Yöntem Dil Başvurusu

ÖNDER ÖRNEK

Yöntem Dil Başvurusu

BİRİNCİ BASKI
2026

Yöntem Dil Başvurusu
Önder Örnek

Telif hakkı © 2026 Önder Örnek.

Bu eser Creative Commons Atıf 4.0 Uluslararası Lisansı (CC BY 4.0) ile lisanslanmıştır. Yazarın adı belirtilerek, lisansa bağlantı verilerek ve yapılan değişiklikler gösterilerek ticari amaçlar dahil her ortamda ve biçimde kopyalanabilir, dağıtılabilir ve uyarlanabilir. Lisans metni:
`https://creativecommons.org/licenses/by/4.0/deed.tr`

Eserdeki örnek programlar MIT lisansıyla kullanılabilir. Yöntem yazılımı da MIT lisanslıdır.

Birinci baskı: Kasım 2026

Dizgi: Metin TeX Gyre Pagella, program kodları DejaVu Sans Mono yazı tipiyle, XeLaTeX kullanılarak dizilmiştir.

İçindekiler

Bu başvuru ne anlatır? 1
- Uygulamalar 1
- Gösterim 2

1. Sözcük yapısı 3
- Harf duyarsızlığı 3
- Yorumlar 3
- Adlar 4
- Anahtar sözcükler 4
- Sayılar 4
- Metinler 5
- Ekler 5
- İşleçler ve ayırıcılar 5
- Satır sonları 6

2. Program yapısı 7
- Bildirim bölümleri 8
- Kapsam 8

3. Tipler ve değerler 9
- Bileşik tip adları 10
- Tip denetimi 10
- Değer ve referans 10
- Liste 10
- Boyutu sabit liste 11
- Sözlük 11
- Metin 12

4. İfadeler 13
Aritmetik 14
Karşılaştırma 14
Mantık 14
Boru 15
Birincil ifadeler 15
Adsız alt programlar 16

5. Deyimler 17
Atama 17
Çağrı deyimi 17
Blok içi değişken 17
Koşul 18
Çok seçenekli karar 18
Sayan döngü 18
Koleksiyon döngüsü 19
Koşullu döngüler 19
Döngü denetimi 20
Eleman silme cümlesi 20
Dönüş 20

6. Alt programlar 21
Sonuç 21
Parametreler 21
Direktifler 22
Özyineleme ve iç içe alt programlar 22

7. Sınıflar ve kayıtlar 23
Metodlar 24
Nesne oluşturma 24
Kalıtım 24
Görünürlük 25
Özel adlı metodlar 25
Kayıtlar 25

8. Hatalar 27

9. Birimler 29

10. Giriş ve çıkış 31

11. Katı kip 33

12. Uygulama farkları 35

A Dilbilgisinin tamamı 37

B Yerleşik işlevler 41

Bu başvuru ne anlatır?

Bu belge Yöntem dilinin sözdizimini ve anlamını tanımlar: bir programın hangi sözcüklerden oluştuğunu, bu sözcüklerin hangi kurallarla bir araya geldiğini ve yazılan her yapının çalışırken ne yaptığını. Dili öğrenmek için yazılmamıştır; öğrenmek isterseniz *Yöntem ile Programlamaya Giriş* kitabından başlayabilirsiniz. Bu başvuruya, bir kuralın tam olarak ne dediğini görmek istediğinizde bakabilirsiniz.

Belge Yöntem 0.6.0 sürümünü anlatır. Dilbilgisinin tamamı depodaki `yontem.gram` dosyasındadır; buradaki kurallar o dosyadan alınmıştır.

Uygulamalar

Yöntem'in üç uygulaması vardır ve bu başvuru üçü için de geçerlidir:

Uygulama	Ne yapar
JavaScript derleyicisi	Programı JavaScript'e çevirip Node.js'te ya da tarayıcıda çalıştırır
Python yorumlayıcısı	Programı doğrudan yorumlar; etkileşimli kabuk buradadır
Gömülü arka uç	Programın statik tipli bir alt kümesini C++17'ye çevirir (ESP32)

İlk ikisi aynı programa aynı çıktıyı verir. Gömülü arka uç dilin tamamını değil, tipleri derleme sırasında belirlenebilen bölümünü kabul eder; farklar "Uygulama farkları" bölümündedir.

Gösterim

Sözdizimi kuralları şu gösterimle yazılmıştır:

- `kural: a b` — `kural`, art arda gelen `a` ve `b`'den oluşur.
- `a | b` — `a` ya da `b`; ilk uyan seçenek geçerlidir.
- `[a]` — `a` isteğe bağlıdır.
- `a*`, `a+` — `a` sıfır ya da daha çok kez, bir ya da daha çok kez.
- `',' .a+` — virgülle ayrılmış bir ya da daha çok `a`.
- `'başla'` — sözcüğün kendisi (anahtar sözcük ya da işleç).
- `AD`, `SAYI`, `METIN` — sözcük türleri.

Örneklerde program çıktısı, kodun hemen ardındaki ayrı blokta gösterilir.

1. Sözcük yapısı

Bir Yöntem programı UTF-8 kodlu bir metin dosyasıdır; uzantısı `.y`'dir. Derleyici dosyayı önce sözcüklere ayırır.

Harf duyarsızlığı

Adlar ve anahtar sözcükler karşılaştırılırken büyük/küçük harf ayrımı yapılmaz ve Türkçe harfler ASCII karşılıklarına katlanır: `ç→c`, `ğ→g`, `ı→i`, `ö→o`, `ş→s`, `ü→u`. Bu yüzden `Başla`, `BAŞLA` ve `basla` aynı sözcüktür; Türkçe klavyesi olmayan biri de program yazabilir.

Bunun bir sonucu vardır: `sayı` ile `sayi`, `ış` ile `is`, `x` ile `X` aynı addır. Aynı kapsamda böyle iki ad bildirilirse derleyici hata verir. Hata iletilerinde ve çıktıda adlar sizin yazdığınız biçimiyle görünür.

Yorumlar

```
{ süslü ayraç içinde }
(* yıldızlı ayraç içinde *)
// satır sonuna kadar
```

Yorumlar birden çok satıra yayılabilir, iç içe yazılamaz. Dosyada `{$katı}` yorumu bulunuyorsa derleyici o dosyayı katı kipte denetler (bkz. "Katı kip").

Adlar

```
AD: (harf | '_') (harf | rakam | '_')*
```

Harf, Unicode'daki herhangi bir harftir. Anahtar sözcükler ad olarak kullanılamaz.

Anahtar sözcükler

Aşağıdaki sözcükler ayrılmıştır:

```
program    birim     kullan    arayüz    uygulama   değişken
sabit      tip       prosedür  fonksiyon kurucu     başla
bitir      eğer      ise       değilse   iken       için
kadar      her       içinde    içindeki  tekrarla   dene
yakala     sonunda   fırlat    dön       çık        kır
devam      sınıf     kayıt     miras     kendi      doğru
yanlış     boş       ve        veya      yada       değil
mod        böl       dizi      azalarak  türünde
```

`azalarak` ve `türünde` eski sürümlerin yazımından kalmıştır; yalnızca yeni yazımı gösteren hata iletisi verebilmek için ayrılmıştır.

Bazı sözcükler yalnızca belirli bir yerde özel anlam taşır, başka yerlerde sıradan ad olarak kullanılabilir:

Sözcük	Özel anlam taşıdığı yer
`adımda`, `artan`, `azalan`	sayan döngü başlığı
`kez`	`n kez tekrarla`
`olana dek`	`tekrarla` döngüsünün sonu
`değerine göre`	çok seçenekli karar
`listesindeki`, `dizisindeki`, `elemanı`	eleman silme cümlesi
`listesi`, `dizisi`, `sözlüğü`	tip ekleri
`sanal`, `ezer`, `soyut`, `ileri`, `aşırı`, `statik`	alt program direktifleri
`özel`, `genel`, `korumalı`, `yayınlanmış`	sınıf içinde görünürlük bölümleri

Sayılar

Biçim	Örnek	Tip
Onluk tamsayı	`42`, `1_000_000`	`tamsayı`
Onaltılık tamsayı	`$FF`	`tamsayı`
Ondalık	`3.14`, `1e-3`, `2.5E+4`	`ondalık`

Alt çizgi yalnızca okunaklılık içindir. Noktadan sonra en az bir rakam gelmelidir; `2.` bir ondalık sayı değildir.

Metinler

Biçim	Kaçış kuralı
`'tek tırnak'`	İçindeki tırnak iki kez yazılır: `'Ali''nin'`
`"çift tırnak"`	`\n`, `\t`, `\r`, `\\`, `\"`, `\'`, `\0`
`#10`	Onluk karakter kodu

Yan yana yazılan parçalar tek metin olur: `'a'#10'b'` üç karakterlik bir metindir. Bir metin satır sonunu aşamaz.

Ekler

Türkçede olduğu gibi sayılara, adlara ve ayraçlı ifadelere kesme işaretiyle ek getirilebilir. Dil iki eki tanır ve yalnızca sayan döngüde kullanır:

Ek	Yazımları
Ayrılma	`'den`, `'dan`, `'ten`, `'tan`, `'nden`, `'ndan`
Yönelme	`'e`, `'a`, `'ye`, `'ya`, `'ne`, `'na`

Kesme işareti bir sayıya, ada, `)` ya da `]` işaretine bitişikse ve ardından bu eklerden biri geliyorsa ek sayılır; diğer her durumda metin başlatır. Ses uyumu denetlenmez: `1'dan` da kabul edilir.

İşleçler ve ayırıcılar

```
:=  +=  -=  *=  /=  **  <>  <=  >=  ..  |>  =>
+   -   *   /   @   =   <   >
(   )   [   ]   ,   ;   :   .
```

Satır sonları

Deyimler ve bildirimler satır sonuyla ayrılır; `;` isteğe bağlıdır ve aynı satıra birden çok deyim yazmak için kullanılır: `a := 1; b := 2`. Satır sonu şu yerlerde de anlam taşır:

- Bir çağrının `(` ve bir dizinin `[` işareti, uygulandığı ifadeyle aynı satırda başlamalıdır.
- `listesi`, `dizisi`, `sözlüğü` ekleri tiple aynı satırda olmalıdır.
- `dön` deyiminin değeri `dön` ile aynı satırda başlamalıdır.

Bir ifade, ikili bir işleçten sonra alt satırda sürebilir.

2. Program yapısı

```
dosya:   birim | program
program: ['program' AD] kullan* bildirim* blok '.'
kullan:  'kullan' ','.AD+
blok:    'başla' deyimler 'bitir'
```

Bir dosya ya bir program ya da bir birimdir. Program başlığı isteğe bağlıdır; ardından kullanılan birimler, bildirimler ve `başla … bitir.` ile yazılan ana gövde gelir.

```
program Selam
kullan Matematik

sabit
  KEZ = 3
değişken
  i: tamsayı

prosedür Selamla(kime: metin = 'dünya')
başla
  yaz('Merhaba, ' + kime, 1)
bitir

başla
  1'den KEZ'e kadar her i için Selamla
  Selamla('Yöntem')
bitir.
```

Bildirim bölümleri

```
bildirim: 'sabit' (AD [':' tip] '=' ifade)*
        | 'değişken' (','.AD+ [':' tip] ['=' ifade])*
        | 'tip' (AD '=' tip_tanimi)*
        | altprogram
```

Bölümler istenen sırada ve istenen sayıda yazılabilir. Değişken bildiriminde tip ya da ilk değerden en az biri bulunmalıdır; `=` yerine `:=` de yazılabilir.

Bildirim	Anlamı
`sayaç: tamsayı`	Tipli değişken; tipin başlangıç değerini alır
`ad: metin = 'Yöntem'`	Tipli ve ilk değerli
`oran := 0.5`	Tipsiz; her tipte değer alabilir
`a, b: ondalık`	Aynı tipte birden çok değişken

Başlangıç değerleri: `tamsayı` için `0`, `ondalık` için `0.0`, `metin` için boş metin, `mantıksal` için `yanlış`, `liste` için `[]`, `sözlük` için `[:]`, kayıt tipleri için alanları başlangıç değerinde bir kayıt, sınıf tipleri ve `herhangi` için `boş`.

Sabite atama yapılamaz; denenirse program "Sabite atama yapılamaz" hatasıyla durur.

Kapsam

Bir ad, bildirildiği yerden itibaren o kapsamda ve içteki kapsamlarda görünür. Kapsamlar şunlardır: program (ya da birim), her alt program, her sınıf (alanlar ve metodlar, metod gövdelerinde doğrudan görünür). Bir blok içinde `değişken` ile bildirilen ad, içinde bulunduğu alt programın (ya da ana gövdenin) kapsamına girer.

İçteki bir bildirim dıştaki aynı adı gölgeler. Birimlerden gelen adlar da program içinde yeniden bildirilerek gölgelenebilir.

3. Tipler ve değerler

```
tip:        tip_tabani tip_eki*
tip_tabani: 'dizi' | [AD '.'] AD
tip_eki:    'listesi' | 'dizisi' | 'sözlüğü'
```

Tip	Değerler	Not
`tamsayı`	`42`, `-7`, `10 ** 30`	Masaüstünde büyüklük sınırı yoktur
`ondalık` (`gerçek`)	`3.14`	Çift duyarlıklı kayan nokta
`metin`	`'abc'`	Karakterler 1'den numaralanır
`karakter`	`'a'`	Tek karakterlik metin
`mantıksal`	`doğru`, `yanlış`	
`liste`	`[1, 2, 3]`	Elemanlar 0'dan numaralanır
`sözlük`	`['a': 1]`, `[:]`	Ekleme sırasını korur
`tensör`	`Tensör([[1, 2]])`	`YZ` ve `Meşale` birimleri
`herhangi`	her değer	Tip denetimi yapılmaz
sınıf ve kayıt adları	nesneler	

`boş`, "değer yok" anlamına gelen tek değerdir.

Bileşik tip adları

Eleman tipi başa, kap sona yazılır; ekler soldan sağa birikir:

Yazım	Anlamı
`tamsayı listesi`	tamsayılardan oluşan liste
`metin sözlüğü`	değerleri metin olan sözlük
`ondalık listesi listesi`	ondalık listelerinin listesi
`tamsayı dizisi`	`tamsayı listesi` ile aynıdır

`dizisi` ile `listesi` eş anlamlıdır. Köşeli ayraçla sınır yazımı (`dizisi[1..12]`) dilden kaldırılmıştır; sabit boyut programda verilir (bkz. “Boyutu sabit liste”).

Tip denetimi

Tipi bildirilmiş bir değişkene, parametreye ya da alana uymayan bir değer atandığında program çalışırken hata verir:

```
değişken n: tamsayı
n := 'metin'
```

```
Çalışma hatası: 'n' için tip uyuşmazlığı: tamsayı bekleniyordu, metin verildi
```

Tek örtük dönüşüm `tamsayı → ondalık` yönündedir: `ondalık` tipli bir değişkene `3` atanırsa değer `3.0` olur. Katı kipte aynı denetimler program çalışmadan önce yapılır ve eleman tipleri de denetlenir.

Tip adları aynı zamanda dönüştürücüdür: `tamsayı('42')`, `ondalık('3,5')`, `metin(12)`, `liste(x)`. `Tür(x)` değerin tip adını metin olarak verir.

Değer ve referans

Sayılar, metinler ve mantıksal değerler değiştirilemez. Kayıtlar değer olarak davranır: atandıklarında ve parametre olarak verildiklerinde kopyalanır. Listeler, sözlükler, tensörler ve sınıf nesneleri referansla paylaşılır: `b := a` yazdıktan sonra `b` üzerinden yapılan değişiklik `a`’da da görünür. Bağımsız bir kopya için `Kopya(a)` ya da `liste(a)` kullanılabilir.

Liste

Liste sıralı ve boyutu değişebilen bir elemanlar topluluğudur.

İşlem	Yazım
Okuma, değiştirme	`L[0]`, `L[0] := 5`
Sondan sayma	`L[-1]` son elemandır
Eleman sayısı	`Uzunluk(L)`, `L.Uzunluk`
Sona ekleme	`Ekle(L, x)`, `L.Ekle(x)`
Konuma ekleme	`Ekle(L, x, konum)`, `L.Ekle(x, konum)`
Silme	`Sil(L, 2)`, `L.Sil(2)`, `L(2).Sil`, `L` listesindeki 2. elemanı Sil
Birleştirme	`L + M`
Üyelik	`x içinde L`

Silmenin dört yazımı aynı anlama gelir ve 2 numaralı elemanı (`L[2]`) siler; `Sil` çıkardığı elemanı döndürür. Sınır dışındaki bir numaraya erişmek hata verir.

Boyutu sabit liste

```
değişken aylar: metin listesi
başla
  aylar.Uzunluk(12, 1)
  aylar[1] := 'Ocak'
```

`L.Uzunluk(n)` listeyi `n` elemanlı yapar ve boyutunu sabitler; numaralar 0'dan başlar. `L.Uzunluk(n, ilk)` numaralamayı `ilk`'ten başlatır. Yeni elemanlar eleman tipinin başlangıç değeriyle dolar (eleman tipi bilinmiyorsa `boş`); listede önceden eleman varsa baştan itibaren korunur.

Boyutu sabit listede:

- geçerli numaralar `L.Alt` ile `L.Üst` arasındadır; dışına çıkmak hata verir, sondan sayma yoktur;
- `Ekle` ve `Sil` hata verir;
- okuyan liste metodları (`İçerir`, `Sırala`, `Toplam` gibi) aynen çalışır;
- boyut yalnızca yeni bir `Uzunluk` çağrısıyla değişir; `liste(L)` serbest bir kopya verir.

`Uzunluk` çağrısı değişkene yeni bir liste atar; aynı listeyi gösteren başka değişkenler eski listeyi göstermeyi sürdürür.

Sözlük

Sözlük anahtarları değerlere eşler. Sabit yazımı `['a': 1, 'b': 2]`, boş sözlük `[:]` biçimindedir. `S['a']` okur, `S['a'] := 3` yazar ya da ekler; olmayan bir anahtarı

okumak hata verir. `x içinde S` anahtar üyeliğini sınar, `Sil(S, anahtar)` anahtarı kaldırır. Anahtarlar ekleme sırasını korur.

Metin

Metin karakterleri 1'den numaralanır: `s[1]` ilk karakterdir. `Uzunluk(s)` karakter sayısını verir. `+` iki metni birleştirir; işlenenlerden biri metinse diğeri metne çevrilir: `'x = ' + 5`.

4. İfadeler

İşleçler öncelik sırasına göre, en az bağlayandan en çok bağlayana doğru:

Düzey	İşleçler	Birleşme
1	boru işleci (bkz. ``Boru'')	soldan
2	`veya`, `yada`	soldan
3	`ve`	soldan
4	`değil`	tekli
5	`=`, `<>`, `<`, `>`, `< =`, `> =`, `içinde`	zincirlenmez
6	`+`, `-`	soldan
7	`*`, `/`, `böl`, `mod`, `@`	soldan
8	tekli `-`, `+`	tekli
9	`**`	sağdan
10	çağrı `f(x)`, dizin `L[i]`, üye `n.ad`	soldan

```
ifade:   veya ('|>' veya)*
veya:    ve (('veya' | 'yada') ve)*
ve:      degil ('ve' degil)*
degil:   'değil' degil | iliski
iliski:  toplama [('=' | '<>' | '<' | '>' | '< =' | '> =' | 'içinde') toplama]
toplama: carpma (('+' | '-') carpma)*
carpma:  tekli (('*' | '/' | '@' | 'böl' | 'mod') tekli)*
tekli:   ('-' | '+') tekli | '@' sonek | sonek ['**' tekli]
sonek:   birincil ('(' [',' .arguman+] ')' | '[' ',' .ifade+ ']' | '.' AD)*
```

Aritmetik

İşleç	Anlamı	Örnek	Sonuç
`+`, `-`, `*`	toplama, çıkarma, çarpma	`2 + 3 * 4`	`14`
`/`	bölme; sonuç her zaman ondalıktır	`7 / 2`	`3.5`
`böl`	tamsayı bölmesi; sıfıra doğru keser	`-7 böl 2`	`-3`
`mod`	kalan; işareti bölünenin işaretidir	`-7 mod 3`	`-1`
`**`	üs; sağdan birleşir	`2 ** 3 ** 2`	`512`
`@`	matris çarpımı (tensörlerde)	`A @ B`	

Tekli eksi üsten sonra uygulanır: `-2 ** 2` değeri `-4`'tür. İki tamsayının `+`, `-`, `*`, `böl`, `mod` ve `**` sonucu tamsayıdır; işlenenlerden biri ondalıksa sonuç ondalıktır. Sıfıra bölmek "Sıfıra bölme" hatası verir.

Karşılaştırma

`=`, `<>`, `<`, `>`, `< =`, `> =` mantıksal sonuç verir. Sayılar değerleriyle karşılaştırılır (`1 = 1.0` doğrudur), metinler sözlük sırasıyla. Listeler ve sözlükler yalnızca eşitlik bakımından (`=`, `<>`) karşılaştırılabilir; eşitlik eleman eleman sınanır. İlişki işleçleri zincirlenmez: `1 < x < 5` sözdizimi hatasıdır; `1 < x ve x < 5` yazılır.

`x içinde K`, `x` değeri `K` listesinin elemanı, `K` sözlüğünün anahtarı ya da `K` metninin bir parçası olduğunda doğrudur.

Mantık

`ve`, `veya` kısa devrelidir: sonuç sol işlenenle belli oluyorsa sağ işlenen hesaplanmaz. `yada` iki işlenenden yalnızca biri doğru olduğunda doğrudur. `değil` değeri tersine çevirir.

Mantık işleçleri karşılaştırmalardan sonra bağlar; bu yüzden `a > 1 ve b < 2` ayraç gerektirmez. İşlenenler tamsayıysa aynı işleçler bit işlemi yapar: `6 ve 3` değeri `2`, `6 veya 3` değeri `7`, `6 yada 3` değeri `5`'tir.

Boru

`x |> f` ifadesi `f(x)` ile, `x |> f(y)` ifadesi `f(x, y)` ile aynıdır: soldaki değer sağdaki çağrının ilk argümanı olur. Zincirleme soldan sağa okunur:

```
yaz([3, 1, 2] |> Sırala |> Ters, 1)
```

```
[3, 2, 1]
```

Birincil ifadeler

```
birincil: SAYI | METIN | AD | 'doğru' | 'yanlış' | 'boş' | 'kendi'
        | 'eğer' ifade 'ise' ifade 'değilse' ifade
        | 'miras' [AD]
        | adsiz_altprogram
        | '(' ifade ')'
        | koleksiyon
```

Koşullu ifade. `eğer k ise a değilse b`, `k` doğruysa `a`'nın, değilse `b`'nin değerini verir; yalnızca seçilen dal hesaplanır.

Koleksiyon sabitleri.

Yazım	Değer
`[]`	boş liste
`[1, 2, 3]`	liste
`[1..5]`	`[1, 2, 3, 4, 5]`
`[:]`	boş sözlük
`['a': 1, 'b': 2]`	sözlük

Son elemandan sonra virgül bırakılabilir.

Çağrı. `f(a, b)` alt programı çağırır. Parametresi olmayan bir alt program ayraçsız da çağrılabilir: `Selamla` ile `Selamla()` aynıdır. Deyim olarak tek başına yazılan bir alt program adı ise parametreleri olsa da çağrıdır; eksik argümanlar varsayılan değerlerini alır. Bir alt programı çağırmadan değer olarak almak için başına `@` yazılır: `@Selamla`.

Dizin. `L[i]` liste, sözlük, metin ve tensörlere uygulanır. Çok boyutlu erişim `M[i, j]` ya da `M[i][j]` biçiminde yazılır.

Üye erişimi. `n.ad` bir nesnenin alanına, metoduna ya da bir birimin dışa aktardığı ada erişir.

Biçimli argüman. Bir çağrının argümanı `değer:genişlik` ya da `değer:genişlik:basamak` biçiminde yazılabilir; değer metne çevrilip sağa yaslanır. `yaz(3.14159:8:2, 1)` çıktısı `3.14` olur.

Adsız alt programlar

```
adsiz_altprogram:
    ('fonksiyon' | 'prosedür') [parametreler] [':' tip] '=>' ifade
  | ('fonksiyon' | 'prosedür') [parametreler] [':' tip] bildirim* blok
```

```
kare := fonksiyon(x) => x * x
artır := fonksiyon(x: tamsayı): tamsayı
başla
  dön x + 1
bitir
```

Adsız alt programlar tanımlandıkları kapsamdaki değişkenleri yakalar ve onlar üzerinde çalışmayı, tanımlandıkları alt program döndükten sonra da sürdürür.

5. Deyimler

```
deyimler: (deyim (';' | satır sonu))*
govde:    deyim | boş
```

Döngü ve koşul yapılarının gövdesi **tek bir deyimdir**. Birden çok deyim çalıştırmak için gövde `başla … bitir` arasına alınır. `tekrarla … olana dek`, `değerine göre … bitir` ve `dene … bitir` kendi sınırlarını taşır.

Atama

```
atama: hedef (':=' | '+ =' | '- =' | '* =' | '/ =') ifade
```

Hedef bir değişken, alan (`n.x`) ya da dizin (`L[i]`) olabilir. `a + = b`, `a := a + b` ile aynıdır. Atama bir ifade değildir; karşılaştırma `=`, atama `:=` ile yazılır.

Çağrı deyimi

Tek başına yazılan bir çağrı, ad ya da üye erişimi alt programı çağırır ve varsa sonucunu atar: `Selamla`, `yaz('a', 1)`, `liste.Temizle`.

Blok içi değişken

```
değişken toplam := 0
```

Deyimlerin arasında yazılan `değişken` bildirimi, içinde bulunduğu alt programın kapsamında yeni bir değişken tanımlar.

Koşul

```
eger_deyimi: 'eğer' ifade 'ise' govde ['değilse' govde]
```

Koşul mantıksal olmalıdır. `değilse` en yakın `eğer`'e bağlanır.

```
eğer x > 0 ise yaz('pozitif', 1)
değilse eğer x < 0 ise yaz('negatif', 1)
değilse yaz('sıfır', 1)
```

Çok seçenekli karar

```
degerine_gore: ifade 'değerine' 'göre' kol* ['değilse' [':'] deyimler] 'bitir'
kol:           ','.etiket+ ':' govde
etiket:        ifade ['..' ifade]
```

İfade bir kez hesaplanır ve kollar yukarıdan aşağıya denenir; uyan ilk kolun gövdesi çalışır, diğerleri atlanır. Etiket tek bir değer ya da `alt..üst` aralığı olabilir. Hiçbir kol uymazsa `değilse` bölümü çalışır.

```
puan değerine göre
  90..100: harf := 'A'
  80..89:  harf := 'B'
  0, 1, 2: harf := 'X'
  değilse harf := 'F'
bitir
```

Sayan döngü

```
sayan_dongu:
    ifade AYRILMA ifade YONELME 'kadar' 'her' AD 'için' govde
  | ifade AYRILMA ifade YONELME 'adımda' ifade ('artan' | 'azalan')
    'her' AD 'için' govde
```

Yazım	Sayaç
`1'den 10'a kadar her i için`	1, 2, … 10
`10'dan 0'a kadar her j için`	10, 9, … 0
`a'dan b'ye adımda c artan her d için`	a, a+c, … (b'yi geçmeden)
`z'den y'ye adımda v azalan her u için`	z, z−v, … (y'nin altına inmeden)

Kurallar:

- Başlangıç, bitiş ve adım döngüye girilirken **bir kez** hesaplanır ve tamsayı olmalıdır.
- `kadar` birer birer sayar. İki sınır da tamsayı sabiti olarak yazılmışsa ve ilki büyükse sayaç azalır; bu karar derleme sırasında verilir.
- Sınırlardan biri değişken ya da hesaplanan bir ifadeyse `kadar` yalnızca ileriye sayar; başlangıç bitişten büyükse gövde hiç çalışmaz. Böylece boş bir listede `0'dan Uzunluk(L) - 1'e kadar` güvenle yazılabilir.
- `adımda` ile verilen adım pozitif bir tamsayı olmalıdır; yönü `artan` ya da `azalan` belirler. Yön sınırlarla uyuşmuyorsa gövde hiç çalışmaz.
- Sayaç bildirilmemişse döngü onu tanımlar. Döngü bittikten sonra sayacın değeri son turdaki değeridir.

Koleksiyon döngüsü

```
her_dongusu: ifade 'içindeki' 'her' AD [',' AD] 'için' govde
```

Liste, sözlük, metin ve aralık üzerinde dolaşır. Tek adla listede elemanlar, sözlükte anahtarlar, metinde karakterler gelir. İki adla listede `numara, eleman`, sözlükte `anahtar, değer` çiftleri gelir.

```
adlar içindeki her ad için yaz(ad, 1)
yaşlar içindeki her ad, yaş için yaz(ad + ': ' + yaş, 1)
```

Koşullu döngüler

```
iken_dongusu:    ifade 'iken' govde
tekrarla_deyimi: 'tekrarla' deyimler ifade 'olana' 'dek'
kez_dongusu:     ifade 'kez' 'tekrarla' govde
```

`koşul iken gövde`, koşul doğru olduğu sürece gövdeyi çalıştırır; koşul her turdan önce sınanır. `tekrarla … koşul olana dek` gövdeyi en az bir kez çalıştırır ve koşul doğru olunca durur. `n kez tekrarla gövde` gövdeyi `n` kez çalıştırır.

Döngü denetimi

`kır` en içteki döngüden çıkar. `devam` en içteki döngünün sıradaki turuna geçer.

Eleman silme cümlesi

```
eleman_silme: ifade ('listesindeki' | 'dizisindeki') konum '.' 'elemanı' 'Sil'
```

`L listesindeki 2. elemanı Sil`, `Sil(L, 2)` ile aynıdır. Konum bir sayı, bir ad ya da ayraç içinde bir ifade olabilir: `L listesindeki (i + 1). elemanı Sil`.

Dönüş

`dön ifade` fonksiyonun sonucunu belirler ve fonksiyondan çıkar. `dön` tek başına ya da `çık` deyimi, sonucu değiştirmeden alt programdan çıkar. Ana gövdede `çık` programı bitirir.

6. Alt programlar

```
altprogram: ('prosedür' | 'fonksiyon' | 'kurucu') [AD '.'] AD
            [parametreler] [':' tip] direktif*
            bildirim* blok

parametreler:    '(' [parametre_grubu ((';' | ',') parametre_grubu)*] ')'
parametre_grubu: ['değişken' | 'sabit'] ','.AD+ [':' tip] ['=' ifade]
```

`prosedür` değer döndürmez, `fonksiyon` döndürür. Alt programların kendi `sabit`, `değişken`, `tip` bölümleri ve iç içe alt programları olabilir.

Sonuç

Bir fonksiyonun sonucu iki yoldan verilir: `dön ifade` ile ya da `sonuç` adlı örtük değişkene atayarak. `sonuç`, dönüş tipinin başlangıç değeriyle başlar; fonksiyon `dön` yazılmadan biterse `sonuç`'un değeri döner.

```
fonksiyon Topla(a, b: tamsayı): tamsayı
başla
  sonuç := a + b
bitir
```

Parametreler

Yazım	Aktarım
`a: tamsayı`	Değerle; liste, sözlük ve nesnelerde aynı nesne paylaşılır
`değişken a: tamsayı`	Referansla; alt programdaki atama çağıranın değişkenini değiştirir
`sabit a: tamsayı`	Değerle; sözcük belgeleme amaçlıdır
`b: tamsayı = 10`	Varsayılan değerli; çağrıda yazılmayabilir

Tipi yazılmayan parametre her tipte değer alır. Parametre grupları `;` ya da `,` ile ayrılır. `değişken` parametreye yalnızca bir değişken, alan ya da dizin verilebilir. Eksik ya da fazla argüman hata verir.

Direktifler

Başlığın ardından yazılabilir: `sanal`, `ezer`, `soyut`, `ileri`, `aşırı`, `statik`. `ileri`, gövdesi daha sonra yazılacak bir alt programı önceden bildirir. `soyut`, gövdesi olmayan bir metodu bildirir; çağrılırsa hata verir. Diğerleri ayrıştırılır ama davranışı değiştirmez: bütün metodlar zaten sanaldır.

Özyineleme ve iç içe alt programlar

Alt programlar kendilerini çağırabilir. İçteki bir alt program, dıştakinin değişkenlerini görür ve değiştirebilir.

7. Sınıflar ve kayıtlar

```
sinif: 'sınıf' ['(' AD ')'] uye* 'bitir'
kayit: 'kayıt' alan* 'bitir'
uye:   gorunurluk | alan | metod
alan:  ','.AD+ [':' tip] ['=' ifade]
```

```
tip
  Hayvan = sınıf
    ad: metin
    kurucu Oluştur(a: metin)
    fonksiyon Ses: metin; sanal
    fonksiyon Tanıt: metin
  bitir

  Kedi = sınıf(Hayvan)
    fonksiyon Ses: metin; ezer
  bitir

kurucu Hayvan.Oluştur(a: metin)
başla
  ad := a
bitir

fonksiyon Hayvan.Ses: metin
başla
  dön '...'
bitir

fonksiyon Hayvan.Tanıt: metin
başla
```

```
  dön ad + ': ' + Ses
bitir

fonksiyon Kedi.Ses: metin
başla
  dön 'Miyav'
bitir
```

Metodlar

Bir metodun gövdesi sınıfın içinde ya da dışında yazılabilir. Sınıf içinde başlığın ardından `başla` (ya da yerel `değişken`/`sabit` bölümü) geliyorsa gövde oradadır; gelmiyorsa gövde sınıfın dışında `Sınıf.Metod` adıyla yazılır.

Metod gövdesinde nesnenin alanlarına ve metodlarına adlarıyla doğrudan ya da `kendi.ad` biçiminde erişilir. Bütün metodlar dinamik bağlanır: çağrılan metod, değişkenin bildirilen tipine göre değil, nesnenin gerçek sınıfına göre seçilir.

Nesne oluşturma

Yazım	Anlamı
`Kedi.Oluştur('Tekir')`	Kurucuyu adıyla çağırır
`Kedi('Tekir')`	Aynı kurucuyu sınıf adıyla çağırır
`Nokta.Oluştur(3, 4)`	Kurucusu olmayan sınıf ya da kayıtta alanları sırayla doldurur

Alanlar önce bildirimdeki ilk değerlerini (`sayaç: tamsayı = 0`) ya da tiplerinin başlangıç değerini alır, sonra kurucu çalışır.

Kalıtım

Bir sınıfın en çok bir ebeveyni olabilir: `sınıf(Ebeveyn)`. Alt sınıf ebeveynin alanlarını ve metodlarını devralır, aynı adla yeniden tanımladığı metod ebeveyninkinin yerini alır.

Ebeveynin metodu `miras` ile çağrılır: `miras Oluştur(a)` adı verilen metodu, tek başına `miras` ise içinde bulunulan metodun ebeveyndeki karşılığını aynı argümanlarla çağırır. `ÖrneğiMi(nesne, Sınıf)` nesnenin o sınıftan ya da alt sınıflarından olup olmadığını söyler.

Görünürlük

`özel`, `genel`, `korumalı` ve `yayınlanmış` sözcükleri sınıf içinde bölüm başlığı olarak yazılabilir. Bu sözcükler belgeleme amaçlıdır: derleyici erişimi kısıtlamaz.

Özel adlı metodlar

Metod	Kullanıldığı yer
`MetneÇevir`	`yaz(nesne)` ve metne çevirme
`Topla`, `Çıkar`, `Çarp`, `Bölme`	`+`, `-`, `*`, `/`
`Eşit`	`=` ve `<>`
`Küçük`, `Büyük`	`<`, `>`

Kayıtlar

Kayıt, metodu olmayan ve değer olarak davranan bir sınıftır: atandığında ve parametre olarak verildiğinde kopyalanır. Kayıt tipli bir değişken bildirildiğinde alanları başlangıç değerinde hazır gelir; sınıf tipli bir değişken ise bir nesne atanana kadar `boştur`.

8. Hatalar

```
dene_deyimi: 'dene' deyimler
             ['yakala' ['(' AD ')'] deyimler]
             ['sonunda' deyimler]
             'bitir'
```

`dene` bölümündeki deyimler çalışırken bir hata oluşursa denetim `yakala` bölümüne geçer; ayraç içindeki ad hata nesnesine bağlanır. `sonunda` bölümü, hata olsa da olmasa da, `dene` deyiminden çıkılırken çalışır. Yakalanmayan hata dıştaki `dene` deyimine, o da yoksa programın sonuna kadar yayılır; program "Çalışma hatası" iletisi ve 1 çıkış koduyla biter.

```
dene
  fırlat Hata('Geçersiz giriş', 'DoğrulamaHatası')
yakala (h)
  yaz(h.mesaj + ' / ' + h.tür, 1)
sonunda
  yaz('bitti', 1)
bitir
```

```
Geçersiz giriş / DoğrulamaHatası
bitti
```

`fırlat ifade` bir hata başlatır. `Hata(mesaj)` ya da `Hata(mesaj, tür)` bir hata nesnesi oluşturur; nesnenin `mesaj` ve `tür` alanları vardır. Kendi sınıflarınızın nesneleri de fırlatılabilir. Çalışma zamanı hataları (sıfıra bölme, sınır aşımı, tip uyuş-

mazlığı, tanımsız ad) aynı yolla yakalanır; türleri `Hata`'dır. `İddia(koşul, mesaj)` koşul yanlışsa hata fırlatır.

Sözdizimi hataları program çalışmadan önce bildirilir ve yakalanamaz.

9. Birimler

```
birim: 'birim' AD kullan*
       ['arayüz' kullan* arayuz_bildirimi* 'uygulama' kullan*]
       bildirim*
       ['başla' deyimler]
       'bitir' '.'
```

```
birim Geometri

arayüz
  sabit UZUNLUK_BİRİMİ: metin = 'cm'
  fonksiyon DaireAlanı(r: ondalık): ondalık

uygulama
  kullan Matematik

  fonksiyon DaireAlanı(r: ondalık): ondalık
  başla
    dön pi * r * r
  bitir

bitir.
```

`arayüz` bölümü birimin dışa açtığı adları bildirir; alt programlar burada yalnızca başlık olarak yazılır, gövdeleri `uygulama` bölümündedir. `uygulama` bölümünde bildirilip `arayüz`'de yer almayan adlar birimin dışından görünmez. `arayüz` bölümü yoksa birimdeki bütün adlar dışa açılır. İsteğe bağlı `başla` bölümü, birim ilk kez yüklendiğinde bir kez çalışır.

`kullan Geometri` birimi yükler. Dışa açılan adlar doğrudan (`DaireAlanı(2)`) ya da nitelikli (`Geometri.DaireAlanı(2)`) kullanılabilir. Bir birim, kaç yerden kullanılırsa kullanılsın, yalnızca bir kez yüklenir.

Birim dosyaları şu sırayla aranır: programın klasörü, onun `kutuphane/` alt klasörü, Yöntem'in kendi `kutuphane/` klasörü ve `YONTEM_YOL` ortam değişkenindeki klasörler; `yontem paket` ile kurulan paketler projenin `paketler/` klasöründen bulunur. Yerleşik birimler (`Matematik`, `Sistem`, `YZ`, `Kuantum`, `Meşale`, `Pencere`, `Ağ` ve diğerleri) dosya aranmadan bulunur.

10. Giriş ve çıkış

Dilde ayrı bir "satır sonu" komutu yoktur; `yaz` ve `oku` komutlarının ikinci parametresi kaç satır ilerleneceğini belirtir.

Çağrı	Anlamı
`yaz(değer)`	Değeri yazar, aynı satırda kalır
`yaz(değer, 1)`	Değeri yazar ve alt satıra geçer
`yaz('', 2)`	İki satır ilerler
`oku(x)`	x sayıysa sıradaki sözcüğü, metinse satırın kalanını okur
`oku(x, 1)`	Okur ve sonraki satıra geçer

`oku` okuduğu bilgiyi değişkenin tipine çevirir. Değerler metne şöyle çevrilir: tamsayılar olduğu gibi, ondalıklar en az bir basamakla (`3.0`), mantıksal değerler `doğru`/`yanlış`, `boş` değeri `boş`, listeler `[1, 2]`, sözlükler `['a': 1]`, nesneler `MetneÇevir` metoduyla.

11. Katı kip

Dosyada `{$katı}` yönergesi varsa (ya da `yontem.json` içinde `"kati": true` yazıyorsa, ya da derleyici `--kati` seçeneğiyle çalıştırılmışsa) derleyici tipleri program çalışmadan önce denetler ve bulduğu bütün tip hatalarını bir arada listeler.

Katı kipte:

- her değişken, parametre ve fonksiyon sonucu tipiyle bildirilmelidir;
- atamalar, argümanlar, dönüş değerleri ve işleçlerin işlenenleri tiplerine göre denetlenir;
- `tamsayı listesi` gibi eleman tipi bildirilmiş koleksiyonlara uymayan eleman konamaz;
- sınıf ve kayıt tiplerinde olmayan bir alana ya da metoda erişilemez;
- tanımsız adlar derleme sırasında bildirilir.

`herhangi` tipli değerler ve eleman tipi yazılmamış `liste` denetim dışında kalır ve çalışırken denetlenir. Yönerge yalnızca bulunduğu dosyayı etkiler: katı bir program katı olmayan bir birimi kullanabilir. Katı kip JavaScript derleyicisindedir; Python yorumlayıcısı yönergeyi yorum olarak görür ve programı aynı sonuçla çalıştırır.

12. Uygulama farkları

Konu	JavaScript	Python	Gömülü (C++)
Tamsayı büyüklüğü	sınırsız	sınırsız	64 bit
Katı kip denetimi	var	yönerge yok sayılır	tipler her zaman statiktir
Tipsiz değişken	var	var	tipi ilk değerden çıkarılır
`herhangi` tipi	var	var	yok
Hata ayıklayıcı, paket komutları	var	yok	yok
Etkileşimli kabuk	yok	var	yok
Birimler	hepsi	hepsi	`Matematik`, `YZ`, `Donanım`

Gömülü arka uç, tipleri derleme sırasında belirlenemeyen programları açıklayıcı bir hatayla reddeder.

EK

A

Dilbilgisinin tamamı

Aşağıdaki kurallar depodaki `yontem.gram` dosyasının açıklamaları çıkarılmış özetidir. Sözcük tanımları ve ayrıntılı açıklamalar için o dosyaya bakabilirsiniz.

```
dosya:    birim | program
program:  ['program' AD ayirici] kullan* bildirim* blok ('.' | SON)
birim:    'birim' AD ayirici kullan*
          ['arayüz' kullan* arayuz_bildirimi* 'uygulama' kullan*]
          bildirim* ['başla' deyimler] 'bitir' ['.']
kullan:   'kullan' ',' .AD+ ayirici
ayirici:  ';' | satır sonu | SON

bildirim: sabit_bolumu | degisken_bolumu | tip_bolumu | altprogram
arayuz_bildirimi:
          sabit_bolumu | degisken_bolumu | tip_bolumu | altprogram_basligi
sabit_bolumu:    'sabit' (AD [':' tip] ('=' | ':=') ifade ayirici)*
degisken_bolumu: 'değişken' (degisken_grubu ayirici)*
degisken_grubu:  ',' .AD+ ':' tip [('=' | ':=') ifade]
               | ',' .AD+ ('=' | ':=') ifade
```

```
tip_bolumu:      'tip' (AD '=' tip_tanimi ayirici)*
tip_tanimi:      sinif | kayit | tip

tip:        tip_tabani tip_eki*
tip_tabani: 'dizi' | [AD '.'] AD
tip_eki:    'dizisi' | 'listesi' | 'sözlüğü'

sinif:       'sınıf' ['(' AD ')'] sinif_uyesi* 'bitir'
kayit:       'kayıt' kayit_uyesi* 'bitir'
sinif_uyesi: gorunurluk | 'değişken' | metod | degisken_grubu ayirici
kayit_uyesi: gorunurluk | 'değişken' | degisken_grubu ayirici
gorunurluk:  'özel' | 'genel' | 'korumalı' | 'yayınlanmış'
metod:       altprogram_basligi [bildirim* blok ayirici]

altprogram:  altprogram_basligi [bildirim* blok ayirici]
altprogram_basligi:
             ('prosedür' | 'fonksiyon' | 'kurucu') [AD '.'] AD
             [parametreler] [':' tip] ayirici (direktif ayirici)*
direktif:    'sanal' | 'ezer' | 'soyut' | 'ileri' | 'aşırı' | 'statik'
parametreler:
             '(' [parametre_grubu ((';' | ',') parametre_grubu)*] ')'
parametre_grubu:
             ['değişken' | 'sabit'] ','.AD+ [':' tip] [('=' | ':=') ifade]

blok:     'başla' deyimler 'bitir'
deyimler: (deyim ayirici)*
govde:    deyim | boş
deyim:    blok
        | 'eğer' ifade 'ise' govde [[';'] 'değilse' govde]
        | 'tekrarla' deyimler ifade 'olana' 'dek'
        | 'dene' deyimler ['yakala' ['(' AD ')'] deyimler]
          ['sonunda' deyimler] 'bitir'
        | 'fırlat' ifade
        | 'dön' [ifade]
        | 'çık' | 'kır' | 'devam'
        | 'değişken' degisken_grubu
        | ifade AYRILMA ifade YONELME 'kadar' 'her' AD 'için' govde
        | ifade AYRILMA ifade YONELME 'adımda' ifade
          ('artan' | 'azalan') 'her' AD 'için' govde
        | ifade 'içindeki' 'her' AD [',' AD] 'için' govde
        | ifade ('listesindeki' | 'dizisindeki') sonek
          '.' ('elemanı' | 'elemanını') 'Sil'
        | ifade 'iken' govde
        | ifade 'kez' 'tekrarla' govde
        | ifade 'değerine' 'göre' kol*
          ['değilse' [':'] deyimler] 'bitir'
        | sonek (':=' | '+ =' | '- =' | '* =' | '/ =') ifade
        | ifade
kol:      ','.etiket+ ':' govde ayirici
etiket:   ifade ['..' ifade]

ifade:    veya ('|>' veya)*
```

```
veya:     ve (('veya' | 'yada') ve)*
ve:       degil ('ve' degil)*
degil:    'değil' degil | iliski
iliski:   toplama [('=' | '<>' | '<' | '>' | '< =' | '> =' | 'içinde')
          toplama]
toplama:  carpma (('+' | '-') carpma)*
carpma:   tekli (('*' | '/' | '@' | 'böl' | 'mod') tekli)*
tekli:    ('-' | '+') tekli | '@' sonek | sonek ['**' tekli]
sonek:    birincil sonek_eki*
sonek_eki:
          '(' [',' .arguman+] ')' | '[' ',' .ifade+ ']' | '.' (AD | AS)
arguman:  ifade [':' ifade [':' ifade]]
birincil: SAYI | METIN | AD | 'doğru' | 'yanlış' | 'boş' | 'kendi'
        | 'eğer' ifade 'ise' ifade 'değilse' ifade
        | 'miras' [AD]
        | ('fonksiyon' | 'prosedür') [parametreler] [':' tip]
          ('=>' ifade | bildirim* blok)
        | '(' ifade ')'
        | '[' ']' | '[' ':' ']'
        | '[' ifade '..' ifade ']'
        | '[' ifade ':' ifade (',' ifade ':' ifade)* [','] ']'
        | '[' ',' .ifade+ [','] ']'
```

EK

B

Yerleşik işlevler

Her programda, birim kullanmadan erişilebilen işlevler:

Alan	İşlevler
Giriş ve çıkış	`yaz`, `oku`
Dönüşüm ve tip	`tamsayı`, `ondalık`, `metin`, `mantıksal`, `liste`, `Tür`, `ÖrneğiMi`
Koleksiyon	`Uzunluk`, `Ekle`, `Sil`, `Aralık`, `Eşle`, `Süz`, `İndirge`, `Sırala`, `Ters`, `Toplam`, `Maks`, `Min`, `Eşleştir`, `Numaralandır`, `Dilim`, `Kopya`, `Anahtarlar`, `Değerler`, `İçerir`, `Benzersiz`, `Say`
Metin	`AltMetin`, `Konum`, `BüyükHarf`, `KüçükHarf`, `Ayır`, `Birleştir`, `Kırp`, `Değiştir`, `Biçimle`, `Sıra`, `Karakter`, `HarfMi`, `RakamMı`
Sayısal	`Mutlak`, `Kare`, `Kök`, `Yuvarla`, `Kes`, `Rastgele`, `Rastgeleleştir`, `Artır`, `Azalt`, `Değiş`, `pi`

Alan	İşlevler
Denetim	`Hata`, `İddia`, `Durdur`, `Çağır`, `ParametreSayısı`, `Parametre`

Birimlerin sunduğu işlevler *Yöntem Birimler ve Araçlar Kılavuzu*'nda ve README dosyasında anlatılır.