GUI Tabanlı Python-Benzeri Kod Düzenleyici: Teknik Belgeleme

Bu proje üç ana bileşen içermektedir: (leksik analizör),

lexical\_analyzer.py

parser.py

gui.py

(sözdizimi analizi ve ayrıştırıcı) ve (sözdizimi vurgulama ve grafik arayüz). Uygulama Python ile

yazılmıştır. Oluşturulan mini-dil, Python’a benzer bir sözdizimine sahiptir; blok yapıları girintilerle



if / elif / else



for

def

<label>

belirlenir ve

return

yapıları, while ve

döngüleri,

ile fonksiyon tanımlama ile

ifadeleri desteklenir. Yorumlar # karakteri ile başlar ve kod içinde metin blokları

etiketleri ve renkli işaretlerle vurgulanır. Aşağıda her bileşenin işlevi ve uygulama detayları incelenmektedir.

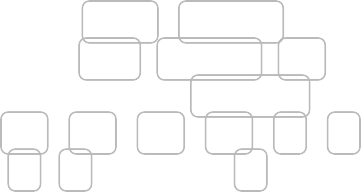
# Dil ve Dilbilgisi Seçimi

Bu düzenleyicide Python tarzı bir dil işlenmektedir. Değişken ve fonksiyon adları bir harf ya da alt çizgi



ID

ile başlayıp harf veya rakamlarla devam edebilir ( token). Tam sayı ve ondalıklı sayılar (örneğin



if

else

while

123 , 3.14 ) NUMBER tokenı olarak tanımlanır. Anahtar kelimeler yalnızca



ID

,

, ,

for , def , return , in şeklindedir; bir tokenının değeri bu anahtar kelimelerden birine eşit

olduğunda türü KEYWORD olarak değiştirilir. Aritmetik ve karşılaştırma operatörleri ( + , - , \* , / ,



== , != , <= , >= , < , > ) OP tokenı; atama operatörü = ise ASSIGN tokenıdır. Parantez ( ( , ) ), iki nokta ( : ) ve virgül ( , ) karakterleri sırasıyla LPAREN , RPAREN , COLON ve tokenlarıdır. Python’dan farklı olarak bu dilde bloklar süslü parantezlerle değil girintilerle ifade edilir. Her

COMMA

INDENT

DEDENT

satır başındaki boşluk miktarı takip edilerek girinti artışında

, azalışında

tokenları

üretilir; böylece blokların başlangıcı ve sonu girintilerle belirlenir. Leksik analizör bu kuralları düzenli ifadeler (regex) ile uygular; derleyici teorisinde leksik analizörün düzenli ifadeler veya DFA kullanarak token tanıması yaygın bir yöntemdir.

# Leksiksel Analiz Detayları

Kaynak kod satır satır okunur ve her parça en uzun eşleşen tokena dönüştürülür. Her satırın başındaki

INDENT

DEDENT

boşluk miktarı ölçülerek girinti artışında

, azalışında

tokenları eklenir; böylece

blokların başlangıcı ve sonu girintilerle belirlenir. Tanımlanan token türleri şunlardır:

* NUMBER: Tam veya ondalıklı sayılar (örneğin , ).

123

3.14

* ID: Değişken veya fonksiyon isimleri; bir harf ya da alt çizgi ile başlayıp harf ve rakamlardan

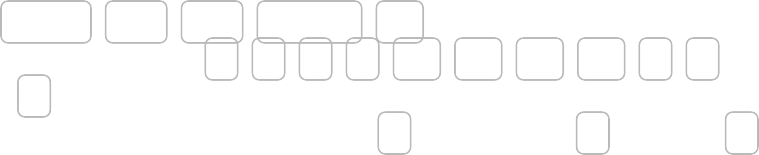


\_temp



ID

oluşur (örneğin myVar ,

türü KEYWORD olarak değiştirilir.

). Bir

tokenının değeri anahtar kelimelerden birine eşitse

* KEYWORD: if , else , while , for , def , return , in gibi özel anahtar kelimeler.
* OP: Aritmetik ve karşılaştırma operatörleri ( + , - , \* , / , == , != , <= , >= , < , > ).
* ASSIGN: Atama operatörü = .
* LPAREN, RPAREN, COLON, COMMA: Sırasıyla sol parantez ( , sağ parantez ) , iki nokta : ,



,

virgül karakterleri.

* NEWLINE: Satır sonu karakteri ( \n ).
* SKIP: Boşluk ve sekme karakterleri (token olarak atlanır).



#

COMMENT

* COMMENT: Satır içi yorumlar (

ile başlayan). Yorum tespit edildiğinde bir

tokenı

eklenir ve satırın geri kalanı işleme alınmaz.

* INDENT, DEDENT: Girinti değişikliklerine bağlı olarak blok başı ( tokenları.

INDENT

DEDENT

) ve sonu ( )

* EOF: Dosya sonu işaretçisi; kaynak tamamen ayrıştırıldıktan sonra eklenir.

Her bir eşleşen parça için tür, değer, satır ve sütun bilgilerini içeren bir Token nesnesi oluşturulur.

SyntaxError

Tanımlanamayan bir karakterle karşılaşılırsa fırlatılır; böylece geçersiz semboller gibi

basit sözdizimi hataları erken aşamada tespit edilip kullanıcıya bildirilebilir.

# Sözdizimi Analizi Süreci

GUI uygulamasında her tuş basımında veya metin değişiminde fonksiyonu çağrılır. Üretilen

lex()

token listesinde COMMENT türündeki öğeler sözdizimi vurgulaması için kullanılırken, sözdizimi denetimi için yorumlar listeden çıkarılır ve kalan tokenlar Parser sınıfına iletilir. Parser(tokens).parse() metodu tüm kodu baştan sona tarar ve yapısal akıştan sapma olduğunda

parse()

statements()

EOF

ParseError fırlatır. Örneğin

metodu önce

yöntemi ile tüm satırları

ayrıştırır, sonra son tokenın dosya sonu simgesi ( ) olduğunu kontrol eder. Hata yoksa arayüzün

durum çubuğunda “Geçerli sözdizimi” mesajı gösterilir; hata varsa ilgili satır ve sütun bilgisiyle bir hata mesajı görüntülenir. Böylece kullanıcıya anlık geribildirim sağlanır. Parser, her dil kuralını ilgili yöntemlerle kontrol eden rekürsif-çıkışlı (recursive descent) bir parser olarak tasarlanmıştır.

# Ayrıştırma (Parsing) Yöntemi

Parser, her bir dil yapısı için ayrı yöntemler içerir. En üstte metodu yer alır; bu metot

statements()

KEYWORD



ID

satır başında bir

veya

tokenı bulunduğu sürece döngüye devam eder ve satırın ilk

kelimesine göre uygun alt yöntemi çağırır. Desteklenen yapılar ve kuralları şunlardır:

* If/Elif/Else Blokları: satırının ardından bir girintili blok beklenir. Gerekli

if <ifade>:

INDENT / DEDENT

else:

tokenları eklenir. İsteğe bağlı elif <ifade>: ve benzer şekilde işlenir; her bloğun sonunda DEDENT ile çıkılır.



blokları da

* While Döngüsü: while <ifade>: yapısı ile başlanır. Parser önce while ve koşul ifadesini okur, ardından : ve yeni satırı kontrol eder; iç blok statements() yöntemi ile ayrıştırılır.
* For Döngüsü: for <ID> in <ifade>: biçimindedir. Bu yapıda for anahtar kelimesi, bir

değişken adı, in anahtar kelimesi ve bir ifade yer alır. İki nokta ardından yeni satırda girinti gereklidir; iç blok statements() yöntemi ile işlenir.

* Fonksiyon Tanımı: def <ID>(parametreler): satırı işlenir. Parametreler bir veya daha fazla



ID

olabilir, virgülle ayrılır. İki noktadan sonraki yeni satır ve girintili blok (fonksiyon gövdesi) kontrol edilir.

return <ifade>

* Return İfadesi:

gelmelidir.

ID = <ifade>

* Atama İfadesi:

ifade gelmesi beklenir.

satırı ayrıştırılır; satır sonunda yeni satır veya blok sonu biçiminde ele alınır. Önce değişken adı, ardından eşittir ve bir

* İfade (Expression): Aritmetik ve karşılaştırmalı ifadeler için kurallar tanımlanmıştır. Örneğin



term

toplama/çıkarma add\_expr , çarpma/bölme ile ayrıştırılır; bir factor olarak sayı

NUMBER

( ), değişken ( ID ), parantezli alt ifade veya fonksiyon çağrısı kabul edilir. Her kurala

uygun yöntemlerle rekürsif ayrıştırma yapılır; beklenmeyen bir token görüldüğünde hata fırlatılır.

# Sözdizimi Vurgulama Şeması

Editör penceresinde sözdizimi vurgulaması (syntax highlighting) token türlerine göre renkli etiketler

keyword

number

comment

operator

kullanılarak yapılır. Tkinter’ın Text widget’ı üzerinde

,

,

,

gibi

etiketler tanımlanır ve her birine renk atanır (örneğin anahtar kelimeler mavi, sayılar mor, yorumlar

on\_key\_release

yeşil). Kullanıcı kod yazarken her tuş basımında gerçekleşen olayı ile önce mevcut

text.tag\_remove

etiketler ile temizlenir.

Ardından her token için uygun etiket adı belirlenir (örneğin

KEYWORD

number

text.tag\_add

için

,

için

vb.) ve fonksiyonu kullanılarak bu etiketler ilgili kod aralığına uygulanır.

Sonuç olarak editörde renkli vurgulamalar görünür hale gelir ve kodun yapısı görsel olarak ayrılmış olur.

keyword

NUMBER



if

keyword

123

number

Örneğin

kelimesi mavi

etiketiyle,

sayısı mor

etiketiyle vurgulanır.

# GUI Uygulaması

Uygulamanın arayüzü Tkinter kütüphanesi ile oluşturulmuştur. Ana pencerede Consolas fontunda geniş bir Text widget bulunur; bu widget kod girişi için metin alanı sağlar. Pencerenin altında yer alan durum çubuğu ise bir Label ile gösterilir; bu kısımda sözdizimi hatası veya başarı mesajları görüntülenir.

text.get("1.0","end-1c")

Her tuş basımında tetiklenen

on\_key\_release

fonksiyonu,

ile kodu

alır ve lex() ile tokenlara ayırır. Eski etiketler temizlendikten sonra yenileri uygulanır ve Parser(tokens).parse() ile sözdizimi kontrolü yapılır. Hata bulunursa durum etiketine ilgili hata mesajı yazılır; hata yoksa “Geçerli sözdizimi” mesajı görüntülenir.

Bu işlev text.bind("<KeyRelease>", on\_key\_release) koduyla klavye olayıyla ilişkilendirilmiştir. Pencere döngüsü root.mainloop() ile başlatılır. Böylece kod düzenleme, eşzamanlı sözdizimi denetimi ve vurgulama işlevleri bir arada çalışır.

# Demo ve Yayımlanan Makale

Projenin temel özelliklerini gösteren bir demo videosu halka açık bir platformda (örneğin YouTube) yayımlanmıştır. Ayrıca teknik uygulama detaylarını anlatan kapsamlı bir makale web üzerinde erişilebilir durumdadır. Bu teknik doküman, öğretmenler ve geliştiricilerin düzenleyiciyi anlayıp genişletmeleri için gerekli tüm bilgileri içermektedir. Aynı zamanda bu belge ders materyali veya araştırma amaçlı çalışmalar için de uygun bir kaynak olarak düzenlenmiştir.