**🎬 Video Tanıtım Senaryosu**

**Proje Adı:** Gerçek Zamanlı Biçimbilgisel Temelli Sözdizimi Vurgulama Arayüzü  
**Hazırlayan:** Rima Farah Eleuch

*(Ekranda proje başlığı belirir, arka planda yumuşak bir müzik varsa çok düşük seviyede)*

**Merhaba, ben Rima Farah Eleuch. Bu videoda sizlere Programlama Dilleri dersi kapsamında geliştirdiğim projenin tanıtımını yapacağım.**

**Projemin adı: Gerçek Zamanlı Biçimbilgisel Temelli Sözdizimi Vurgulama Arayüzü.**

**🧾 1. Proje Amacı ve Genel Tanım**

**Bu proje kapsamında, gerçek zamanlı çalışan bir sözdizimi vurgulayıcı geliştirdim. Proje, üç temel bileşene dayanıyor:**

* **Sözcüksel analiz (lexical analysis),**
* **Sözdizimsel analiz (syntax analysis)**
* ve **bir grafik kullanıcı arayüzü (GUI).**

**Amaç, kullanıcı kod yazarken, yazdığı kodu anında analiz edip belirli token türlerini renklendirerek görsel olarak vurgulamak.**

**💻 2. Kullanılan Teknolojiler**

**Geliştirme dili olarak Python kullandım. Arayüz için Python’un standart kütüphanesi olan Tkinter’ı tercih ettim.**

**Ayrıca sözcüksel analiz ve ayrıştırma işlemlerini de kendi yazdığım modüllerle gerçekleştirdim — herhangi bir hazır sözdizimi kütüphanesi kullanmadım.**

**🔠 3. Sözcüksel Analiz (Tokenizer)**

**Şimdi ilk olarak tokenizer modülünü göstermek istiyorum.**  
*(Kod editöründe* ***tokenizer.py*** *dosyası açılır.)*

**Burada yazdığım kod sayesinde, girilen metni kelime kelime tarıyor, rakamlar, anahtar kelimeler, değişken adları, string ifadeler, semboller ve yorum satırlarını belirliyor.**

**Ayrıca hatalı yazılmış string'ler veya tanınmayan karakterler için ERROR ve UNKNOWN token türleri tanımladım.**

**🧠 4. Sözdizimi Analizi (Parser)**

**Şimdi de parser modülüne bakalım.**  
*(****parser\_module.py*** *dosyası açılır.)*

**Burada üstten-aşağı doğru çalışan, recursive descent parser olarak bilinen yöntemi kullandım. if, while, print, end gibi yapıları analiz edebiliyor**

**Kodda herhangi bir sözdizimi hatası varsa, örneğin eksik end gibi, sistem hata mesajını üretiyor ve kullanıcıya gösteriyor.**

**🖼️ 5. Grafik Arayüz (GUI)**

**Arayüz kısmı ise gui.py dosyasında yer alıyor.**  
*(GUI gösterilirken konuşmaya devam edilir.)*

**Burada Tkinter ile basit ama işlevsel bir arayüz oluşturdum. Kullanıcı soldaki alanda kodunu yazıyor.**

**Klavye üzerinden her tuşa bastığında içerik anında token'lara ayrılıyor ve her tür token kendi rengine göre renklendiriliyor.**

**Örneğin:**

* Anahtar kelimeler siyah,
* Değişkenler mavi,
* Sayılar yeşil,
* String’ler magenta (mor),
* Semboller turuncu,
* Yorum satırları gri,
* Hatalı ifadeler kırmızı olarak gösteriliyor.

**✅ 6. Çalışma Gösterimi**

**Şimdi örnek bir kod bloğu yazalım.**  
*(GUI'de aşağıdaki kod yazılırken anlatım yapılır.)*

#Test kodu

x = 10

if x > 5:

print("Büyük")

elif x < 5:

print("Küçük")

else:

print(“Eşit”)

end

**Kod yazıldıkça vurgulama anlık olarak gerçekleşiyor. Daha sonra 'Çözümle' butonuna basıyorum...**

**Eğer hata yoksa durum çubuğunda 'Çözümleme başarılı' mesajı yeşil renkte gösteriliyor.**

**Ama şimdi örneğin end ifadesini silersem ve tekrar çözümle dersem...**

**Kırmızı renkte bir hata mesajı alıyorum: 'Beklenen: KEYWORD end' gibi.**

**Ayrıca hatalı token da kırmızıyla işaretleniyor.**

**🧪 7. Testler**

**Ayrıca tüm bu işlemleri doğrulamak için birim testler de yazdım.**  
*(Komut satırında* ***test\_tokenizer.py*** *ve* ***test\_parser.py*** *çalıştırılır.)*

**Tokenların doğru algılanıp algılanmadığını ve parser’ın farklı yapıları doğru çözümleyip çözümlemediğini kontrol eden testlerim mevcut.**

**🏁 8. Kapanış**

**Sonuç olarak, bu projede hem biçimbilgisel hem sözdizimsel analiz konularını gerçek zamanlı görsel hale getirmiş oldum.**

**Her şey sıfırdan yazdım, dış kütüphane kullanmadım ve Python dilinin temel araçlarıyla gerçekleştirdim.**

**İzlediğiniz için teşekkür ederim. Umarım beğenmişsinizdir. Hoşça kalın!**

*(Ekranda “Teşekkürler” yazısı belirir ve video kapanır.)*