GUI Tabanlı Python-Benzeri Kod Düzenleyici: Teknik Belgeleme

Bu proje üç ana bileşen içermektedir: `lexical_analyzer.py` (leksik analizör), `parser.py` (sözdizimi analizi ve ayrıştırıcı) ve `gui.py` (sözdizimi vurgulama ve grafik arayüz). Uygulama Python ile yazılmıştır. Oluşturulan mini-dil, Python'a benzer bir sözdizimine sahiptir; blok yapıları girintilerle belirlenir ve `if` / `elif` / `else` yapıları, `while` ve `for` döngüleri, `def` ile fonksiyon tanımlama ile `return` ifadeleri desteklenir. Yorumlar `#` karakteri ile başlar ve kod içinde metin blokları `<label>` etiketleri ve renkli işaretlerle vurgulanır. Aşağıda her bileşenin işlevi ve uygulama detayları incelenmektedir.

Dil ve Dilbilgisi Seçimi

Bu düzenleyicide Python tarzı bir dil işlenmektedir. Değişken ve fonksiyon adları bir harf ya da alt çizgi ile başlayıp harf veya rakamlarla devam edebilir (`ID` token). Tam sayı ve ondalıklı sayılar (örneğin `123`, `3.14`) `NUMBER` tokenı olarak tanımlanır. Anahtar kelimeler yalnızca `if`, `else`, `while`, `for`, `def`, `return`, `in` şeklindedir; bir `ID` tokenının değeri bu anahtar kelimelerden birine eşit olduğunda türü `KEYWORD` olarak değiştirilir. Aritmetik ve karşılaştırma operatörleri (`+`, `-`, `*`, `/`, `==`, `!=`, `<=`, `>=`, `<`, `>`) `OP` tokenı; atama operatörü `=` ise `ASSIGN` tokenıdır. Parantez (`(`, `)`), iki nokta (`:`) ve virgül (`,`) karakterleri sırasıyla `LPAREN`, `RPAREN`, `COLON` ve `COMMA` tokenlarıdır. Python'dan farklı olarak bu dilde bloklar süslü parantezlerle değil girintilerle ifade edilir. Her satır başındaki boşluk miktarı takip edilerek girinti artışında `INDENT`, azalışında `DEDENT` tokenları üretilir; böylece blokların başlangıcı ve sonu girintilerle belirlenir. Leksik analizör bu kuralları düzenli ifadeler (regex) ile uygular; derleyici teorisinde leksik analizörün düzenli ifadeler veya DFA kullanarak token tanıması yaygın bir yöntemdir.

Leksiksel Analiz Detayları

Kaynak kod satır satır okunur ve her parça en uzun eşleşen tokena dönüştürülür. Her satırın başındaki boşluk miktarı ölçülerek girinti artışında `INDENT`, azalışında `DEDENT` tokenları eklenir; böylece blokların başlangıcı ve sonu girintilerle belirlenir. Tanımlanan token türleri şunlardır:

- **NUMBER:** Tam veya ondalıklı sayılar (örneğin `123`, `3.14`).
- **ID:** Değişken veya fonksiyon isimleri; bir harf ya da alt çizgi ile başlayıp harf ve rakamlardan oluşur (örneğin `myVar`, `_temp`). Bir `ID` tokenının değeri anahtar kelimelerden birine eşitse türü `KEYWORD` olarak değiştirilir.
- **KEYWORD:** `if`, `else`, `while`, `for`, `def`, `return`, `in` gibi özel anahtar kelimeler.
- **OP:** Aritmetik ve karşılaştırma operatörleri (`+`, `-`, `*`, `/`, `==`, `!=`, `<=`, `>=`, `<`, `>`).
- **ASSIGN:** Atama operatörü `=`.
- **LPAREN**, **RPAREN**, **COLON**, **COMMA:** Sırasıyla sol parantez `(`, sağ parantez `)`, iki nokta `:`, virgül `,` karakterleri.
- **NEWLINE:** Satır sonu karakteri (`\n`).
- **SKIP:** Boşluk ve sekme karakterleri (token olarak atlanır).
- **COMMENT:** Satır içi yorumlar (`#` ile başlayan). Yorum tespit edildiğinde bir `COMMENT` tokenı eklenir ve satırın geri kalanı işleme alınmaz.

- **INDENT**, **DEDENT:** Girinti değişikliklerine bağlı olarak blok başı (`INDENT`) ve sonu (`DEDENT`) tokenları.
- **EOF:** Dosya sonu işaretçisi; kaynak tamamen ayrıştırıldıktan sonra eklenir.

Her bir eşleşen parça için tür, değer, satır ve sütun bilgilerini içeren bir Token nesnesi oluşturulur. Tanımlanamayan bir karakterle karşılaşılırsa `SyntaxError` fırlatılır; böylece geçersiz semboller gibi basit sözdizimi hataları erken aşamada tespit edilip kullanıcıya bildirilebilir.

Sözdizimi Analizi Süreci

GUI uygulamasında her tuş basımında veya metin değişiminde `lex()` fonksiyonu çağrılır. Üretilen token listesinde **COMMENT** türündeki öğeler sözdizimi vurgulaması için kullanılırken, sözdizimi denetimi için yorumlar listeden çıkarılır ve kalan tokenlar Parser sınıfına iletilir. `Parser(tokens).parse()` metodu tüm kodu baştan sona tarar ve yapısal akıştan sapma olduğunda `ParseError` fırlatır. Örneğin `parse()` metodu önce `statements()` yöntemi ile tüm satırları ayrıştırır, sonra son tokenın dosya sonu simgesi (`EOF`) olduğunu kontrol eder. Hata yoksa arayüzün durum çubuğunda “Geçerli sözdizimi” mesajı gösterilir; hata varsa ilgili satır ve sütun bilgisiyle bir hata mesajı görüntülenir. Böylece kullanıcıya anlık geribildirim sağlanır. Parser, her dil kuralını ilgili yöntemlerle kontrol eden rekürsif-çıkışlı (recursive descent) bir parser olarak tasarlanmıştır.

Ayrıştırma (Parsing) Yöntemi

Parser, her bir dil yapısı için ayrı yöntemler içerir. En üstte `statements()` metodu yer alır; bu metot satır başında bir `KEYWORD` veya `ID` tokenı bulunduğu sürece döngüye devam eder ve satırın ilk kelimesine göre uygun alt yöntemi çağırır. Desteklenen yapılar ve kuralları şunlardır:

- **If/Elif/Else Blokları:** `if <ifade>:` satırının ardından bir girintili blok beklenir. Gerekli `INDENT` / `DEDENT` tokenları eklenir. İsteğe bağlı `elif <ifade>:` ve `else:` blokları da benzer şekilde işlenir; her bloğun sonunda `DEDENT` ile çıkılır.
- **While Döngüsü:** `while <ifade>:` yapısı ile başlanır. Parser önce `while` ve koşul ifadesini okur, ardından `:` ve yeni satırı kontrol eder; iç blok `statements()` yöntemi ile ayrıştırılır.
- **For Döngüsü:** `for <ID> in <ifade>:` biçimindedir. Bu yapıda `for` anahtar kelimesi, bir değişken adı, `in` anahtar kelimesi ve bir ifade yer alır. İki nokta ardından yeni satırda girinti gereklidir; iç blok `statements()` yöntemi ile işlenir.
- **Fonksiyon Tanımı:** `def <ID>(parametreler):` satırı işlenir. Parametreler bir veya daha fazla `ID` olabilir, virgülle ayrılır. İki noktadan sonraki yeni satır ve girintili blok (fonksiyon gövdesi) kontrol edilir.
- **Return İfadesi:** `return <ifade>` satırı ayrıştırılır; satır sonunda yeni satır veya blok sonu gelmelidir.
- **Atama İfadesi:** `ID = <ifade>` biçiminde ele alınır. Önce değişken adı, ardından eşittir ve bir ifade gelmesi beklenir.
- **İfade (Expression):** Aritmetik ve karşılaştırmalı ifadeler için kurallar tanımlanmıştır. Örneğin toplama/çıkarma `add_expr`, çarpma/bölme `term` ile ayrıştırılır; bir factor olarak sayı (`NUMBER`), değişken (`ID`), parantezli alt ifade veya fonksiyon çağrısı kabul edilir. Her kurala uygun yöntemlerle rekürsif ayrıştırma yapılır; beklenmeyen bir token görüldüğünde hata fırlatılır.

Sözdizimi Vurgulama Şeması

Editör penceresinde sözdizimi vurgulaması (syntax highlighting) token türlerine göre renkli etiketler kullanılarak yapılır. Tkinter’ın Text widget’ı üzerinde `keyword`, `number`, `comment`, `operator` gibi

etiketler tanımlanır ve her birine renk atanır (örneğin anahtar kelimeler mavi, sayılar mor, yorumlar yeşil). Kullanıcı kod yazarken her tuş basımında gerçekleşen `on_key_release` olayı ile önce mevcut etiketler `text.tag_remove` ile temizlenir.

Ardından her token için uygun etiket adı belirlenir (örneğin `KEYWORD` için `keyword`, `NUMBER` için `number` vb.) ve `text.tag_add` fonksiyonu kullanılarak bu etiketler ilgili kod aralığına uygulanır. Sonuç olarak editörde renkli vurgulamalar görünür hale gelir ve kodun yapısı görsel olarak ayrılmış olur. Örneğin `if` kelimesi mavi `keyword` etiketiyle, `123` sayısı mor `number` etiketiyle vurgulanır.

GUI Uygulaması

Uygulamanın arayüzü Tkinter kütüphanesi ile oluşturulmuştur. Ana pencerede Consolas fontunda geniş bir Text widget bulunur; bu widget kod girişi için metin alanı sağlar. Pencerenin altında yer alan durum çubuğu ise bir Label ile gösterilir; bu kısımda sözdizimi hatası veya başarı mesajları görüntülenir.

Her tuş basımında tetiklenen `on_key_release` fonksiyonu, `text.get("1.0","end-1c")` ile kodu alır ve `lex()` ile tokenlara ayırır. Eski etiketler temizlendikten sonra yenileri uygulanır ve `Parser(tokens).parse()` ile sözdizimi kontrolü yapılır. Hata bulunursa durum etiketine ilgili hata mesajı yazılır; hata yoksa “Geçerli sözdizimi” mesajı görüntülenir.

Bu işlev `text.bind("<KeyRelease>", on_key_release)` koduyla klavye olayıyla ilişkilendirilmiştir. Pencere döngüsü `root.mainloop()` ile başlatılır. Böylece kod düzenleme, eşzamanlı sözdizimi denetimi ve vurgulama işlevleri bir arada çalışır.

Demo ve Yayımlanan Makale

Projenin temel özelliklerini gösteren bir demo videosu halka açık bir platformda (YouTube) yayımlanmıştır. Ayrıca teknik uygulama detaylarını anlatan kapsamlı bir makale web üzerinde erişilebilir durumdadır. Bu teknik doküman, öğretmenler ve geliştiricilerin düzenleyiciyi anlayıp genişletmeleri için gerekli tüm bilgileri içermektedir. Aynı zamanda bu belge ders materyali veya araştırma amaçlı çalışmalar için de uygun bir kaynak olarak düzenlenmiştir.
