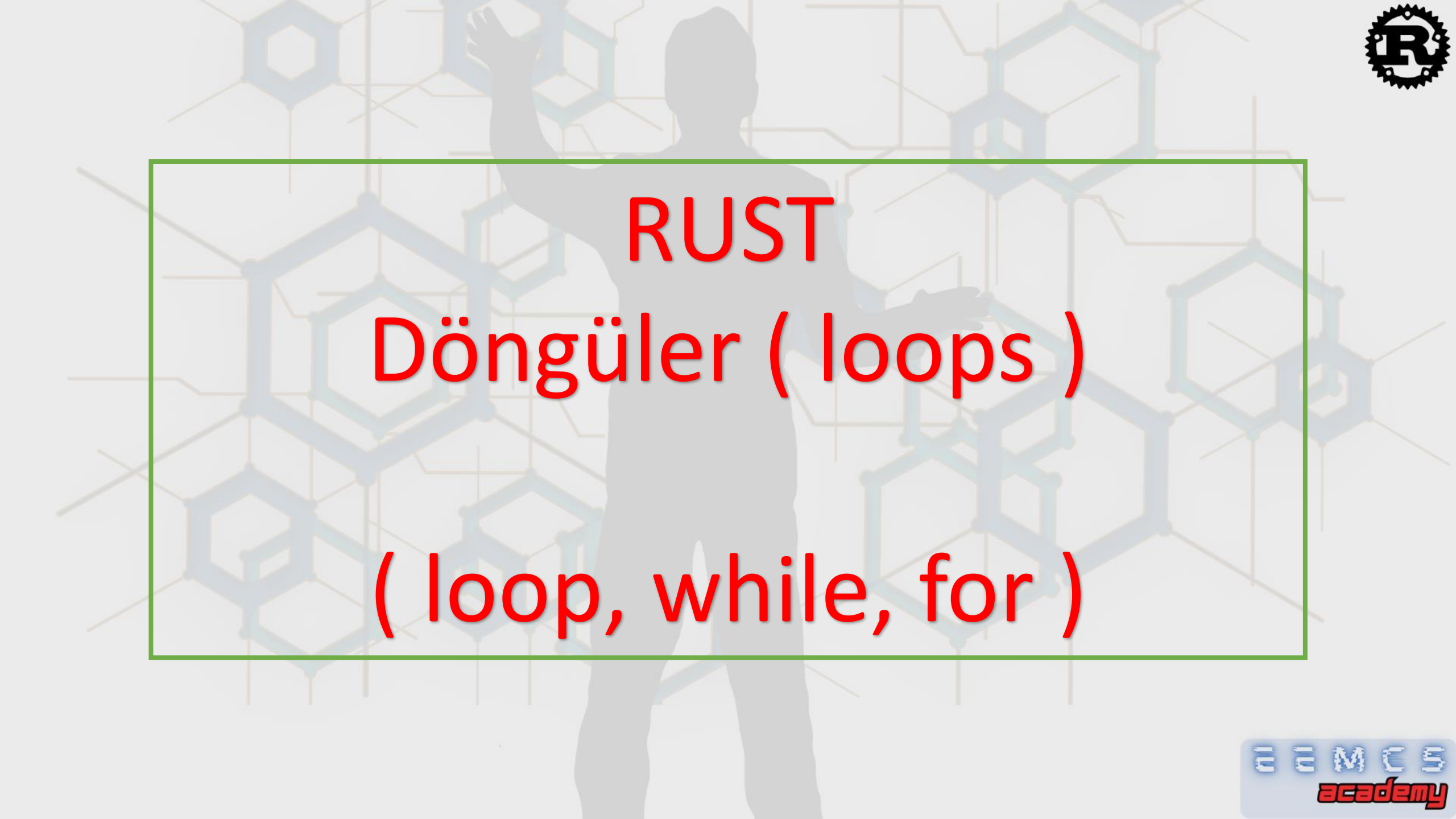
RUST
Döngüler (loops)
(loop, while, for)

Döngüler - loops

Bir kod bloğunun birden fazla çalıştırılması

Aynı işi birden fazla kez yapmak

Çoklu veri yapılarının üzerinde çalışmak

vb.

loop

```
loop {
    println!("Merhaba Rust");
}
```

Sonsuz döngü - infinite loop

Kırmak için Ctrl + C tuş kombinasyonunu kullanın

Döngünün Çalışma Sayısını Kontrol Etmek ve Döngüden Çıkmak (sayaç)

Sayaç - Counter → Döngünün çalışma sayısını saymak için kullanılır.

→ Başlangıç değeri olması gerekir. Bu değer herhangi bir sayı olabilir.

let mut sayac = 0 let mut sayac = 10

→ İleriye (başlangıç değerinden itibaren arttırılması) veya geriye (başlangıç değerinden itibaren azaltılması) doğru değer değiştirilir.

→ Arttırma veya azaltma değerinin olması gerekir.

sayaç += 1 (sayac = sayac + 1)

sayac -= 1 (sayac = sayac - 1)

→ Döngüden önce tanımlanmalıdır.

Döngünün Çalışma Sayısını Kontrol Etmek ve Döngüden Çıkmak (break)

Döngüyü kırma/bitirme - Break →Bir koşulun sağlanması durumun döngünün çalışmasını durdurmak için kullanır.

Döngünün Çalışma Sayısını Kontrol Etmek ve Döngüden Çıkmak (break ve sayaç)

```
let mut counter = 0;

loop {
    counter += 1;

    println!("Merhaba Rust - {} kez yazıldı", counter);

    if counter == 10 {
        break
    }
};
```

```rust
    let dizi = [20, 30, 40];

    let mut index_no = 0;

    loop {
        println!("Dizinin {} elmanı {}", index_no,
dizi[index_no]);
        index_no = index_no + 1;

        if index_no == dizi.len(){
            break;
        }
    }
```

Döngünün Çalışma Sayısını Kontrol Etmek ve Döngüden Çıkmak (break ve sayaç)

Döngü kırılırken;

Sayaç arttırma ve azaltma kodunun,
Yapılan iş için yazılan kodun

Döngüyü kırmadan önce mi yoksa sonra mı yazılacağına dikkat etmelisiniz.

Döngüde veri üretmek ve veri almak

```
let mut counter = 0;

let result = loop {
    counter += 1;

    if counter == 10 {
        break counter * 2;
    }
};

println!("The result is {result}");
```

break değiminden sonra yazılan değişken ya da işlem result değişkenine atanır

Döngüde veri üretmek ve veri almak

break değiminden sonra yazılan değişken ya da işlem result değişkenine atanır

Bu işleme aslında " Return Value " değer döndürme denir. Yani kod bloğunda üretilen bir verinin kod bloğu dışında (ana programda da diyebiliriz) kullanılabilmemizi sağlar. Bunun içinde döngü bir değişkene atanarak oluşturulur.

Buradaki kullanımın dışında değer döndüren bütün yapılar bir değişkene atanarak kullanılır. Ve böylece o yapının döndürdüğü değer (başka bir değişle ürettiği veri) ana program içerisinde kullanılabilir. Bu durum fonksiyonlarda ya da metotlarda daha çok mevcuttur.

while - Koşullu Döngü

```
while koşul-yada-bool_değişken {

        // koşul true olduğunda kodlar çalışır
        // döngünün durması için koşulun false olması gerekir ya da break kullanılır
}
```

loop, if, else ve break kullanarak oluşturulabilecek yapılarda kullanılarak sadelik sağlar

while - Koşullu Döngü

```
let mut sayac = 5;

while  sayac != 0 {
    println!("Merhaba Rust");
    sayac -= 1;

};
```

for - Çoklu Veri Yapıları İçin Döngü Oluşturma

```
for tek_veri_için_değişken in çoklu_veri_tutan_değişken {

        // birden fazla veri tutan değişkendeki her tek veri için buradaki kodlar çalışır
        // max döngü sayısı birden fazla veri tutan değişkenin eleman sayısı kadardır.

        // kodlar yazılırken tek bir veri üzerinde işlem yapılacağına dikkat edilmelidir
        // çoklu_veri_tutan_değişken içinde bulunan veriler sırayla
tek_veri_için_değişken e atanır ve bu değişken kullanılarak kodlar yazılır.

}
```

çoklu_veri_tutan_değişken
let dizi = [10,20,30,40]

tek_veri_için_değişken → eleman

eleman üzerinde yani döngüde yapılacak işlem

Karesini al → eleman*eleman

DÖNGÜ NO	Eleman değişkeninin değeri	Çıktı
1	10	100
2	20	400
3	30	900
4	40	1600

```rust
let a = [10, 20, 30, 40, 50];

for element in a {
    println!("the value is: {element}");
}
```

Ayrıca döngü içerisinde çoklu yapının index numarası da kullanılabilir.

Bu kullanımda index değişkeni sayaç yerine de geçer.

let mut index = 0 → döngü dışında

index += 1 → döngü içinde

Aralık belirtmek → (1..4)

Çoklu yapı için aralık belirtebiliriz.
Son değer dahil değildir (örnek için 4 dahil değildir.)

rev() → Aralığı tersten işlemek için kullanılmaktadır.

```
for number in (1..4).rev() {
    println!("{number}!");
}
println!("LIFTOFF!!!");
```

Hesap Makinesi Örneği : Döngü ile programın düzenlenmesi

EEMES
academy

```rust
use std::io::*;
#[allow(unused_variables)]
fn main() {

    loop {
        // Kullanıcı verilerini almak için kullanılan değişkenlerin tanımlanması
        let mut kullanıcı_verisi1 = String::new();
        let mut kullanıcı_verisi2 = String::new();

        // Kullanıcıya program ile ilgili bilgilerin gösterilmesi
        println!("Bu program basit bir hesap makinesidir.");

        let mesaj = r####"
            Burada yapılan işlemler
            1 - Toplama
            2 - Çıkarma
            3 - Çarpma
            4 - Bölme
        "####;

        println!("{mesaj}");

        // Kullanıcıdan yapması gereken işlemi seçme kısmı
        let mut islem = String::new();

        println!("Yapmak istediğiniz işlemin mumarasını giriniz . ( 1/2/3/4 )");

        stdin().read_line(&mut islem).expect("hata");

        let islem:u16 = islem.trim().parse().expect("hata");

        // Kullanıcı veri girişinin yapılması
        println!("Birinci sayıyı giriniz : ");

        stdin().read_line(&mut kullanıcı_verisi1).expect("hata");

        let sayı1:u32 = kullanıcı_verisi1.trim().parse().expect("hata");

        println!("İkinci sayıyı giriniz : ");

        stdin().read_line(&mut kullanıcı_verisi2).expect("hata");

        let sayı2:u32 = kullanıcı_verisi2.trim().parse().expect("hata");

        // İşlemlerin yapılması
        if islem == 1 {
            let islem_sonucu = sayı1 + sayı2;

            println!("Toplama işleminin sonucu {} ",islem_sonucu);
        } else if islem == 2 {
            let islem_sonucu = sayı1 - sayı2;

            println!("Çıkarma işleminin sonucu {} ",islem_sonucu);
        }

        if islem == 3 {
            let islem_sonucu = sayı1 * sayı2;

            println!("Çarpma işleminin sonucu {} ",islem_sonucu);
        }

        if islem == 4 {
            let islem_sonucu = sayı1 / sayı2;

            println!("Bölme işleminin sonucu {} ",islem_sonucu);
        }

        let mut cevap = String::new();
        println!("Devam etmek istiyor musunuz? Programdan çıkmak için h yazınız!");
        println!("Devam etmek için enter tuşuna basın");
        stdin().read_line(&mut cevap).expect("hata");
        //println!("kullanıcı cevabı  =  {cevap}");
        if cevap.trim() == String::from("h"){
            break;
        }
    }
}
```

1 - While ile döngüyü kurmak için sadece loop { satırını değiştirmemiz yeterlidir.

```
while true {
```

2 - While ile döngüyü kurmak için loop satırını kaldırın aşağıdaki satırları ekleyin. Ve programın sonundaki if bloğunu silin.

```
    let mut cevap = String::new();

    while cevap.trim() != String::from("h")  {
```

Aşağıdaki bloğu silin.

```
        if cevap.trim() == String::from("h"){
            break;
        }
```

Celal AKSU
Bilişim Teknolojileri Öğretmeni

celalaksu@gmail.com

https://www.linkedin.com/in/cllaksu/

https://twitter.com/ksacll

https://www.youtube.com/@eemcs