Chapter 2

Programming Language Syntax

Objective

Today's primary objective:

- Define the terms token and lexeme, and tell how they fit into the process of language translation
- Define regular expressions that can be used to identify tokens
- Practice writing regular expressions

Lexeme & Tokens

Lexeme – lowest-level syntactic unit of a language

Token – category of lexemes

Consider the program:

sum := sum + 10;

Lexeme	Token
sum	identifier
:=	assign_op
+	add_op
10	int_literal
;	semicolon

Syntax

Specify rules	Program translation	Basis
Regular expressions	Scanners / lexical analyzers, Unix tools lex and flex	Deterministic Finite Automatons (DFAs)
Context-free grammars	Parsers, Unix tools yacc and bison	Push-Down Automatons (PDAs)

Definition of Regular Expression

Regular expressions are defined recursively as:

Basis

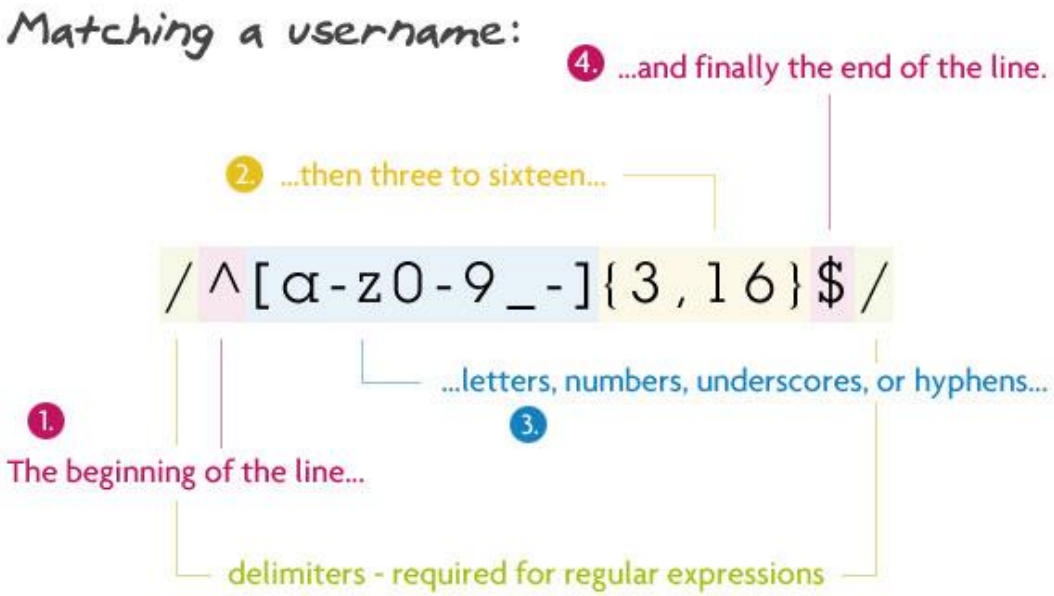
- Character in the alphabet
- Empty string, ε

Induction

- Concatenation, ° or just write symbols beside each other
- Alteration, |
- Kleene closure, * - zero or more

Use parenthesis to avoid ambiguity.

RegEx: Matching a Username



We begin by telling the scanner to find the beginning of the string (^), followed by any lowercase letter (a-z), number (0-9), an underscore, or a hyphen. Next, {3,16} makes sure that are at least 3 of those characters, but no more than 16. Finally, we want the end of the string ($).

String Matched:
my-us3r_n4m3

Not Matched:
th1s1s-wayt00_l0ngt0