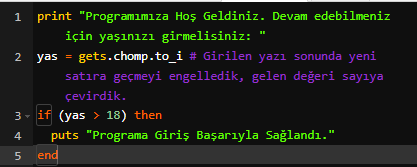
Açıklama otomatik olarak oluşturuldu

Burada VAR1 ve VAR2 sabitlerdir.

***Ruby if…else İfadesi:*** *if else yapısı istenilen şartın gerçekleşmesi durumunda yapılması istenen ya da istenen durumun gerçekleşmediği durumda yapılması istenen yerlerde kullanılır.*

**

Parantez içinde şartımız belirlenir. Then kullanımı isteğe bağlıdır. Kullanım nedeni kafa karışıklığını önlemek için nerede bittiğini belirtiriz. End koşulun sonlandığı yerdir. İf ile end arasında yazılan tüm kodlar şart istenilen gibi olduysa gerçekleşir.



Programı kullanmak isteyen kullanıcıdan yaş girmesini istedik. Eğer yaşı 18’den büyükse giriş yapabilir. Ve ekrana “Giriş Başarılı” ifadesi yazsın istedik.

Eğer yaşı 18’den küçükse ve şart gerçekleşmezse diye else yapısını kullanacağız.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Else yapısı if şartı sağlandığındaki çalışan kodlardan hemen sonra yerini alır ve end den önce gelir. Şartlar sağlanmadığında if kodlarına bakmaksızın else yapısında bulunan kodları çalıştıracaktır.

Elsif de olay şöyle, if koşulu gerçekleşmezse şayet elsif satırı kontrol edilir. Eğer bu satırda istenilen durum gerçekleşirse istenilen kodlar çalışır. Bu durumda da istenilen gerçekleşmezse else yapımızda bulunan kodlar ile devam ederiz. İstenilen miktarda elsif şartı girilebilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Ruby-Loops:**

Ruby’de döngü oluşturmanın en basit yolu loop kullanmaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu örnekteki gibi kullanım pek kullanışlı değildir çünkü sonsuz bir döngüye neden olur. Break adlı bir dosya oluşturarak daha kullanışlı hale getiririz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Break, herhangi bir noktada döngüden çıkmamıza izin verir. Break ile programdan çıkmayız sadece döngüden çıkar ve program kaldığı yerden devam eder.

**While Döngüsü**:

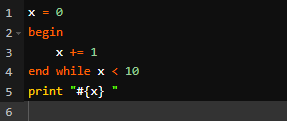
İstenilen şart doğru(true) ise döngü içerisindeki kod tekrar tekrar çalışır.

metin, bağlı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kodumuz mevcut sayıyı yazdırdıktan sonra “x += 1” komutu ile sayımızı bir artırıp bir sonraki adıma geçiyor. Sayımız son adımda tekrar artırılıp 11 olduğunda while döngüsünün şartını sağlamadığı için döngüye girmez ve döngü sonlandırılır.

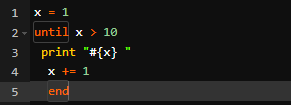
Javada do-while ın ruby’deki karşılığı begin dir.



Kodumuz döngüye girmeden önce bir kez istenilen durumu çalıştıracak ve sonra döngünün şartına uyup uymadığını kontrol edecektir.

**Until Döngüsü:**

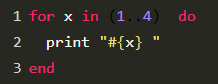
Bunu mantık olarak while ın tersi olarak düşünebiliriz. While da şartın doğruluğu sağlandığı takdirde döngümüz çalışırken until de tam tersi durumda yani şartımız yanlış (false) değer aldığında döngü çalışır.



Ancak bu sefer döngü şartımız bahsettiğimiz gibi yanlış(false) değer alıyor bu sebepten döngü çalıştırılıyor. x değeri en son adımda 11 değerini aldığında şartımız doğru değer verdiğinden döngü sonlanıyor.

**For Döngüsü:**

For döngüsünün sonuç değeri, break kullanılmadığı sürece yinelenen değerdir. Ruby de for çok nadir tercih edilir. Bunun yerine each kullanılır.



Çıktımızda for döngüsü, kendisine şart olarak verilen bir aralıktaki değerleri esas alarak çalışır.

**Next Durumu:**

Bir döngüde yinelemenin mevcut durumunun kalanını es geçmek için next anahtar sözcüğünü kullanırız.

**Ruby Yineleme Bildirimi:**

Bunu redo komutu ile yaparız. Döngü koşulunu kontrol etmeden en dahili döngünün bu yinelemesini yeniden başlatır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada çıktı sonsuz döner.

**Ruby Yeniden Deneme Bildirimi:**

Retry kullanılarak yapılır. Blokta veya for ifadesinin gövdesinde yeniden deneme görünürse, yineleyici çağrısının çağrısını yeniden başlatır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu