"Real-Time Grammar-Based Syntax Highlighter with GUI in Python"

---

1. Giriş (Introduction)

Bu projede, gerçek zamanlı olarak çalışan, gramer tabanlı bir sözdizimi vurgulayıcı geliştirildi. Proje kapsamında

- Regular expression ile çalışan bir lexical analyzer (tokenizer)
- Context-Free Grammar (CFG) tabanlı bir top-down parser
- Gerçek zamanlı parse kontrolü ve hatalı kodu vurgulayan bir GUI

uygulamaya geçirildi. Syntax highlighting için hiçbir hazır kütüphane kullanılmadı.

---

2. Programlama Dili Seçimi (Language Choice)

Python dili seçildi çünkü:

- Regex işlemleri için güçlü re modülüne sahip.
- tkinter ile GUI kolayca geliştirilebiliyor.
- Hızlı prototipleme için uygun ve okunabilirliği yüksek.

---

3. Gramer Tanımı (Grammar Specification)

Basitleştirilmiş bir Python alt kümesi tanımlandı. Desteklenen yapılar:

- Fonksiyon tanımları: def foo(x):
- Koşullar: if, else, while
- Aritmetik işlemler: +, -, *, /
- Dönüşler: return
- String, sayı, tanımlayıcı, operatör, delimiter ve yorumlar

Bu gramer, top-down parser tarafından analiz edilebilir hale getirildi.

---

4. Lexical Analyzer (Tokenizer)

Tokenizer, aşağıdaki token türlerini tespit eder:

- KEYWORD: if, else, def, return vb.
- IDENTIFIER: değişken/fonksiyon isimleri
- NUMBER: tamsayılar
- STRING: çift veya tek tırnakla tanımlanan string'ler
- OPERATOR: +, -, =, ==, vb.
- DELIMITER: (,), :, ,
- COMMENT: # ile başlayan satırlar
- NEWLINE: satır sonları

Tokenizer, regular expression + state-based logic yaklaşımı ile yazıldı.

---

5. Parser (Top-Down Syntax Analyzer)

Parser recursive descent (top-down) mantığıyla yazıldı:

- parse_stmt_list, parse_stmt, parse_expr, parse_arith_expr gibi fonksiyonlar, grammar'ı adım adım yürütüyor.
- match() ve expect() fonksiyonları ile token tipi kontrol ediliyor.
- Syntax hataları tespit edildiğinde SyntaxError fırlatılıyor.

---

6. Gerçek Zamanlı Syntax Vurgulama (Real-Time Highlighting)

- Kullanıcının yazdığı her değişiklik sonrası tokenize() fonksiyonu çağrılıyor.
- Her token tipi GUI'de farklı renklerle vurgulanıyor.
- Hatalı tokenlar SYNTAX_ERROR etiketi ile arka planı kırmızı olarak işaretleniyor.

---

7. Kullanıcı Arayüzü (GUI with Tkinter)

- tk.Text ile kod giriş kutusu

- Renkli syntax vurgusu için .tag_add() kullanıldı
- Parse hatası olursa kırmızı label'da kullanıcıya gösteriliyor
- "Parse Kontrolü" butonuyla elle kontrol yapılabiliyor

---

8. Zorluklar ve Çözümler

- Parantez ve indent yapılarında hata tespiti
- Kapanmayan stringler
- Parser'ın NEWLINE ve COMMENT ile uyumlu çalışmasını sağlama

---

9. Sonuç ve Geliştirme Fikirleri

Bu proje sayesinde:

- Gerçek zamanlı parse işlemi başarıyla sağlandı
- Token tipi ayırma ve GUI vurgulama çalıştı
- Kullanıcıya anlık geri bildirim verildi

📌 **Geliştirme Önerileri:**

- İndent bazlı gramer desteği (Python gibi)
- Hata türlerine göre detaylı uyarılar
- Daha fazla token tipi: boolean, list, dict, vs.
