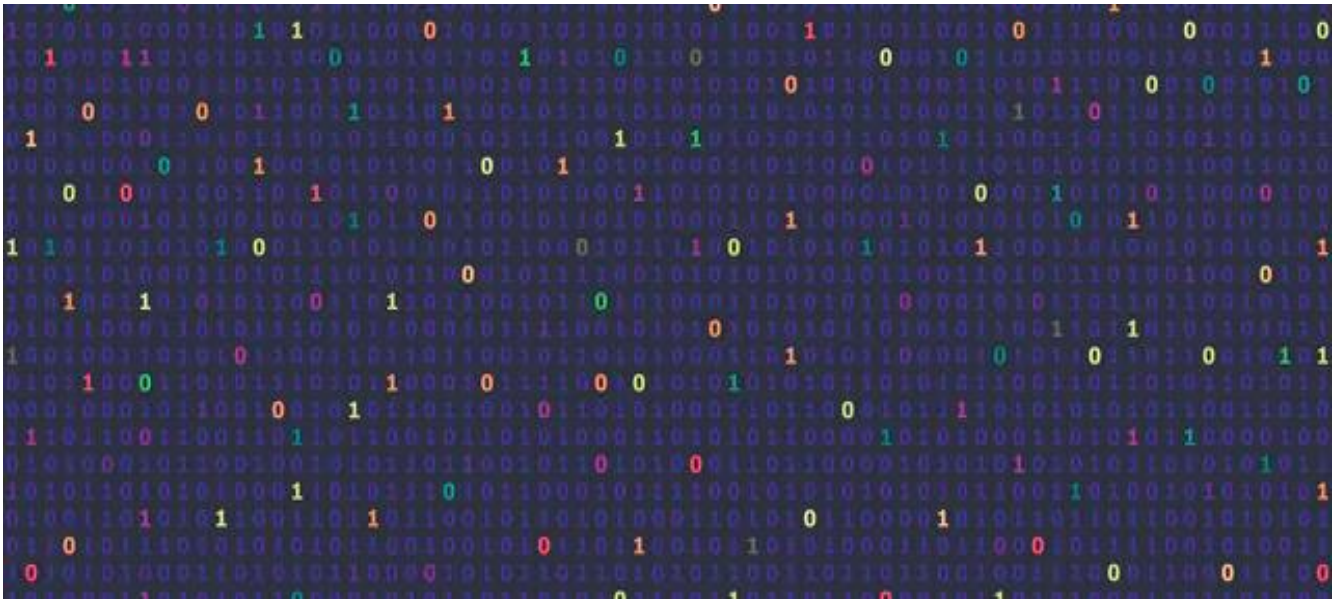
PROGRAMLAMA KAVRAMLARI



Code for Everyday – Her Güne Bir Kod Proje Grubu

Ayşe Büşra SÖKER – Selda MUTLU – Sevilay KOCAPINAR – Yasemin Can SİVAZ
Hüseyin DÖRTLÜ – Beyhan ÇITAK

Algoritmalar

Bilgisayarlarımızın işlevsel şekilde çalışabilmesi için gerekli donanım parçalarının ve yazılımların olması gerekir. Peki nedir donanım ve yazılım?

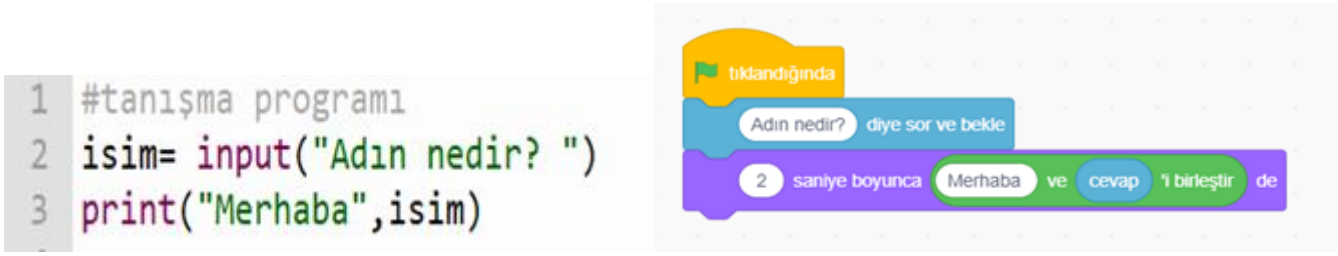
Bilgisayar sistemini oluşturan tüm elektronik parçalara (ekran, kablo, klavye, işlemci vb.) **donanım** denir.

Yazılımlar ise donanım parçalarını istediğimiz amaçlarda kullanabilmemizi sağlayan programlardır. Yani resim yapmak için kullandığınız paint uygulaması, kullandığınız kodlama araçları (scratch, mblock), internete girerken kullandığınız tarayıcılar, oynadığınız oyunlar hepsi birer yazılımdır.

Bu yazılımları oluşturabilmek için bir programlama dili kullanmak gerekir.

Programlama dili; Kullanıcının içerisine çeşitli talimatlar (kodlar) yazarak yeni programlar üretebileceği komut yapılarıdır. Her programlama dilinin kendine ait kod yapısı ve kuralları vardır.

Bilgisayar, telefon ya da başka elektronik cihazları kullanmak için gerekli tüm programlar, yazılımlar ve uygulamalar; programlama dilleri sayesinde üretilir. Bu nedenle programlama bilmek günümüzde son derece önemlidir. C, Java, Scratch, JavaScript, Python programlama dillerine birer örnektir.



Programlama dilleri yabancı bir dil gibidir. Genellikle programla dillerine ait kodlar İngilizce bazı kelime veya eklerden oluşur. Yukarıdaki resimde sol tarafta Python programlama diline ait kodlar görünmektedir. Sağ tarafta ise Scratch blok tabanlı kodlama programı görünmektedir. Scratch gibi blok tabanlı programlama dillerinde programlamaya yeni başlayanlar için kodlar hazır kod blokları şeklinde verilir, kullanıcının amacına uygun şekilde bu kod bloklarını sürükle bırak yöntemiyle bir araya getirmesi beklenir.

Algoritma Nedir?

Belirli bir problemi çözmek veya bir amaca ulaşmak için adım adım tasarlanan yoldur.

Günlük hayatta yaptığımız hemen hemen her işi farkında olmasak da belli bir sırayı takip ederek adım adım yaparız. Örneğin sabah erkenden okula gideceksiniz. Yataktan kalkıp, kapıyı açıp ayakkabılarınızı giymeye kalkmazsınız değil mi? Yataktan kalkarsınız, elinizi yüzünüzü

yıkar, üstünüzü değiştirir, kahvaltınızı yapar gerekli eşyalarınızı alır ve okula doğru yol alırsınız. Hatta düşününce bu adımları daha da ayrıntılandırabilirsiniz. Yani aslında algoritmalar yaşamımızın bir parçasıdır.

Örneğin anneniz size kahvaltı için yumurta haşlayıp, soyup, tabağınıza koydu. Bunun için farkında olmadan aşağıdaki algoritmayı kullanır.

Amaç: Yumurta haşlamak

BAŞLA

Adım 1: Küçük bir cezvenin içerisine su koy.

Adım 2: Yumurtayı cezvenin içerisine koy.

Adım 3: Ocağı aç.

Adım 4: Cezveyi ocağa koy.

Adım 5: Su kaynamaya başladıktan bir süre sonra ocağı kapat.

Adım 6: Yumurtayı soğuk su içerisine koy.

Adım 7: Yumurtayı soy.

Adım 8: Soyulmuş yumurtayı tabağa koy.

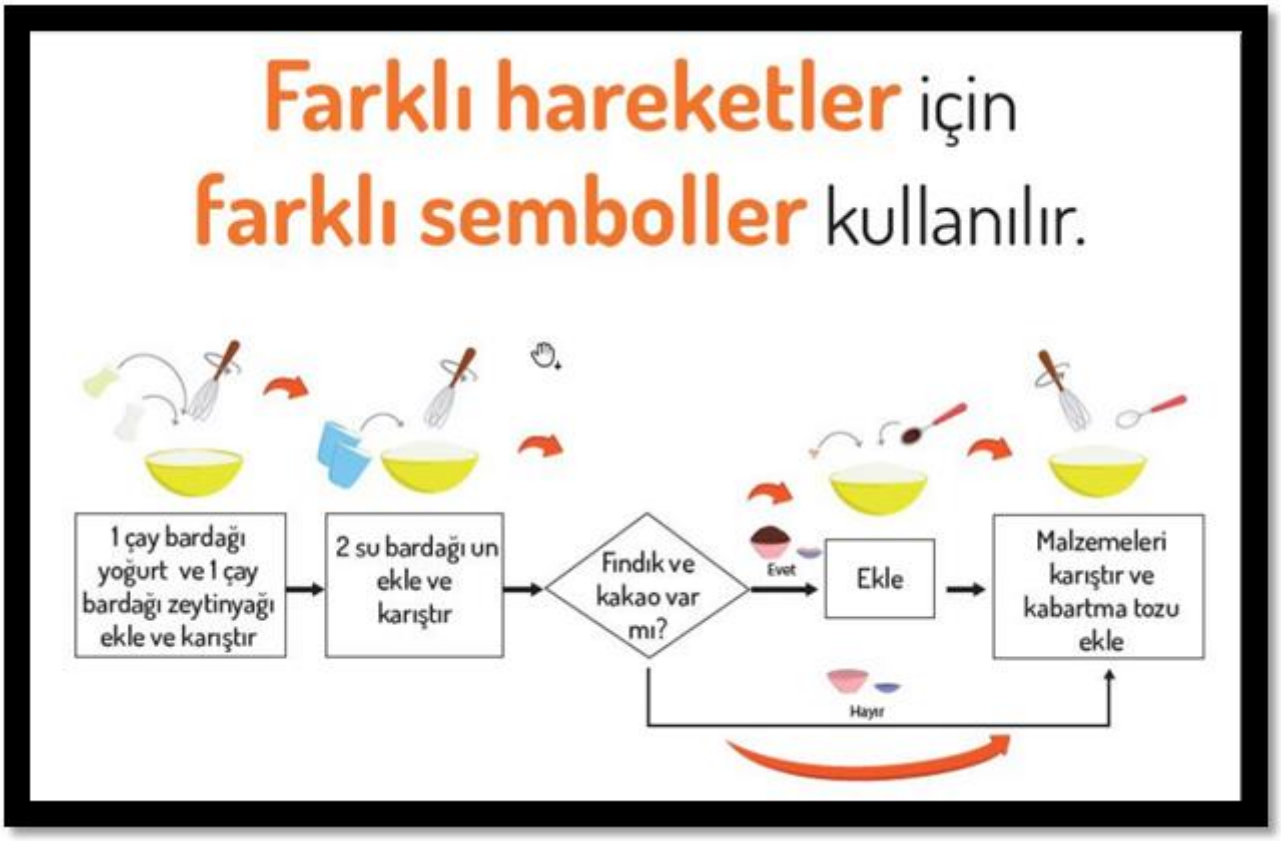
BİTİR

Yukarıdaki yönergeler yumurta haşlamak için tasarlanan bir algoritmadır. Bu algoritmadaki işlem adımlarının sırasının önemli olduğunu ve sıra karıştırılırsa işlemi gerçekleştiremeyeceğimizi unutmayınız. Günlük hayattaki işlerimiz ve problemlerimiz için bu şekilde algoritmalar kullandığımız gibi bilgisayarlar, yazılımlar, web siteleri, robotlar ve dijital olan diğer araçlar algoritmalar yardımıyla programlanarak çalışırlar.

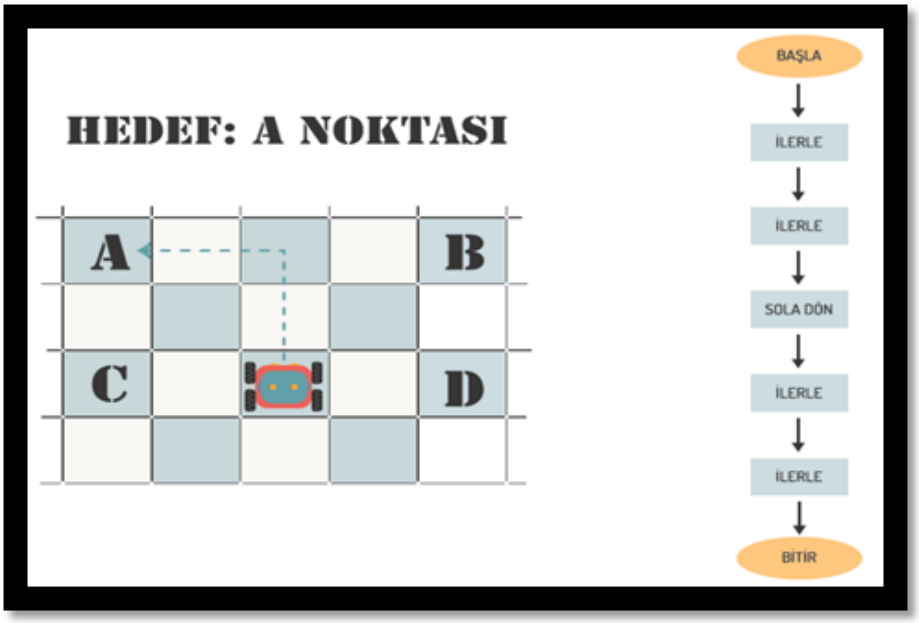
Akış Şeması Nedir?

Bir problemin çözümüne yönelik oluşturduğumuz algoritmaları adım adım tanımak ve programlarken hangi işlemleri yapacağımızı anlamak için akış şemalarından yararlanırız. Akış şeması; bir sürecin adımlarını görsel ya da sembolik olarak gösterir. Her sembolün (Elips, kare, Paralelkenar…) bir anlamı vardır.

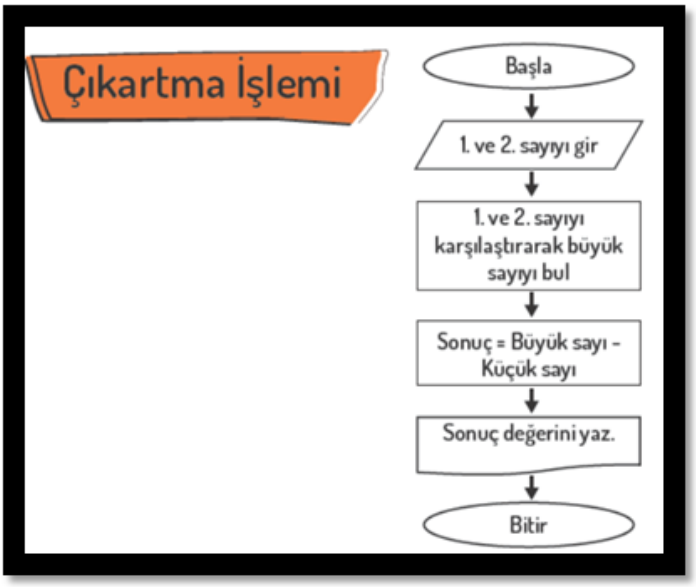
Örnek Akış Şemaları:



* Arabanın A noktasına gitmesini sağlayan algoritmaya ait akış şeması:



*2 sayıyla yapılan bir çıkarma işlemini gerçekleştirmek için gerekli algoritmaya ait akış şeması:



*Basit bir problemi Scratch ya da başka bir programlama dilinde çözerken akış şemasını oluşturarak kolay bir şekilde kodlarız. Karmaşık bir problemde ise çözüm için problemi alt problemlere bölmeli ve bu alt problemlerin algoritmalarını oluşturup çözerek bütüne yani problemin tamamının çözümüne daha kolay bir şekilde ulaşabiliriz.

Problem çözümünde algoritma kullanırsak ne olur?

- Problemleri daha hızlı ve sistematik olarak çözeriz.
- Problem çözme sürecini takip ederiz ve nerede hata yapıldığını görebiliriz.
- Tüm olasılıkları gözden geçirebiliriz.
- Hatalı işlem yapma olasılığımızı azaltırız.
- Olası hatalarımızı düzeltebiliriz.
- Çözüme ulaşmak için farklı yolları deneyebiliriz.
- Problemin çözümü için harcayacağımız süreyi kısaltırız.

Kurduğumuz algoritmalarda en kısa, en hızlı ve en etkili şekilde problemi çözmeye çalışmalıyız. Çünkü programlamada gereksiz kod yazımı hem zaman kaybı olacak hem de programımızın daha yavaş çalışmasına neden olacaktır. Ancak tabi ki kısa yoldan yapacağız diye gerekli hiçbir adımı atlamamamız gerekir. Amacımız gerekli adımlara yer vererek uzatmadan problemi çözmek olmalıdır.

Etkinlik 1: Yapabildiğiniz bir yemek varsa o yemeği hazırlarken yaptığınız adımlara dair bir algoritma oluşturunuz.

Etkinlik 2: Kullanıcının girdiği 2 sayıyı toplayan ve sonucu kullanıcıya gösteren bir algoritma oluşturunuz.

Etkinlik 3: Kullanıcının doğum tarihini yıl olarak girdiği ve bu yıl üzerinden yaş hesaplaması yapan bir programa dair algoritmayı oluşturunuz.

Veri Tipleri

Veri Nedir: Bilgisayarların sonuca ulaşabilmek için algıladığı, işlediği, sonuç ürettiği veya daha sonra kullanmak üzere depoladığı her şeye veri denir.

Veri, işlenmemiş ham gerçeklerdir. Sayılar, harfler işlenmemiş her türlü gerçekler veridirler.

Örneğin; 150 cm ifadesi bir veridir.

Bilgi Nedir: verilerin işlenerek anlamlı bir hale gelmiş halidir. Bir başka deyişle bilgi: Araştırma, gözlem ve benzeri öğrenme yolları ile elde edilen gerçeklerdir. Yukarıdaki veri değiştirilerek ağacın boyu 150 cm'dir dersek artık 150 cm ifadesi bizim için bir bilgi olur.

Başka bir örnek:

Öğrencinin sınavda her sorudan aldığı puanlar verileri, toplam 100 üzerinden aldığı puan da bize bilgi verir.

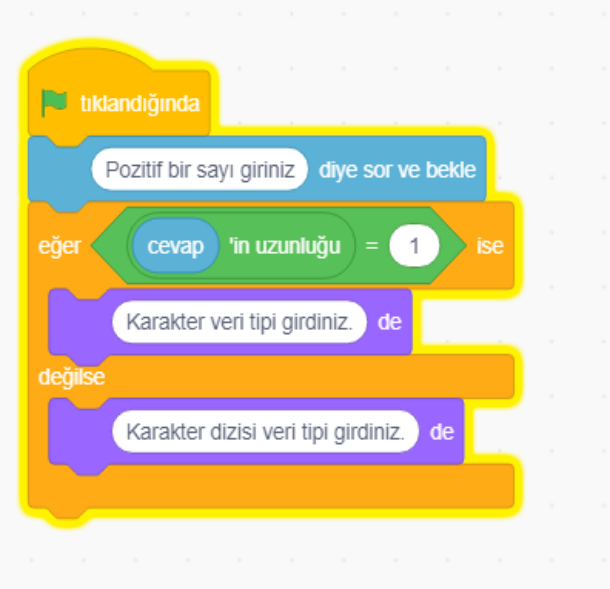
Karakter Veri Tipi: 0'dan 9'a kadar olan rakamları, harfleri ve özel karakterleri kapsar. Tırnak içinde belirtilir. Scratch sözcüğündeki "S" harfi bir karakteri ifade eder.

Örneğin:

İsminizin baş harfi nedir? Soyadı hangi harfle başlıyor? gibi soruların cevabı tek haneli olduğu için karakter veri tipidir.

Karakter Dizisi Veri Tipi: Birden fazla karakterin bir araya gelmesiyle oluşan veri tipleridir.

Örneğin: En sevdiği arkadaşı kim? Hangi şehirde yaşıyorsunuz? Boyunuz kaç cm? gibi soruların cevabı karakter dizisi veri tipine girer.



Örnekte girilen sayının tek haneli olması durumunda karakter veri tipi; tek haneli olmaması durumunda karakter dizisi veri tipi olduğu mesajı veriliyor.

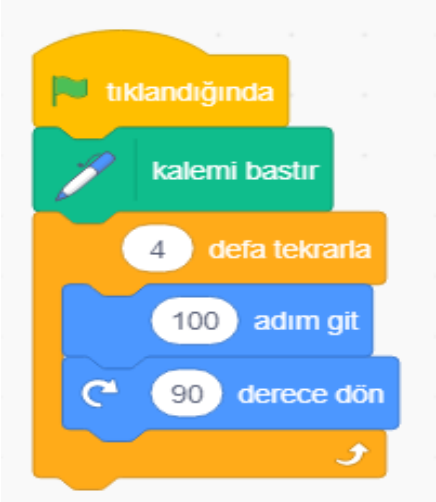
Mantıksal Veri Tipi: Evet - Hayır şeklinde karar verme süreçlerinde kullanılan veri tipidir.

Örneğin : Öğrenci mi? Arabası var mı? Elma sever misin? gibi soruları evet veya hayır şeklinde cevapladığımızda mantıksal veri tipine girer.

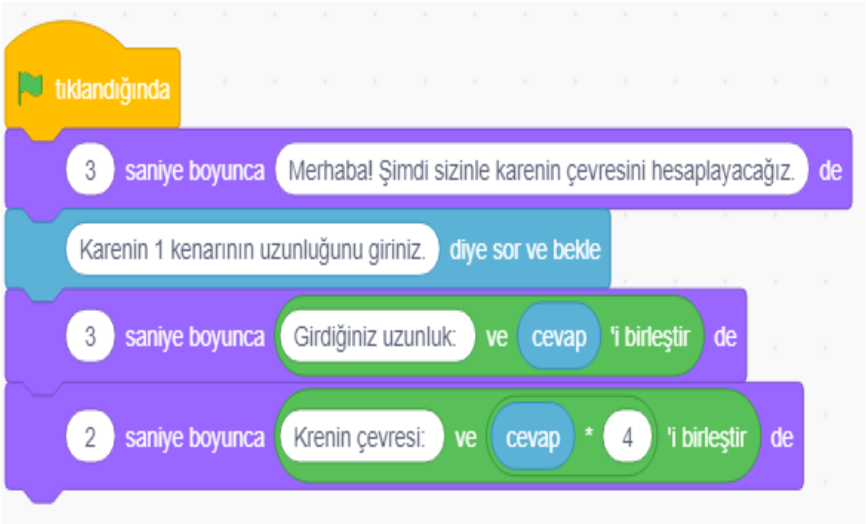
Yandaki örnekte Okula gidiyor musun? sorusuna E veya H şeklinde cevap verilmesi bekleniyor. Giriş farklı yapılırsa hatalı giriş yaptığı için bilgilendiriliyor.

Sayısal Veri Tipi: Tüm sayı çeşitlerini içeren veri tipidir. Pozitif, negatif, tamsayı ya da ondalık sayı olabilir. Örneğin; açılar, yaş, boy, uzaklık, nüfus, ücret, yarıçap, yükseklik gibi veriler sayısal verilerdir.

Kaçıncı sırada oturuyor? Boyu kaç cm? Kaç kilogram? gibi soruların cevabı sayısal veri tipine girer.



Yandaki örnekte kalemle 100 birim uzunluğunda çizgi çizip 90 derecelik açı ile sağa dönülen ve bu işlemi 4 defa tekrarlayan bir program görüyorsunuz.



Yandaki örnekte karenin 1 kenarının uzunluğunun kullanıcı tarafından girilmesi isteniyor. Ardından 4 ile çarpma işlemi yapılıp ekrana yazdırılıyor.

Özel Veri Tipi: Tarih, saat, hesap numarası, araba plakası, kimlik numarası, okul numarası, ev adresi gibi veriler özel veri tipine girer.

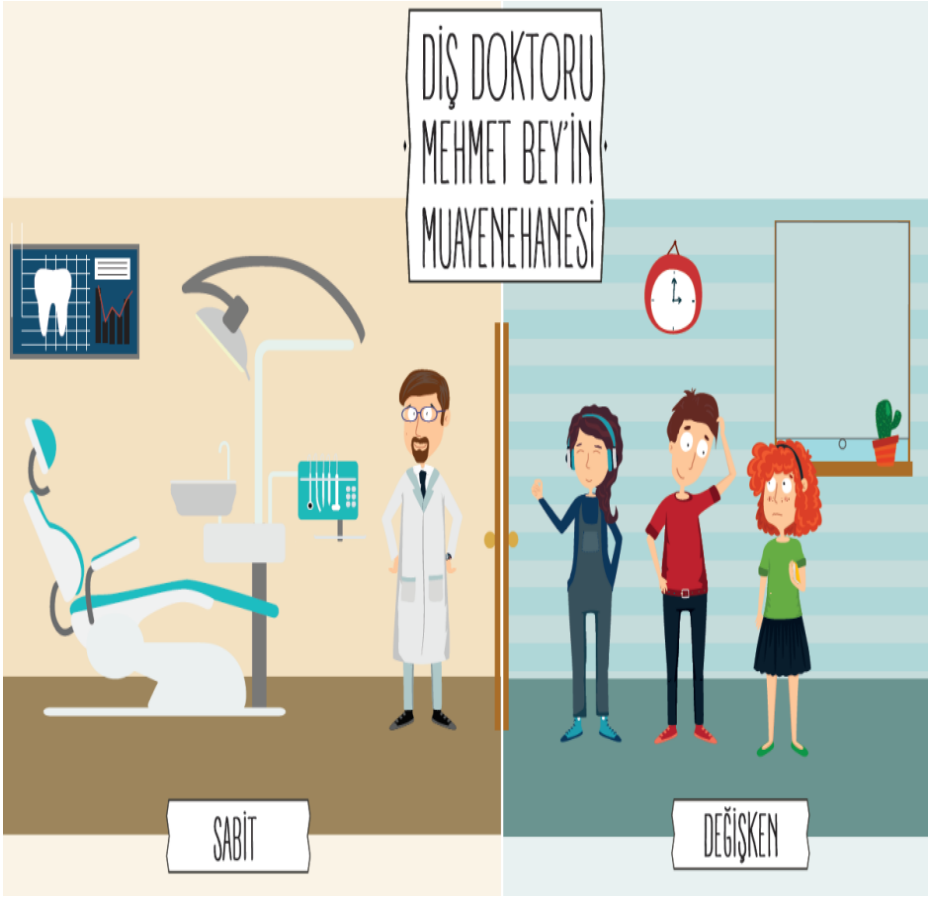
Yukarıdaki örnekte kullanıcıdan saati tahmin etmesi isteniyor. Ardından kullanıcının girdiği saat ekrana yazdırılıyor ve devamında gerçek saat ve dakika ekranda görünüyor.

Örnek:

Sayısal Veri Tipi ✔	Yüz ölçümü nedir? 783. 562 km²
Sayısal Veri Tipi ✔	Nüfusu nedir? 79.814.871
Sayısal Veri Tipi ✔	Nüfus yoğunluğu nasıl hesaplanır? Nüfus/Yüzölçümü
Karakter Dizisi V... ✔	En çok toprağı hangi kıtada yer alıyor? Anadolu
Özel Veri Tipi ✔	Türkiye Cumhuriyeti Devleti hangi tarihte kuruldu? 29 Ekim 1923
Karakter Dizisi V... ✔	Başkenti neresidir? Ankara
Mantıksal Veri Tipi ✔	Birleşmiş Milletler Uluslararası Çocuklara Yardım Fonu üyesi mi? Evet
Karakter Veri Tipi ✔	Ülkemizin en batı ucundaki yerleşim biriminin ilk harfi nedir? G (Gökçeada)

Değişkenler

Örnek 1: Resimde görüldüğü gibi bir diş doktorunun muayenehanesinde Diş Doktoru Mehmet Bey akşama kadar 15 hasta ile ilgilenmiştir. Buradaki dişçi koltuğu **sabit**, gelip giden hastalar ise **değişken**dir.



Bilgisayarların işleyişinde de bazı veriler değişkenler aracılığıyla depolanırken bazı veriler ise sabit olarak kalır. Sabit olarak kalan bu birimler ne olursa olsun değişmez ve ilk depolandıkları hâliyle kullanılmaya devam ederler. Değişkenler ise duruma göre değişirler ve sabit kalmazlar.

Örnek 2:

Sizce bu fotoğrafta yer alan sabit ve değişkenler nelerdir?

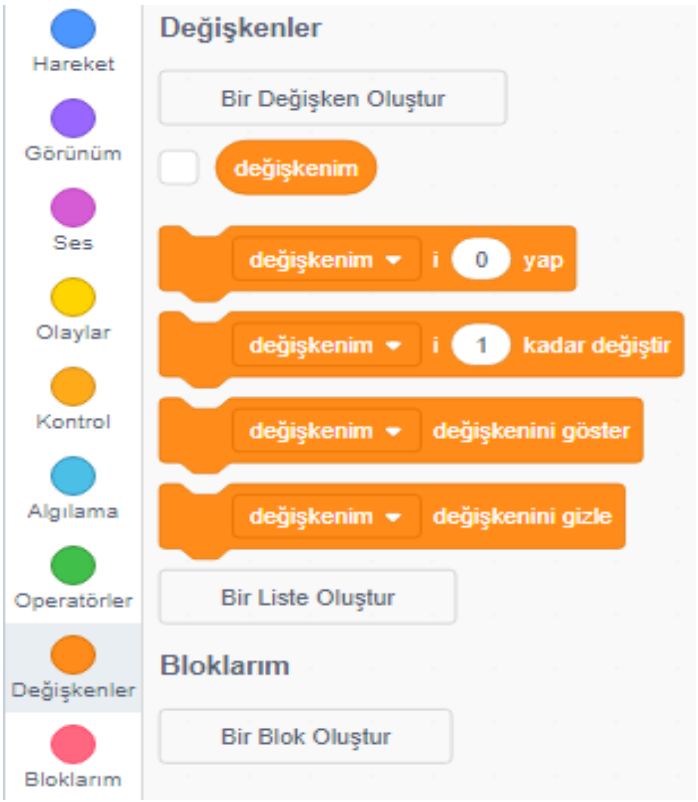
Örnek 3:

Peki bir çocuk parkında yer alan sabit ve değişkenler nelerdir?

Parkta yer alan oyuncaklar sabittir.

Parkı ziyarete ya da oynamaya gelen çocuklar,gençler ve insanlar da değişkenlerdir.

Scratch Blok Tabanlı Kodlamada Değişkenler



Scratch blok tabanlı kodlama da Değişkenler bloğundan Bir değişken oluştur tıklanarak değişkene isim verilir ve ok tıklanarak değişken oluşturulur.

Değişkeni … yap komutu ile değişkene değer aktarılır.

Değişkeni … kadar değiştir komutu ile değişken belirlenen oranda artırılıp azaltılabilir.

Değişkeni göster ve Değişkeni gizle komutlarıyla program içerisinde değişkenin değerinin gösterilmesi ya da gizlenenilmesi sağlanır.

Örnek Program Algoritması: Adım 1: Adın ne? Sorularak kullanıcıdan adının girmesi sağlanır. Adım 2: Kullanıcının girdiği ad isim diye bir değişkene kaydedilir. Adım 3: Merhaba isim diye ekrana çıktı verir. Burada isim değişkendir yani kullanıcı ne ad girerse o ekranda çıkar.

Örnek Program Algoritması Adım 1: Saati saat değişkene aktarılacak. Adım 2: Dakikayı soracak ve dakika da bir değişkene aktarılacak. Adım 3: Merhaba şu an gün boyunca kaç dakika geçtiğini ekranda gösterecek.

Saat ve dakika sürekli değişen değerler olduğu için değişkene aktarılarak işlem yapılacak. Burada sabit olan bir saatin 60 dakika olmasıdır.

Karar Yapıları

Günlük hayatımızda pek çok karar veririz ve bu kararlara göre yapacağımız şeyler, sonuçlar değişir. Örneğin; hava güneşli ise pikniğe gitme, değilse evde film izleme **kararına** varabiliriz. Eğer mutluysak hareketli şarkılar, üzgün veya yorgun hissediyorsak daha sakin şarkılar dinlemeye karar verebiliriz. Ailemiz bize çok istediğimiz bisikleti almak için derslerde başarılı olma şartı koyabilir.

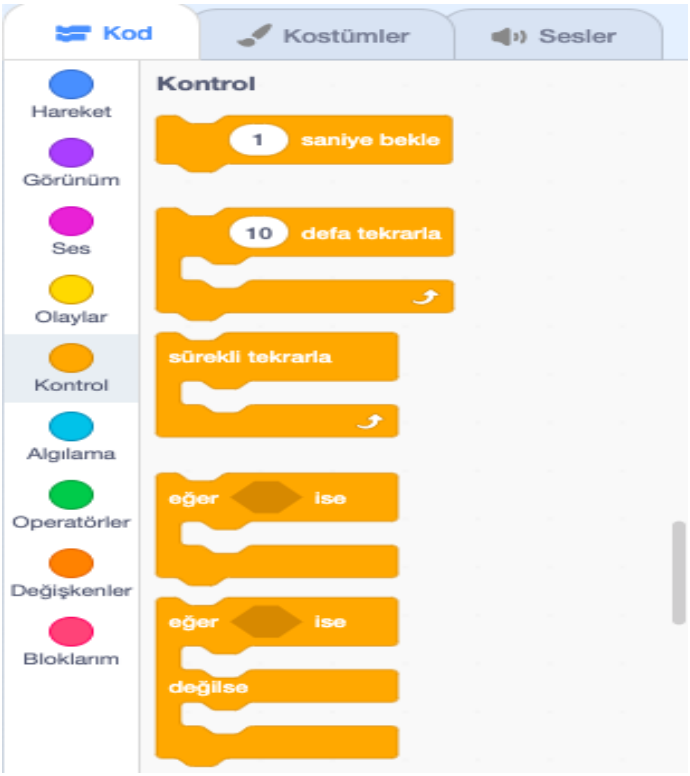
Programlamada da bazı işlem adımlarının istenen durumlarda veya şartlarda çalışması istendiğinde **Karar Yapıları** kullanılır.

Doğrusal mantık yapısında kodlar hiçbir şarta bağlı kalmaksızın, ard arda ve sıralı bir şekilde çalışırken, karar yapısında bir şart veya duruma bağlı olarak sonuç değişir.

Algoritmada Karar Yapısı Örneği

Doğrusal Mantık Yapısı	**Karar Yapısı**
1. Başla 2. “Doğum yılınızı giriniz:”, dy 3. Yas=2021-dy 4. Yas değerini ekrana yazdır. 5. Bitir.	1. Başla 2. “Doğum yılınızı giriniz:”, dy 3. Yas=2021-dy 4. Eğer Yas>=18 ise; Ekrana “Ehliyet Alabilirsiniz”. yaz. Adım 6 ‘ ya git. 5. Değilse Ekrana “Ehliyet Alamazsınız.” yaz. 6. Bitir.

Scratch Blok Tabanlı Kodlama Uygulamasında Karar Yapıları



Scratch uygulamasında Kontrol kategorisinde Eğer-ise ve Eğer-ise-değilse kod blokları karar yapıları için kullanılan bloklardır. Operatörler kategorisindeki Büyüktür-Küçüktür-Eşittir vb operatörler ile karşılaştırma ve karar verme süreçleri için kullanılabilir. Aşağıdaki tabloda kullanım örneklerini görebilirsiniz.

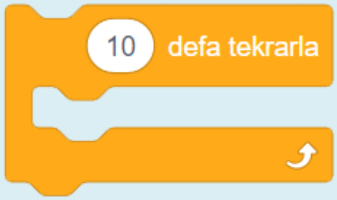
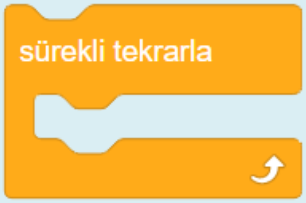
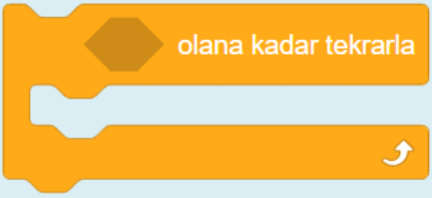
Doğrusal Mantık Yapısı

Karar Yapısı

Döngüler

Döngüler belirli işlemleri belirli bir sayıda yapan ya da bu işlemleri belirli koşullara bağlı olarak gerçekleştiren kod bloklarıdır. İkinci ve daha basit bir tanım yapacak olursak döngüler, tekrarlanan işlemleri azaltan kod bloklarıdır da diyebiliriz. Eğer döngüler belirli bir sayıda ya da sürekli tekrar ediyorsa **sabit döngü**, döngü sayısı belli değil ve belirli bir şarta bağlı olarak gerçekleşecekse buna **koşullu döngü** diyebiliriz.

Döngü Tipleri

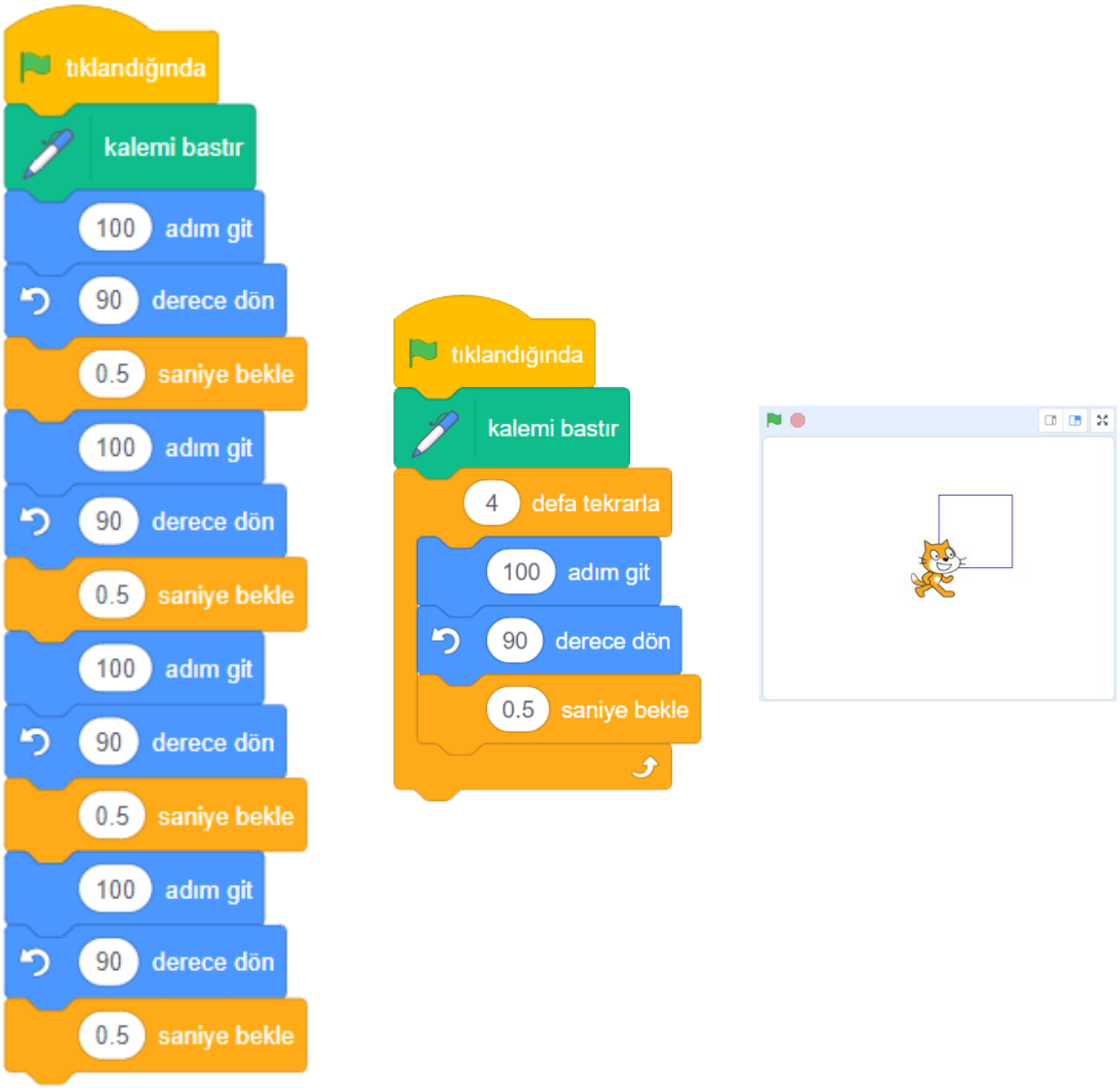
Sabit Döngüler	Koşullu Döngü
• **Belirli sayıda tekrarlayan döngü**  • **Sürekli döngü** 	• **Olana kadar tekrarla** 

Belirli Sayıda Tekrarlayan Döngü

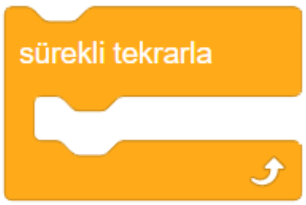
En çok kullanılan döngü komutlarından biridir. Eğer yapacağınız işlemdeki döngü sayısı sabit ve belirli bir sayıda ise bu döngü komutu tercih edilir. Yanda da görüldüğü gibi bu döngüde döngünün kaç kez çalıştırılacağını belirleyen bir alan vardır. Görünen kod bloğundaki ağza yerleştireceğiniz kod veya kod bloklarını belirlenen tekrar sayısında çalıştırabilirsiniz. Ayrıca tekrar sayısı bölümüne yerleştireceğiniz değişkenler ile de tekrar sayısını kontrol edebilirsiniz.

Bu komuta günlük hayattan bazı örnekler vererek pekiştirelim. Örneğin evinizden okula gelene kadar yaptığınız eylem yani yürüme eylemi adım atmanın tekrarıdır. Okula ulaşana kadar

atmış olduğunuz bu adımlar aslında tekrar eden bir döngüdür. Beden eğitimi dersinde öğretmeninizin ısınmanız için yaptırdığı hareketlerin tekrarı birer döngüdür. Şimdi de kodlamadan örnekler verelim. Ekrana bir kare çizdiğinizi düşünün. Dört kenar için ayrı ayrı kodlama yapmak yerine bir kenar için kod yazarak bunu dört kez tekrar ettirebilirsiniz. Örnek kodlamayı aşağıdan inceleyelim.



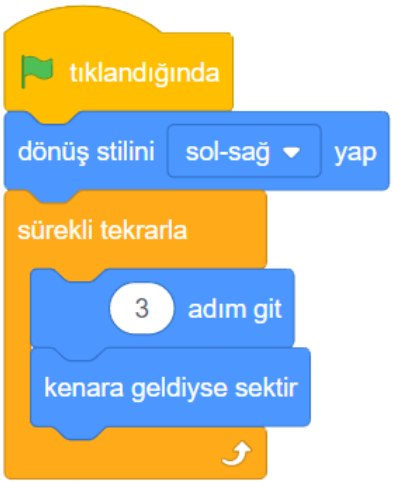
Sürekli Döngü

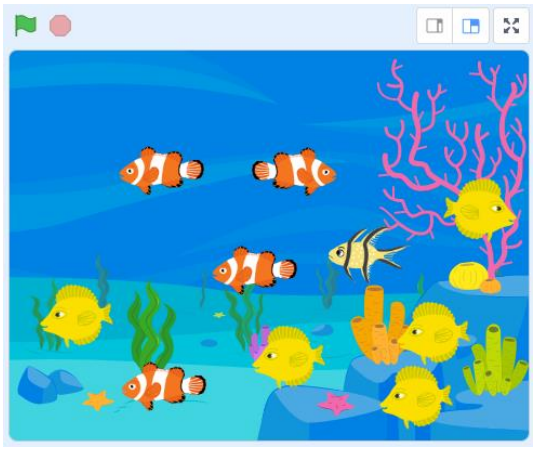


Bir diğer çok kullanılan döngü komutlarından birisi de sürekli tekrarla komutudur. Bu blok içerisine koyulan komutlar sürekli olarak çalışır. Bu sebepledir ki bu komutun altına kod eklenemez. Bunu anlamamız için de altındaki çıkıntıya bakmamız yeterlidir. Tüm kodlarda

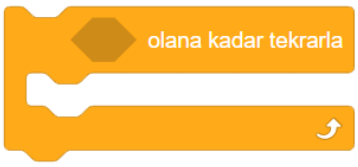
olan çıkıntı sürekli tekrarla bloğunda yoktur. Sürekli tekrarlaya sonsuz döngü de diyebiliriz.

Bu komuta günlük hayattan örnek vermek gerekirse balıkları örnek ele alabiliriz. Balıkların akvaryum içerisindeki sürekli ve rastgele hareketleri buna örnek olabilir. Örnek bir akvaryum kodlamasını inceleyelim. Balıklarımız sürekli hareket edecek ve kenara geldiklerinde yönünü değiştirip hareketlerine devam edecekler.



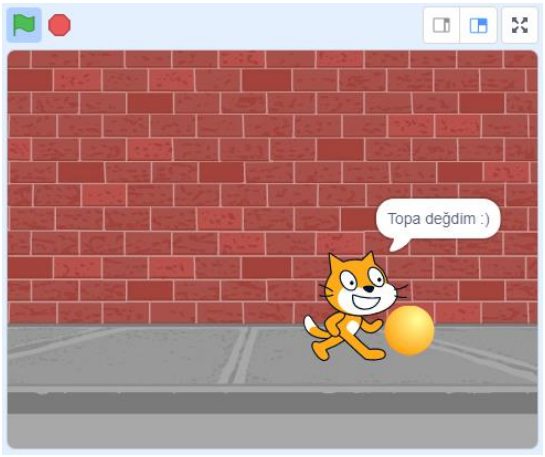


Koşullu Döngü



Önceki döngülerde herhangi bir koşul ya da şart aranmaksızın belirli sayıda ya da sürekli döngüyle kodlar çalıştırılırken bu kod bloğu ise koşula bağlı olarak gerçekleşen bir döngüdür. Bu döngü de öncelikle koşul kontrol edilir. Belirlenen koşul sağlanana kadar da döngü çalışmaya devam eder. Koşul sağlandığı zaman döngü sonlandırılır ve görevini altındaki komutlara devrederek programı çalışması devam ettirilir.

Bu komuta günlük hayattan oldukça fazla örnek verebiliriz. Öğretmenlerin verdiği ödevlerin bitene kadar yapılması, yemeğin bitene kadar yenmesi, uyanana kadar veya saat çalana kadar uyumak gibi birçok örnek verilebilir. Dikkat edilmesi gereken nokta hepsinde bir koşul bulunmasıdır. Şimdi buna kodlamadan örnek verelim. Örnek kodlamada kuklamız diğer kuklaya ulaşana kadar hareket edecek ve daha sonra döngü sonlanıp görevini alttaki komuta devredecek.

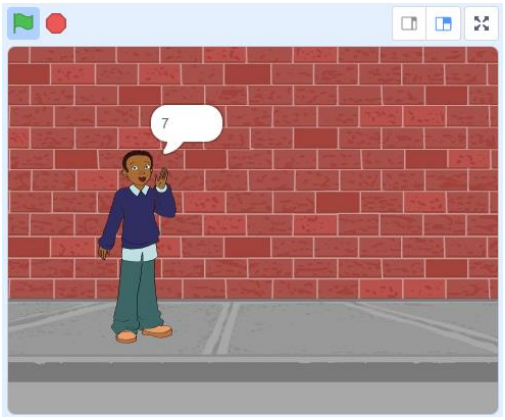


Bir örnek üzerinde üç farklı döngü kullanımını da inceleyelim. Bu örnekte kuklamız 1'den 10'a kadar sayıları sayacak.

Belirli Sayıda Tekrarlayan Döngü

Sürekli Döngü

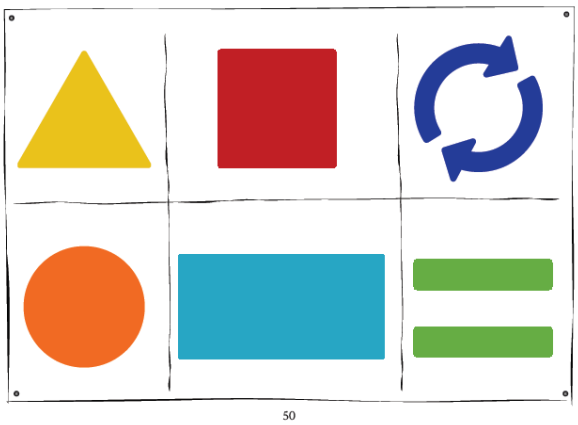
Koşullu Döngü



Döngü seçimindeki dikkat edilmesi gereken nokta algoritmanın temelindeki kurallarda yatar. Sizi sonuca en hızlı, en basit ve en kısa yoldan götürecek döngü seçilmelidir.

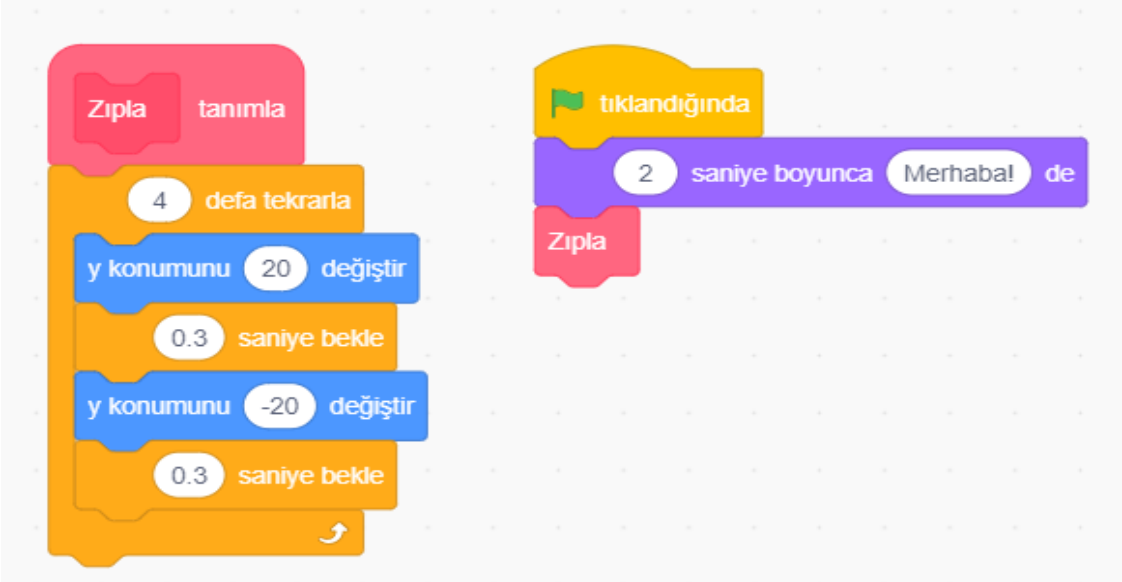
Fonksiyonlar

Fonksiyon, belirli bir iş veya işleri yapması için tanımlanmış komut grubudur.



Örneğin yandaki çizim makinemizden üçgeni çizmek için Ahmet, kareyi çizmek için Ayşe, daireyi çizmek için Kerem arkadaşlarımız görevlendirilir. Böylece biz üçgen çizmek istediğimizde Ahmet'i, kareyi çizmek istersek Ayşe'yi, daireyi çizmek istediğimizde Kerem'i çağırırız. Şekilleri çizmeye uğraşmayız. Biz sadece kişinin adını çağırırız. Kişinin adını çağırmamız fonksiyona benzetilebilir.

Scratch'te fonksiyona aşağıdaki kod blokları örnek olarak verilebilir: Kod bölümünde Bloklarım kategorisinden Bir Blok Oluştura tıklanır. Blok adına Zıpla yazılır. Bloğun altına zıplama için gerekli kod blokları eklenir. Artık Zıpla bir fonksiyondur. Bundan sonra zıpla fonksiyonunu nerede çağırırsak kodları tekrar yazmadan kuklamızın zıplamasını sağlarız.



Kaynakça

1- 6.Sınıf Bilişim Teknolojileri ve Yazılım Dersi Öğretmen Rehberi, Millî Eğitim Bakanlığı Yayınları, 2018.

2- 5.Sınıf Bilişim Teknolojileri ve Yazılım Dersi Öğretmen Rehberi, Millî Eğitim Bakanlığı Yayınları, 2018.

3- Fatih Erdinç ve Zeynep Erdinç, Uygulamalarla Programlama Öğreniyorum, Abaküs Yayınları 2015, 163-178.

4- Kırkkonaklar İffet Güneşoğlu Ortaokulu URL: https://www.kigobilisim.com/kopyasi-veri-tuerlerim (Erişim zamanı: 04.02.2021)