

**PROGRAMLAMA DİLLERİ DERSİ**

**Real-Time Grammar-Based Syntax Highlighter with GUI**

**Proje Ödevi Raporu**

**BÜŞRA YESİN**

23360859076

HAZİRAN 2025

# 1. Programlama Dili

Bu proje, basit sözdizimi kurallarını kontrol eden bir sözdizimi denetleyici ve vurgulayıcı geliştirmek için **Python** programlama dili kullanılarak geliştirildi. Python’un:

* re (regular expressions) modülü ile güçlü bir şekilde metinden token çıkarmak mümkün oldu.
* tkinter modülü ile de basit ve etkili bir grafik arayüz (GUI) tasarlandı.

# 2. Sözdizimi Analizi (Syntax Analysis)

Projenin sözdizimi analiz bölümü, bir **top-down (yukarıdan aşağı) parser** ile gerçekleştirildi. Bu analiz, lexer’dan gelen token listesini sırasıyla kontrol ederek dilin kurallarına uygun olup olmadığını denetler. Desteklenen sözdizim yapıları şunlardır:

* if ... : ve opsiyonel else : blokları
* def function\_name(...): fonksiyon tanımları
* return ifadeleri
* variable = expression şeklinde atamalar
* Parantezli ve aritmetik ifadeler

✅ Doğru sözdizimi olduğunda GUI'de "Syntax is correct!" mesajı görüntülenir.  
❌ Hatalı olduğunda "Syntax error!" mesajı gösterilir.

# 3. Sözcüksel Çözümleyici (Lexical Analysis)

Sözcüksel analiz işlemi, re modülü kullanılarak yazılan bir tokenize fonksiyonu ile yapılır. Bu fonksiyon girilen metni tanımlı kurallara göre parçalayarak Token nesnelerine dönüştürür. Kullanılan token türleri:

| **Token Türü** | **Açıklama** |
| --- | --- |
| KEYWORD | Anahtar kelimeler (if, else, def, return) |
| ID | Değişken ve fonksiyon isimleri |
| NUMBER | Sayılar |
| OP | Operatörler (=, ==, +, \*, vs.) |
| COLON | : karakteri (blok başlangıcı) |
| LPAREN / RPAREN | Parantezler |
| COMMA | Virgül |
| COMMENT | # ile başlayan yorum satırları |

Boşluklar atlanır, geçersiz karakterler durumunda hata fırlatılır.

# 4. Ayrıştırıcı (Parser)

Sözdizim analizini gerçekleştiren parser sınıfı şu yapıya sahiptir:

* **Top-down (yukarıdan aşağı)** yaklaşımı kullanılmıştır (recursive descent parser).
* Her ifade için statement, expression, block, condition gibi ayrı işlevler yazılmıştır.
* Yorum satırları ve boşluklar analiz dışı bırakılır.
* Token'lar sırayla kontrol edilerek program yapısının doğru olup olmadığı test edilir.

Parser, belirli bir token türünü bekleyen match(expected\_type) fonksiyonuyla ilerler. Hatalı yapılar kullanıcıya bildirilir.

# 5. Gerçek Zamanlı Vurgulama (Highlighting)

Gerçek zamanlı vurgulama özelliği, tkinter.Text bileşeni üzerinde token türlerine özel **renklendirme** ile uygulanmıştır. Aşağıdaki türlerde en az 5 farklı yapı renklendirilmektedir:

| **Token Türü** | **Renk** |
| --- | --- |
| KEYWORD | Mor |
| ID | Mavi |
| NUMBER | Turuncu |
| OP | Kırmızı |
| COMMENT | Gri |

Kullanıcı her tuşa bastığında (<KeyRelease> olayı), metin yeniden tokenize edilip renklendirilir.

**6. Grafik Arayüz (GUI)**

Projenin grafik arayüzü Python'un tkinter kütüphanesi ile tasarlandı. Arayüzde:

* Kod yazılabilecek bir metin alanı (Text)
* Gerçek zamanlı renklendirme
* Sözdizimi doğruysa/yanlışsa bilgilendirme etiketi
* Kullanıcı dostu, sade ve etkileşimli bir tasarım bulunur

Her değişiklikte hem vurgulama hem de sözdizimi denetimi güncellenir.

**✅ Sonuç**

Bu proje, küçük çaplı bir derleyici tasarımı projesi gibi çalışarak:

* **Lexical analyzer (sözcük çözücü)**
* **Syntax analyzer (sözdizimi çözücü)**
* **Gerçek zamanlı GUI**
* **Canlı sözdizimi vurgulama**  
  gibi bileşenleri başarıyla birleştirmiştir.