***Gerçek Zamanlı Biçimbilgisel Temelli Sözdizimi Vurgulama Arayüzü***

***Giriş***

Bu yazıda, RBLM0238 Programlama Dilleri dersi kapsamında geliştirilen "Gerçek Zamanlı Sözdizimi Vurgulama Arayüzü" projesi detaylı bir şekilde ele alacağız. Amaç, girilen kodun çözümlemesini yaparak farklı token türlerini anında renklendiren bir grafik arayüz sunmaktır. Projede lexer, parser, çözümleme ve arayüz bileşenleri tamamen Python dili kullanılarak sıfırdan yazdım. Herhangi bir hazır vurgulama kütüphanesi kullanılmadım.

***Proje Hedefleri***

* Python dili kullanarak temel bir programlama dili tasarılamak
* Bu dil için bir sözcüksel analiz (tokenizer) geliştirmek
* Gramer kurallarına göre ayrıştırma (parser) uygulamak
* Gerçek zamanlı sözdizimi vurgulaması sağlamak
* GUI tasarlamak ve vurgulamayı bu arayüzle birleştirmek

***Dil ve Gramer Tanımı***

Projede kullanılan dil, Python'a benzer basitleştirilmiş bir pseudo-language olarak tasarladım. Temel yapılar şunlardır:

* Atamalar: x = 3
* Koşullar: if, elif, else, end
* Döngüler: while ... end
* Yazdırma: print("Merhaba")
* Yorum satırları: # Bu bir yorum

***Gramer Taslağı (Basitleştirilmiş)***

Program -> Statement\*

Statement -> Assignment | IfStmt | WhileStmt | PrintStmt | COMMENT

Assignment -> IDENTIFIER '=' Expression

IfStmt -> 'if' Expression ':' Block ( 'elif' Expression ':' Block )\* ( 'else' ':' Block )? 'end'

WhileStmt -> 'while' Expression ':' Block 'end'

PrintStmt -> 'print' '(' Expression (',' Expression)\* ')'

Block -> Statement\*

Expression -> LogicalOr ...

***Sözcüksel Çözümleme (Lexical Analysis)***

Sözcüksel analiz, **tokenizer.py** dosyasında gerçekleştirildi. Kod, tek tek karakterleri okuyarak anlamlı token'lara ayırır. Aşağıdaki token türleri desteklenmektedir:

* KEYWORD: if, else, elif, while, end, true, false, print
* IDENTIFIER: değişken isimleri (x, toplam, vs.)
* NUMBER: 123, 3.14, .5
* STRING: "merhaba"
* SYMBOL: +, -, ==, :, (, ), =, vs.
* COMMENT: # ile başlayan satırlar
* ERROR: hatalı string veya eksik karakterler
* UNKNOWN: tanımlanamayan karakterler

***Örnek:***

x = 4.5

y = x + 1

# bu bir yorum

yaz("Merhaba")

***Ayrıştırma (Syntax Analysis)***

Parser **parser\_module.py** dosyasında yazılmıştır. Recursive descent parsing mantığı kullanılmıştır. Her gramer kuralı bir fonksiyonla temsil edilir. Kod token listesine dönüştürüldükten sonra parser bu listeyi kontrol eder.

***Örnek Kullanım:***

if x > 3:

y = y + 1

else:

y = 0

end

Parser yukarıdaki yapının **if**, **else** bloklarını ve **end** ile kapatılıp kapatılmadığını kontrol eder.

***Vurgulama Şemasi***

Tkinter içindeki Text widget'ının **tag\_configure** fonksiyonuyla her token türüne bir renk tanımlandı.

|  |  |
| --- | --- |
| Token | Renk |
| KEYWORD | Siyah |
| IDENTIFIER | Mavi |
| NUMBER | Yeşil |
| STRING | Magenta |
| SYMBOL | Turuncu |
| COMMENT | Gri |
| ERROR | Kırmızı |
| UNKNOWN | Pembe |

***Grafik Kullanıcı Arayüzü (GUI)***

GUI, **gui.py** dosyasında yer almakta ve **Tkinter** kütüphanesiyle geliştirilmiştir. Arayüzde:

* Kod girilecek metin kutusu
* Scroll bar
* Durum çubuğu
* "Çözümle" ve "Temizle" butonları

***Gerçek Zamanlı Vurgulama***

Metin alanında her tuş bırakıldığında içerik tekrar analiz edilir ve renkler güncellenir. Bu sayede kullanıcı anlık olarak vurgulamaya göre hataları görebilir.

***Testler***

Projede 2 adet test dosyası hazırlandı:

* **test\_tokenizer.py:** sayı, yorum, string ve anahtar kelime kontrolü
* **test\_parser.py:** eksik end, hatalı ifadeler, nested blok testleri

Testler unittest kütüphanesi kullanılarak çalıştırıldı ve başarıyla sonuçlandı.

***Sonuç***

Bu proje ile lexer, parser, vurgulama mantığı ve grafik arayüz entegrasyonu gerçekleştirdim. Tüm bileşenler başka bir kütüphane kullanılmadan Python ile sıfırdan yazdım. Kod renklendirmesi, hata tespiti ve anlık geri bildirim gibi özellikler sade ama işlevsel bir yapıda birleştirdim.