# KHAL 2025-2026 EĞİTİM VE ÖĞRETİM YILI

# BİLİŞİM TEKNOLOJİLERİ VE YAZILIM DERSİ

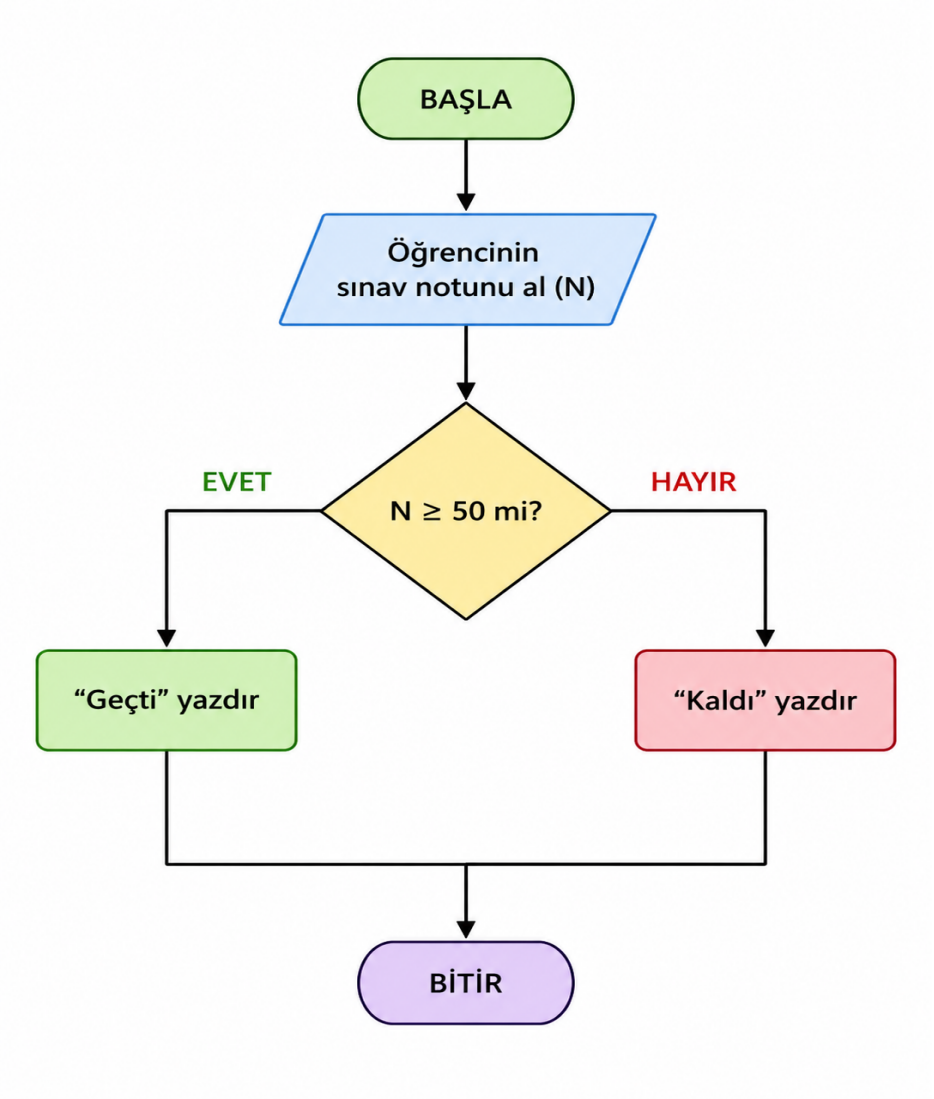
# 2. DÖNEM 2. YAZILI SINAVA HAZIRLIK DERS NOTLARI

## AKIŞ ŞEMALARINDA KULLANILAN TEMEL SEMBOLLER

| Şekil | Adı | Görevi |
|---|---|---|
| Oval | Başla / Bitir | Programın başlangıç ve bitişini gösterir. |
| Dikdörtgen | İşlem | Hesaplama veya işlem yapılacağını gösterir. |
| Paralelkenar | Giriş / Çıkış | Veri alma veya ekrana yazdırma işlemlerini gösterir. |
| Baklava | Karar | Koşul kontrolünü gösterir. |
| Ok | Akış çizgisi | İşlem sırasını gösterir. |

**Örnek**: Aşağıdaki problemin akış diyagramını hazırlayınız.

**Problem**: Bir öğrencinin başarı notu 50 ve yukarısıysa ekrana “Geçti”; 50 ve altındaysa “Kaldı” yazan algoritma




## Programlamada Kullanılan Temel İşlem ve Semboller

| Sembol | Adı | Görevi / Anlamı |
|---|---|---|
| + | Toplama operatörü | Sayıları toplamak için kullanılır. |
| - | Çıkarma operatörü | Sayıları çıkarmak için kullanılır. |
| * | Çarpma operatörü | Sayıları çarpmak için kullanılır. |

| Sembol | Adı | Görevi / Anlamı |
|---|---|---|
| / | Bölme operatörü | Sayıları bölmek için kullanılır. |
| % | Mod operatörü | Bölme işleminden kalanı verir. |
| = | Atama operatörü | Bir değeri değişkene aktarmak için kullanılır. |
| == | Eşittir karşılaştırması | İki değerin eşit olup olmadığını kontrol eder. |
| != | Eşit değildir | İki değerin farklı olup olmadığını kontrol eder. |
| > | Büyüktür | Bir değerin diğerinden büyük olup olmadığını kontrol eder. |
| < | Küçüktür | Bir değerin diğerinden küçük olup olmadığını kontrol eder. |
| >= | Büyük eşittir | Bir değerin diğerinden büyük veya eşit olduğunu kontrol eder. |
| <= | Küçük eşittir | Bir değerin diğerinden küçük veya eşit olduğunu kontrol eder. |
| && veya and | Mantıksal VE | İki koşulun da doğru olmasını ister. |
| ` | | veyaor` |
| ! veya not | Mantıksal DEĞİL | Sonucu tersine çevirir. |
| () | Parantez | İşlem önceliği vermek için kullanılır. |
| { } | Süslü parantez | Kod bloklarını belirtmek için kullanılır. |
| [ ] | Köşeli parantez | Dizi ve liste işlemlerinde kullanılır. |
| " " | Çift tırnak | Metin ifadelerini belirtmek için kullanılır. |

| Sembol | Adı | Görevi / Anlamı |
|---|---|---|
| # | Yorum satırı | Kod açıklaması yazmak için kullanılır. |
| : | İki nokta üst üste | Python'da blok başlangıcını belirtir. |

## VERİ TÜRLERİ

Temel veri türlerini kullanım amaçlarına göre 4 ana grupta inceleyebiliriz:

### 1. Sayısal Veri Türleri

Matematiksel hesaplamalarda, yaş belirtirken, fiyat belirlerken veya bir döngü sayacında kullanılırlar.

- **int (integer - tam Sayılar):** Kesir kısmı olmayan, pozitif veya negatif tüm tam sayılardır.
  - *Kullanım Amacı:* Adet, yaş, yıl, öğrenci sayısı gibi bölünemeyen değerler için.
  - *Örnek:* ogrenci_sayisi = 24, hava_sicakligi = -5

- **float (Floating Point - Ondalıklı Sayılar):** Kesirli, noktalı sayılardır. Python'da ondalık ayırıcı olarak virgül , değil, nokta . kullanılır.
  - *Kullanım Amacı:* Fiyatlar, boy/kilo ölçümleri, pi sayısı gibi hassas matematiksel değerler için.
  - *Örnek:* urun_fiyati = 129.99, boy = 1.78

### 2. Metinsel Veri Türü

Ekranda görünecek mesajlar, kullanıcı adları, adresler veya herhangi bir metin bu türde tutulur.

- **str (String - Metin dizileri):** Tek tırnak '...' veya çift tırnak "..." içinde yazılan her şey string'dir. Tırnak içine yazılan sayılar da matematiksel özelliğini kaybeder ve birer yazıya dönüşür.

- *Kullanım Amacı:* İsimler, mesajlar, e-posta adresleri veya üzerinde matematiksel işlem yapılmayacak sayısal görünümlü veriler (telefon numarası, TC kimlik no gibi) için.
- *Örnek:* isim = "Ahmet", telefon = "05551234567" *(Telefon numarasını int yaparsak baştaki sıfır kaybolur ve toplama çıkarma yapmayacağımız için str olması doğrudur).*

## 3. Mantıksal Veri Türü

Programda bir durumun doğru mu yoksa yanlış mı olduğunu kontrol etmek, "evet/hayır" cevabı almak için kullanılır.

- **bool (Boolean - Mantıksal Değerler):** Sadece iki değer alabilir:

  True (Doğru) veya False (Yanlış).

  Baş harflerinin büyük olması zorunludur.

  - *Kullanım Amacı:* Giriş yapıldı mı?, Kullanıcı reşit mi?, Dosya bulundu mu? gibi iki ihtimalli kontrol durumlarında.
  - *Örnek:*
    sistem_acik_mi = True
    odeme_yapti_mi = False

## 4. Koleksiyon / Veri Yapısı Türleri

Tek bir değişken içinde birden fazla veri saklamak ve bunları organize etmek istediğimizde kullanılır.

- **list (Listeler):** Köşeli parantez [...] ile tanımlanır. İçindeki elemanlar değiştirilebilir (mutable), sıralıdır ve aynı elemandan birden fazla barındırabilir.

- *Kullanım Amacı:* Alışveriş listesi, bir sınıftaki öğrencilerin isimleri, sınav notları gibi üzerinde ekleme, çıkarma ve güncelleme yapacağımız dinamik gruplar için.
- *Örnek:*
  notlar = [85, 90, 78, 100],
  programlama_dilleri = ["Python", "Java", "C#"]

- **dict (Dictionary - Sözlükler):** Süslü parantez {...} ile tanımlanır. Her eleman bir **Anahtar (Key)** ve bir **Değer (Value)** çiftinden oluşur (key: value).
  - *Kullanım Amacı:* Bir nesneye ait farklı özellikleri bir arada tutmak için. Tıpkı gerçek bir sözlükte kelime ve anlamı gibi, bir anahtarla o anahtara ait bilgiye hızlıca ulaşmak istenir.

*Örnek:*

```
ogrenci_bilgi = {
    "isim": "Onur",
    "numara": 403,
    "not_ortalamasi": 88.5
}
```

## KARAR YAPILARI (if - elif - else)

Hayatımızda sürekli kararlar veririz: *"Eğer hava yağmurluysa şemsiye al, değilse alma."* Bilgisayarların da akıllıca davranması ve duruma göre farklı kodları çalıştırması için **Karar Yapılarını** kullanırız.

Python'da karar mekanizmaları **karşılaştırma operatörleri** ve if-elif-else blokları ile kurulur.

### 1. Karşılaştırma Operatörleri (Karar Verme Araçları)

Bilgisayarın bir karara varabilmesi için önce iki durumu karşılaştırması gerekir. Bu işlemlerin sonucu her zaman True (Doğru) ya da False (Yanlış) döner.

- == Eşit mi? (x == 5)
- != Eşit değil mi? (x != 10)
- < Küçük mü? / > Büyük mü?
- <= Küçük eşit mi? / >= Büyük eşit mi?

### 2. Karar Yapısı Blokları

**Sadece Tek Bir Şart Varsa: if (Eğer)**

Belirtilen şart doğruysa (True) altındaki kod bloğu çalışır. Şart yanlışsa hiçbir şey yapılmaz, kod aşağıya doğru akmaya devam eder.

```
# Örnek: Yaş kontrolü
yas = 20

if yas >= 18:
    print("Ehliyet alabilirsiniz.")
```

⚠ **Önemli Kural (Girinti - Indentation):** Python'da if satırının sonuna mutlaka iki nokta üst üste : konur. Bir sonraki satır ise klavyedeki **TAB** tuşu kadar (veya 4 boşluk) içeriden başlar. İçeride kalan kısım, sadece o if doğruysa çalışacak olan bölgedir.

### İki İhtimal Varsa: if - else (eğer - değilse)

Şart doğruysa if bloğu, şart yanlışsa else bloğu çalışır. Ortası yoktur, bu iki bloktan biri **mutlaka** çalışacaktır.

```
# Örnek: Sınavı geçme/kalma durumu
notunuz = 45
if notunuz >= 50:
    print("Tebrikler, dersi geçtiniz!")
else:
    print("Maalesef, dersten kaldınız.")
```

### Birden Fazla Şart Varsa: if - elif - else

"Eğer ilk şart uymadıysa, **şu şart uymuş mu diye bak**" demek için elif (else if) kullanılır. İstediğimiz kadar elif ekleyebiliriz.

```
# Örnek: Trafik ışığı simülasyonu
isik = "Sarı"
if isik == "Kırmızı":
    print("Dur!")
elif isik == "Sarı":
    print("Hazırlan!")
elif isik == "Yeşil":
    print("Geç!")
else:
    print("Hatalı ışık rengi!")
```

## ÖRNEK UYGULAMALAR

### Örnek 1: Kullanıcı Giriş Kontrolü (Gelişmiş if-else)

```
kullanici_adi = input("Kullanıcı adınızı girin: ")
sifre = input("Şifrenizi girin: ")

if kullanici_adi == "admin" and sifre == "1234":
    print("Sisteme başarıyla giriş yapıldı. Hoş geldiniz!")
else:
    print("Kullanıcı adı veya şifre hatalı!")
```

### Örnek 2: Boy/Kilo Endeksi (Çoklu Karar)

```
# Girilen sıcaklığa göre hava durumunu yorumlayan program
sicaklik = int(input("Oda sıcaklığını giriniz (°C): "))

if sicaklik < 15:
    print("Hava soğuk, kombiyi çalıştırın.")
elif sicaklik >= 15 and sicaklik <= 25:
    print("Hava ideal, ortam sıcaklığı güzel.")
else:
    print("Hava sıcak, klimayı açabilirsiniz.")
```

### Sık Yapılan Hatalar

1. **Eşittir Karmaşası:** Şart yazarken tek eşittir = kullanmak hata verir. Tek eşittir *değer atamak* içindir (x = 5). Karşılaştırma yaparken çift eşittir == kullanılmalıdır.
2. **Unutulan İki Nokta:** if, elif ve else kelimelerinin sonuna : koymamak Python'da yazım hatasına (SyntaxError) yol açar.
3. **Hizalama (Syntax/Indentation Error):** if'in içine yazılacak kodların içeride (girintili) olmaması programın hangi kodun nereye ait olduğunu anlamasını engeller.

## DÖNGÜLER (LOOP):

Birden fazla kod dizisinin tekrar tekrar çalışmasını sağlayan yapılardır. İki farklı döngü çeşidi vardır. Bunlar **for** ve **while** döngüsüdür.

**for döngüsü:** Verileri veya nesneleri tek tek işleme sokar.

**Örnek Kod:**

```
#1'den 5' kadar olan sayıları yazdırmak isteyelim.
print(1)
print(2)
print(3)
print(4)
print(5)
```

***Çıktı:***

*1*
*2*
*3*
*4*
*5*

*Amacıma ulaştım ancak ekrana 1'den 100'e kadar yazdırmam gerekseydi bu şekilde bir çözüm yolu saatlerce kod yazmamıza neden olacaktı. İşte böylesi zor durumlarda döngü yapıları kullanmak imdadımıza yetişir.

**Örnek Kod:**

```
#1'den 100'e kadar olan tüm sayıları yazdıralım. Değişkenimiz i olsun;
for i in range(1,101):
    print(i)
```

***Çıktı:***

*1*

*2*

*3*

.

.

.

*99*
*100*

**Ek bilgi:** Buradaki i değişkeni semboliktir. İstenilen isimlendirme yapılabilir ancak genel alışkanlık olarak i harfi kullanılır. Örneğin n harfi de sıklıkla kullanılır. Bu değişken döngüye girecek olan tüm veri veya nesneleri temsil eder.

**Ek bilgi:** Döngülerde ve koşullu yapılarda iki altın kural vardır. Döngünün başladığı satırın sonuna iki nokta üst üste konulur. Satır aşağısında yazılan kodlar ise biraz içeriden yazılır.

**range fonksiyonu:** Türkçe olarak "aralık" anlamına gelmektedir. Python'da sayı aralıklarını belirtmek için kullanırız. Temel olarak çalışma mantığı şu şekildedir:

range (başlangıç değeri, bitiş değeri, atlama sayısı)

Birkaç tane örnek verelim:

**range(4,10,2):** Burada başlangıç değeri 4, bitiş değeri 10, artış değeri ise 2'dir. 4'ten başlayarak bitiş değerine kadar 2'şer sıra atlanır ancak bitiş değeri olan 10 bu gruba dâhil değildir. Çünkü range fonksiyonunda bitiş değeri aralığa dahil edilmez.

***Sonuç:*** *4-6-8*

**range(1,10):** Burada başlangıç değeri 1, bitiş değeri 10'dur. Ancak artış değeri belirtilmemiştir. Burada 1 ile 10 aralığındaki sayılar kastedilir. Ancak buna 10 dâhil değildir çünkü range fonksiyonunda bitiş değeri belirtilmişse işleme alınmaz.

***Sonuç:*** *1-2-3-4-5-6-7-8-9*

**range(10):** Burada başlangıç değeri ve atlama sayısı belirtilmemiştir, bitiş değeri ise 10'dur. Burada 0 ile 10 aralığındaki sayılar kastedilir. Çünkü range fonksiyonunda başlangıç değeri yoksa sıralama 0'dan başlar. Bitiş değeri belirtildiği için yine bu gruba 10 dâhil edilmez.

***Sonuç:*** *0-1-2-3-4-5-6-7-8-9*

**range(10,2,-2):** Burada başlangıç değeri 10, bitiş değeri 2, atlama sayısı ise -2'dir. Bunun anlamı başlangıç değerinden itibaren 2'şer 2'şer azalma olmasıdır.

***Sonuç:*** *10-8-6-4*

**Örnek Kod:**

```
#Şimdi de 1'den 100'e kadar tüm tek sayıları yazdıralım. Değişkenimiz bu sefer n olsun;
for n in range(1,101,2):
    print(n)
```

***Çıktı:***

***1***

***3***

***5***

.

.

.

***97***

***99***

***Açıklama:*** *range(1, 101, 2) ifadesindeki 1 başlangıç sayısıdır. Eğer burası boş bırakılırsa sayı otomatik olarak sıfırdan başlar. 101 ise bu durumda bitiş değer olur. 101 sayısı bitiş değeri olduğu için çıktıya dâhil değildir. 2 ise artış miktarını gösterir. Yani sayıyı 2'şer arttırır.*

* Geriye doğru sıralama işlemi de yapılabilir.

**Örnek Kod:**

```
#21'den 0'a geriye doğru 3'er sıralama.
for n in range(21,0,-3):
    print(n)
```

***Çıktı:***

***21***

***18***

***15***

.

.

.

***6***

***3***

* Belirli aralıktaki sayılar toplanabilir.

**Örnek Kod:**

```
#1'den 100'e kadar olan sayıların toplamı
top= 0
for i in range(1,101):
    top+= i
print(top)
```

***Çıktı: 5050***

**Örnek Kod:**

```
sayılar = "1234"
for sayı in sayılar:
    print(int(sayı) * 2)
```

***Çıktı:***

***2***

***4***

***6***

***8***

**Örnek Kod:**

```
#Döngü ve koşullu yapıların beraber kullanılması
sayilar = "1234567"
for i in sayilar:
    if int(i) > 3:
        print(i)
```

***Çıktı:***

***4***

***5***

***6***

***7***

***Açıklama:*** *sayilar değişkeni oluşturduk. "1234567" ifadesindeki her bir karakteri ayırdık. Yani tüm sayılar artık bağımsızlığını ilan etmiş durumda. Ayrıca her bir karakteri i değişkene atadık. Ancak biz bu karakterleri int(i) kodunu yazarak tamsayıya çevirdik çünkü* matematiksel işlemler yapmak için verileri sayıya çevirmemiz gerekiyor. if komutuyla da 3'ten büyük olan sayıları yazdırdık.

**Örnek Kod:**

```
#Döngü ve liste yapılarının birlikte kullanılması
a=[]
for i in range(2,10,2):
    a+=[i]
print(a)
```

***Çıktı: [2, 4, 6, 8]***

**Örnek Kod:**

```
#Döngü ve liste yapılarının birlikte kullanılması
isim=["Onur", "Duru", "Teoman"]
for i in isim:
    print(i[0])
```

***Çıktı:***

***O***

***D***

***T***

**while döngüsü:** Bir koşul sağlanmaya devam ettiği sürece işlemleri tekrarlar. İngilizce bir kelime olan while, Türkçede '-iken, olduğu sürece' anlamına gelir.

**Örnek kullanım:**

while a == 1: Anlamı: a, 1'e eşit olduğu olduğu sürece şu işi yap…

**sonsuz döngü (infinite loop):**

```
#a, 5'dan küçük olduğu sürece ekrana "Duru" yazdır.
a = 1
while a < 5
        print("Duru")
```

***Çıktı:***

***Duru***

***Duru***

***……. Sonsuza kadar Duru yazmaya devam eder.***

***Açıklaması:*** *Burada a değişken değeri 5'den küçük olduğu sürece ekrana "Duru" yazdıracak ancak sorun şu ki a=1 olduğu için a her zaman 5'den küçük olacak. Bu da ekrana sürekli "Duru" yazılmasına neden olacak. Buna sonsuz döngü diyoruz. Buna son vermek için klavyenizde Ctrl+C veya Ctrl+Z tuşlarına basarak programı durmaya zorlayabilirsiniz.*

---

Sonsuz döngüyü kırmak için koşulları değiştirelim:

**Örnek Kod:**

```
#1'den 4'e kadar olan sayıları yazdıralım (4 dahil değil)
a=1
while a<4:                  #a 4'ten küçük olduğu sürece
    print(a)                #a'yı yazdır.
    a+=1                    #a'yı 1 arttır.
```

***Çıktı:***

***1***

***2***

***3***

***Açıklaması:*** *İlk satırda a değişkene 1 değerini atadık. İkinci satıra geldiğimizde değişkenin 5'ten küçük olup olmadığına baktık, eğer küçükse kodumuz alt satıra geçecek ve böylece değişken ekrana yazdırılacak. Son koda geldiğimizde a değişkeni 1 değer artıp 2 olacak ve döngüye girecek. Sonra döngü devam edip ekrana 2 yazdırılacak. Bu durum a'nın 5'ten küçük olmaması şartına kadar sağlanacak. Yani a artık 5 olduğunda döngü duracak.*

**Örnek Kod:**

```
#Döngünün kaç defa döneceğini kullanıcın girdiği sayı ile belirleyelim.
n = 1
karar= int(input("Sayılar kaça kadar sıralansın? " ))
while n <= karar:
        print(n)
        n += 1
```

***Sayılar kaça kadar sıralansın? 50***

***Çıktı:***

***1***

***2***

***…..50***

**Açıklama:** Bu programda döngünün kaç defa döneceğine kullanıcı karar verecektir. Örneğin kullanıcı 7 sayısını girerse 1'den 7'ye kadar olan sayılar ekrana yazdırılacaktır. Ancak kullanıcının girdiği sayı 1 veya daha az olursa program tepki vermeyecektir.

**while True:** Bu kodu kullandığımızda döngü aksi belirtilmediği müddetçe sonsuza kadar çalışır.

**Örnek Kod:**

```
#ekrana sonsuza kadar bir değer yazdırma
while True:
    print("Bilgisayar çıldırdı!")
```

***Çıktı:** Bilgisayar çıldırdı! …. (sonsuza kadar ekrana bu şekilde yazar)*

**Döngüye son vermek (break komutu)**

**Örnek Kod:**

```
#break komutu ile döngüyü durdurabiliriz.
while True:
    print("Bilgisayar çıldırdı!")
    break
```

***Çıktı:** Bilgisayar çıldırdı! (sadece bir defa ekrana bu şekilde yazar)*

**Örnek Kod:**

```
#Aşağıdaki toplama işlemi sürekli tekrar edilecektir.
while True:
    a=int(input("sayı gir: "))
    b=int(input("sayı gir: "))
    print("Sayıların toplamı:", a+b)
```

**Örnek Kod:**

```
#Belirli bir koşul sağlanana kadar döngüyü sürekli çalıştıralım.
#Koşul sağlanırsa döngüyü sonlandıralım.
while True:
    cevap = input("Nasılsın, iyi misiniz? :  ")
    if cevap == "Tamam":
        print("Tamam kızma sormuyorum artık! ")
        break
```

**Örnek Kod:**

```
#Kullanıcıya bir parola oluştururken belirli şartlar sunalım.
while True:
    parola = input("Bir parola belirleyin: ")
    if not parola:                          #eğer parola değeri yoksa yani oluşturulmamışsa
        print("Parola bölümü boş geçilemez!")
    elif len(parola) < 6:                   #eğer parola karakter sayısı 6'dan küçükse
        print("Parola 6 karakterden kısa olmamalı")
    else:
        print("Yeni parolanız: ", parola)
        break
```

## Döngüyü es geçip başa döndürmek (continue komutu)

Türkçe olarak "devam" anlamına gelen bu ifade döngüde o an için işlemi gerçekleştirmez ve tekrar döngünün başına dönmeyi sağlar.

**Örnek Kod:**

```
#Sınıf listesinden Metehan adlı kişiyi çıkartalım.
sinif=["Onur", "Serkan", "Metehan", "Ezgi"]
for i in sinif:
    if i=="Metehan":    #eğer döngüde "Metehan" varsa onu atla ve diğerlerine devam et
        continue
    print(i)
```

***Çıktı:***

***Onur***

***Serkan***

***Ezgi***

**Örnek Kod:**

```
#Belirli bir koşulu sağlayan döngü verisini es geçerek döngünün başına dönelim.
a=[1,2,3,4,5]
for i in a:
    if i==3:
        continue
    print(i)
```

***Çıktı:***

***1***

***2***

***4***

***5***

**pass komutu:** Herhangi bir işlem yapmadan geçeceğimiz durumlarda kullanılır. Kısaca "Hiçbir şey yapmadan yola devam et!" anlamı taşır. O an için karar veremediğimiz ama daha sonra karar verebileceğimiz işlemler de pass komutu bir yer tutucu olarak kullanılabilir.

**Örnek Kod:**

```
#sayı tahmin oyunu
sayi=9
while True:
    tahmin = int(input("1'den 10'a kadar bir sayı tahmin et: "))
    if tahmin == sayi:
        print("Tebrikler doğru tahmin!")
        break
    elif tahmin > sayi:
        pass
    else:
        pass
```

**Açıklaması:** Buradaki pass ifadesi elif ve else **boş bırakılmaması** için kullanılmıştır.

Kodun ilgili kısmı şu mantıkta çalışır:

Python'da if, else, for, while, function gibi yapıların içi tamamen boş bırakılamaz. Eğer boş bırakılırsa hata oluşur.

Bu nedenle geçici olarak hiçbir işlem yapılmayacağını belirtmek için pass kullanılır.

pass kelimesinin anlamı:

pass → "geç", "hiçbir şey yapma"

Yani burada programın anlamı şudur:

"Tahmin yanlışsa hiçbir işlem yapmadan döngüye devam et."

Bu kodda zaten while True: bulunduğu için kullanıcı tekrar tahmin girebilir.

Buraya daha sonradan farklı bir işlem eklenebilir. Örneğin:

```
elif tahmin > sayi:
    print("Tahmini azalt")

else:
    print("Tahmini artır")
```

Bu durumda pass kullanmaya gerek kalmazdı.

## ALGORİTMA VE AKIŞ DİYAGRAMLARINDA SIK YAPILAN HATALAR

### 1. Başla veya Bitir Adımının Bulunmaması

Her algoritma ve akış diyagramı bir başlangıç ve bitiş noktasına sahip olmalıdır.

#### Hatalı Örnek

1. Sayıyı gir.
2. Sayının karesini al.
3. Sonucu yazdır.

***Hata: Başla ve Bitir adımları yoktur.***

### 2. Adımların Mantıklı Sırada Olmaması

İşlemler doğru sırayla yapılmalıdır.

**Hatalı Örnek**

1. Sonucu yazdır.
2. Sayıyı gir.
3. Sayının karesini hesapla.

***Hata: Sonuç hesaplanmadan yazdırılmaya çalışılmıştır.***

---

### 3. Eksik Adım Bulunması

**Çözüm için gerekli bir işlem unutulabilir.**

**Hatalı Örnek**

1. Başla
2. Sayıyı gir.
3. Sonucu yazdır.
4. Bitir

***Hata: Sayı üzerinde yapılacak işlem belirtilmemiştir.***

---

### 4. Karar Yapısının Yanlış Kullanılması

**Bir koşul varsa "Evet" ve "Hayır" yolları açıkça belirtilmelidir.**

**Örnek**

"Not 50 ve üzeriyse geçti, değilse kaldı."

***Hata: Karar kutusundan yalnızca bir çıkış verilmesi.***

### 5. Sonsuz Döngü Oluşması

**Döngünün sona ermesini sağlayacak koşul unutulabilir.**

#### Hatalı Durum

Başla
Sayıyı Yazdır
Tekrar Sayıyı Yazdır
Tekrar Sayıyı Yazdır
...

***Hata: Programın ne zaman duracağı belirtilmemiştir.***

---

### 6. Yanlış Sembol Kullanılması

**Akış diyagramında her işlem uygun sembolle gösterilmelidir.**

#### Hatalı Örnek

Veri girişi için dikdörtgen kullanılması.

Karar işlemi için paralelkenar kullanılması.

***Hata: Sembol görevleri karıştırılmıştır.***

---

### 7. Ok Yönlerinin Yanlış Çizilmesi

**Akışın hangi yönde ilerlediği açık olmalıdır.,**

#### Hata Örneği

Okların birbirini kesmesi

Nereye gittiği belli olmayan oklar

Geri dönüşlerin gösterilmemesi

## 8. Değişkenlerin Tanımlanmaması

**Algoritmada kullanılan verilerin ne olduğu belirtilmelidir.**

### Hatalı Örnek

1. Başla
2. Toplam = A + B
3. Sonucu yazdır
4. Bitir

***Hata: A ve B değerlerinin nereden geldiği belirtilmemiştir.***

---

## Bir Algoritmayı Kontrol Ederken Sorulacak Sorular

✓ Başla ve Bitir adımları var mı?

✓ Adımlar doğru sırada mı?

✓ Eksik işlem var mı?

✓ Karar yapıları doğru kurulmuş mu?

✓ Döngüler sonlanıyor mu?

✓ Kullanılan semboller doğru mu?

✓ Ok yönleri doğru mu?

✓ Tüm değişkenler tanımlanmış mı?

✓ Algoritma istenen sonucu üretiyor mu?

## AKIŞ DİYAGRAMINDAKİ HATALARIN ETKİSİ

Akış diyagramı algoritmanın görsel gösterimidir. Akış diyagramındaki bir hata, programın yanlış çalışmasına neden olabilir.

---

### Örnek 1: Eksik İşlem

***Problem:*** **İki sayının toplamını bulmak.**

***Hatalı Akış :*** ***Başla → Sayıları Gir → Sonucu Yazdır → Bitir***

***Hata:*** ***Toplama işlemi unutulmuştur.***

**Sonuç:** **Program toplamı hesaplayamaz.**

---

### Örnek 2: Yanlış Karar Yapısı

***Problem:*** ***Notu 50 ve üzeri olan öğrenciler geçsin.***

Hatalı Durum: Karar kutusundan yalnızca "Evet" çıkışı verilmiş.

Sonuç: Notu 50'nin altında olan öğrenciler için ne yapılacağı belirtilmemiştir.

Program eksik çalışacaktır.

---

### Örnek 3: Sonsuz Döngü

**Problem: 1'den 10'a kadar sayıları yazdırmak.**

Hatalı Algoritma

1. Başla
2. Sayıyı yazdır.
3. Sayıyı artırma.
4. Adım 2'ye dön.
5. Bitir

***Hata: Sayı artırılmadığı için döngü hiç sona ermez.***

***Sonuç:*** ***Program sürekli aynı işlemi tekrar eder.***

---

**Hatalı Algoritmaların Oluşturabileceği Sorunlar**

- Yanlış sonuç üretilmesi
- Hiç sonuç üretilememesi
- Programın donması
- Sonsuz döngü oluşması
- Bazı durumların göz ardı edilmesi
- Kullanıcının yanlış bilgi alması
- Zaman ve emek kaybı yaşanması

---

**Algoritma ve Akış Diyagramı Testi**
Bir algoritma veya akış diyagramı hazırlandıktan sonra mutlaka test edilmelidir.

**Test Yaparken**
✓ Farklı giriş değerleri kullanılmalıdır.

✓ Tüm olasılıklar kontrol edilmelidir.

✓ Karar yapıları denenmelidir.

✓ Döngülerin sonlandığından emin olunmalıdır.

✓ Beklenen çıktı ile elde edilen çıktı karşılaştırılmalıdır.

---

**Uygulama Etkinliği**

Aşağıdaki algoritmayı inceleyiniz.

1. Başla
2. Bir sayı gir.
3. Sonucu ekrana yazdır.
4. Bitir

***Sorular***

1. Algoritmadaki hata nedir?
2. Program hangi sonucu verecektir?
3. Hatanın düzeltilmiş hâli nasıl olmalıdır?

***Cevap***

**Hata:** Sayı üzerinde yapılacak işlem belirtilmemiştir.

**Sonuç:** Program istenilen görevi yerine getiremez.

**Düzeltme:** Gerekli işlem adımı eklenmelidir.

## ALGORİTMA VE AKIŞ DİYAGRAMININ HATALARINI DÜZELTME

### Algoritma ve Akış Diyagramlarında Hata Düzeltmenin Önemi

Bir algoritma veya akış diyagramı hazırlanırken bazen eksik, yanlış veya mantık hataları içeren adımlar oluşturulabilir. Bu hatalar düzeltilmediğinde program istenilen sonucu vermez.

Bu nedenle algoritma ve akış diyagramları incelenmeli, hatalar belirlenmeli ve uygun şekilde düzeltilmelidir.

---

### Hata Düzeltme Nedir?

Hata düzeltme, algoritma veya akış diyagramındaki yanlış, eksik veya gereksiz adımların düzeltilerek çözümün doğru hâle getirilmesidir.

#### Amaç

- Doğru sonuç elde etmek
- Programın hatasız çalışmasını sağlamak
- Eksik adımları tamamlamak
- Mantık hatalarını gidermek

---

### Algoritmalarda Karşılaşılan Hatalar ve Düzeltmeleri

#### *1. Adımların Yanlış Sırada Olması*

**Hatalı Algoritma**

1. Başla
2. Sonucu yazdır.
3. Sayıyı gir.
4. Sayının karesini hesapla.
5. Bitir

**Hata:** Sonuç hesaplanmadan önce ekrana yazdırılmaya çalışılmıştır.

**Düzeltilmiş Hâli**

1. Başla
2. Sayıyı gir.
3. Sayının karesini hesapla.

4. Sonucu yazdır.
5. Bitir

---

**2. Eksik İşlem Bulunması**
**Hatalı Algoritma**

1. Başla
2. Bir sayı gir.
3. Sonucu yazdır.
4. Bitir

**Hata:** Girilen sayı üzerinde yapılacak işlem belirtilmemiştir.

**Düzeltilmiş Hâli**

1. Başla
2. Bir sayı gir.
3. Sayının karesini hesapla.
4. Sonucu yazdır.
5. Bitir

---

**3. Eksik Karar Yapısı**

**Problem:** Notu 50 ve üzeri olan öğrenciler geçsin.
**Hatalı Durum**

- Not ≥ 50 → Geçti

**Hata:** Notu 50'nin altında olan öğrenciler için işlem belirtilmemiştir.

**Düzeltilmiş Hâli**

- Not ≥ 50 → Geçti
- Not < 50 → Kaldı

---

**4. Sonsuz Döngü**

**Hatalı Algoritma**

1. Başla
2. Sayıyı yazdır.
3. İkinci adıma dön.
4. Bitir

**Hata:** Döngünün sona ermesini sağlayacak koşul yoktur.

**Düzeltilmiş Hâli**

1. Başla
2. Sayı = 1
3. Sayıyı yazdır.
4. Sayıyı 1 artır.
5. Sayı ≤ 10 ise 3. adıma dön.
6. Bitir

---

**Akış Diyagramlarında Hata Düzeltme**

Akış diyagramındaki hatalar da uygun semboller ve bağlantılar kullanılarak düzeltilmelidir.

---

**Yanlış Sembol Kullanımı**

**Hatalı Durum:** Veri girişinin dikdörtgen sembolü ile gösterilmesi.

**Düzeltme:** Veri girişleri paralelkenar sembolü ile gösterilmelidir.

| İşlem Türü | Doğru Sembol |
|---|---|
| Başla / Bitir | Oval |
| Veri Girişi | Paralelkenar |
| İşlem | Dikdörtgen |
| Karar | Eşkenar Dörtgen |

---

**Eksik Ok Kullanımı**

**Hatalı Durum:** Karar kutusundan çıkan okların belirtilmemesi.

**Düzeltme:** Karar kutusundan çıkan tüm yollar açıkça gösterilmelidir.

Örneğin:

- Evet
- Hayır

çıkışları mutlaka bulunmalıdır.

**Hata Düzeltirken İzlenecek Adımlar**

***1. Problemi Tekrar İnceleme***

Önce çözülmek istenen problem net olarak anlaşılmalıdır.

***2. Adımları Kontrol Etme***

Her adımın gerekli olup olmadığı incelenmelidir.

***3. Mantık Sırasını Kontrol Etme***

İşlemlerin doğru sırada yapıldığından emin olunmalıdır.

***4. Karar Yapılarını İnceleme***

Tüm olasılıkların dikkate alındığı kontrol edilmelidir.

***5. Döngüleri Test Etme***

Döngülerin sonlanıp sonlanmadığı kontrol edilmelidir.

***6. Sonucu Deneme***

Farklı örnek veriler kullanılarak algoritma test edilmelidir.

---

**Uygulama Çalışması**

Aşağıdaki algoritmadaki hataları bulunuz ve düzeltiniz.

**Hatalı Algoritma**

1. Başla
2. Sonucu yazdır.
3. Bir sayı gir.
4. Sayının iki katını hesapla.
5. Bitir

**Çözüm**

**Hata 1:** Sonuç hesaplanmadan yazdırılmaktadır.

**Hata 2:** Hesaplanan sonucun yazdırılması unutulmuştur.

**Düzeltilmiş Algoritma**

1. Başla
2. Bir sayı gir.
3. Sayının iki katını hesapla.

4. Sonucu yazdır.
5. Bitir