

Real-Time Syntax Highlighter

1. Giriş:

Bu proje, kullanıcıların yazdığı basit bir programlama diline ait kodların hem sözdizimsel doğruluğunu denetleyen hem de bu kodları sözdizimine göre renklendiren, grafiksel arayüz tabanlı bir IDE uygulamasıdır. Python programlama dili kullanılarak geliştirilmiştir ve arayüz için thinker kütüphanesi kullanılmıştır.

Projede, bir kaynağın önce **lexical analiz** aşamasından geçirilip token'lara ayrılması, ardından **sözdizim analizi (parsing)** yapılarak geçerli olup olmadığının denetlenmesi ve son olarak bu işlemlerle entegre çalışan **sözdizimi renklendirme (syntax highlighting)** özelliğiyle kullanıcıya görsel geri bildirim sunulması hedeflenmiştir. Kullanıcı tarafından yazılan kod anlık olarak analiz edilir, hatalı satırlar vurgulanır ve arayüz başlığında uyarı gösterilir.

2. Dil ve Gramer Tercihi:

Projede kullanılan dil temel ifadeler içeren küçük yapay bir dildir. Bu dilin sözdizimi şu yapılardan oluşur:

- If, else, while, return gibi temel kontrol ifadeleri
- =, ==, !=, <=, >=, <, > gibi karşılaştırma ve atama operatörleri
- Sayılar
- Değişken isimleri
- Süslü parantez {}, normal parantez (), noktalı virgül ; gibi ayraçlar

Bu dilin grameri öngörülebilir ve sol rekürsif olmayan şekilde tasarlanmıştır. Bu sayede elle yazılmış bir sözdizim analizcisi ile kolayca işlenebilir hale getirilmiştir.

3. Sözdizim Analiz Süreci:

Kullanıcı tarafından kod girildiğinde şunlar gerçekleşir:

1. Kod lexer.py dosyasındaki tokenize() fonksiyonuyla token'lara ayrılır.
2. Bu token listesi parser.py içinde yer alan Parser sınıfına aktarılır.
3. Parser sınıfı, statement() fonksiyonuyla kodun hangi yapıya ait olduğunu tespit eder:
 - a. if yapısı → if_statement()
 - b. while yapısı → while_statement()
 - c. Atama işlemi → assignment()
 - d. return ifadesi → return_statement()
4. Eğer bu yapılar tanımlı kurallara uymuyorsa, hata satırı tespit edilip GUI'ye bildirilir.

4. Lexical Analiz Detayları:

Lexical analiz, lexer.py dosyasında gerçekleştirilmiştir. TOKEN_TYPES listesi ile token türleri ve karşılık gelen düzenli ifadeler tanımlanmıştır:

- Keyword
- Identifier
- Number
- Comment
- Operatör
- Delimiter

WHITESPACE (boşluklar) token olarak kaydedilmez. Diğer token'ların pozisyonu doğru hesaplanabilsin diye sadece göz önünde bulundurulur.

```
3  # Token tanımları
4  TOKEN_TYPES = [
5      ('COMMENT',      r'//[^\n]*'),          # Yorumlar
6      ('KEYWORD',      r'\b(if|else|while|return)\b'),  # Anahtar kelimeler
7      ('NUMBER',       r'\b\d+(\.\d+)?\b'),          # Sayılar
8      ('IDENTIFIER',   r'\b[a-zA-Z_][a-zA-Z0-9_]*\b'),  # Değişken/adlar
9      ('OPERATOR',     r'==|!=|<=|>=|+=\-*|<>'),      # Operatörler
10     ('DELIMITER',    r'[\(\)\{\}\;\,]'),          # Ayraçlar
11     ('WHITESPACE',   r'\s+'),                  # Boşluklar
12 ]
```

5. Pars Etme Yöntemi:

Sözdizim analizcisi (parser), Parser sınıfı içinde her yapıya karşılık gelen bir kontrol fonksiyonu ile tanımlanmıştır. Örneğin:

- `if_statement()`: if koşulu ve opsiyonel else bloğunu kontrol eder.
- `assignment()`: `IDENTIFIER = EXPRESSION`; şeklinde yazılmış ifadeleri denetler.
- `statement_block()`: `{}` ile çevrili blokların içinde ardışık ifadeleri kontrol eder.

Bu yapı, kolay genişletilebilir ve hatanın tespit edildiği satırı kullanıcıya göstermeye elverişlidir.

```
6 //hatalı blok - parantez kapanmamış
7 if(a<=b){
8     b=a*5;
9     a=b+6;
10
```

6. Renklendirme Şeması:

Renklendirme işlemi, token'lar üzerinden gerçekleştirilir. Her token tipi için `TOKEN_COLORS` sözlüğünde tanımlı olan bir renk kullanılır:

- `KEYWORD` → Mavi (blue)
- `IDENTIFIER` → Mor (purple)
- `NUMBER` → Yeşil (green)
- `OPERATOR` → Kırmızı (red)
- `DELIMITER` → Koyu turuncu (darkorange)
- `COMMENT` → Gri (gray)

Kullanıcı kod yazdıkça `highlight()` fonksiyonu sürekli çalışır ve her token'ın başlangıç ve bitiş pozisyonu hesaplanarak uygun renklendirme yapılır.

Hatalı satırların arkaplanı pembe ton (mistyrose) ile renklendirilir.

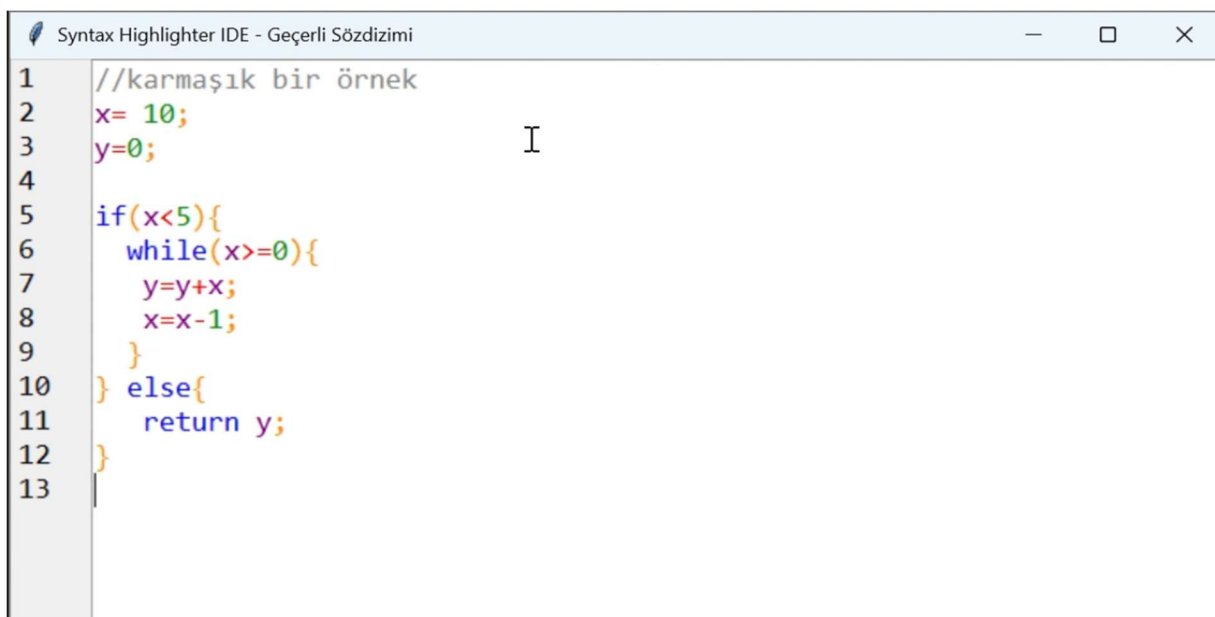
```
5 //hatalı
6 if x==19{
7     x=20;}
8 else { x=2;}
```

7. Arayüz Gerçekleştirme:

Kullanıcı arayüzü tkinter modülü ile gerçekleştirilmiştir. Arayüzde şu bölümler yer alır:

- **Kod Editörü:** Kullanıcının kod girdiği, renklendirmenin yapıldığı alan.
- **Satır Numaraları:** Solda yer alır ve her değişiklikte güncellenerek kaçınıcı satırda olunduğu gösterilir.
- **Hata Vurgulama:** Hatalı satırlar arka plan rengi ile (mistyrose) vurgulanır.
- **Başlık Güncelleme:** Sözdizimi geçerli ise “Geçerli Sözdizimi”, değilse “Hatalı Sözdizimi” olarak pencere başlığı değişir.

Bu sayede kullanıcıya okuma kolaylığı sağlanır.



```
Syntax Highlighter IDE - Geçerli Sözdizimi
1 //karmaşık bir örnek
2 x= 10;
3 y=0;
4
5 if(x<5){
6     while(x>=0){
7         y=y+x;
8         x=x-1;
9     }
10 } else{
11     return y;
12 }
13
```