In theoretical terms, any token set can be matched by regular expressions as long as it is pre-defined. In terms of historical implementations, regexes were originally written to use ASCII characters as their token set though regex libraries have supported numerous other character sets. Many modern regex engines offer at least some support for Unicode. In most respects it makes no difference what the character set is, but some issues do arise when extending regexes to support Unicode. Supported encoding. Some regex libraries expect to work on some particular encoding instead of on abstract Unicode characters. Many of these require the UTF-8 encoding, while others might expect UTF-16, or UTF-32. In contrast, Perl and Java are agnostic on encodings, instead operating on decoded characters internally. Supported Unicode range. Many regex engines support only the Basic Multilingual Plane, that is, the characters which can be encoded with only 16 bits. Currently (as of 2016) only a few regex engines (e.g., Perl's and Java's) can handle the full 21-bit Unicode range. Extending ASCII-oriented constructs to Unicode. For example, in ASCII-based implementations, character ranges of the form [x-y] are valid wherever x and y have code points in the range [0x00,0x7F] and codepoint(x) ≤ codepoint(y). The natural extension of such character ranges to Unicode would simply change the requirement that the endpoints lie in [0x00,0x7F] to the requirement that they lie in [0x0000,0x10FFFF]. However, in practice this is often not the case. Some implementations, such as that of gawk, do not allow character ranges to cross Unicode blocks. A range like [0x61,0x7F] is valid since both endpoints fall within the Basic Latin block, as is [0x0530,0x0560] since both endpoints fall within the Armenian block, but a range like [0x0061,0x0532] is invalid since it includes multiple Unicode blocks. Other engines, such as that of the Vim editor, allow block-crossing but the character values must not be more than 256 apart.[46] Case insensitivity. Some case-insensitivity flags affect only the ASCII characters. Other flags affect all characters. Some engines have two different flags, one for ASCII, the other for Unicode. Exactly which characters belong to the POSIX classes also varies. Cousins of case insensitivity. As ASCII has case distinction, case insensitivity became a logical feature in text searching. Unicode introduced alphabetic scripts without case like Devanagari. For these, case sensitivity is not applicable. For scripts like Chinese, another distinction seems logical: between traditional and simplified. In Arabic scripts, insensitivity to initial, medial, final, and isolated position may be desired. In Japanese, insensitivity between hiragana and katakana is sometimes useful. Normalization. Unicode has combining characters. Like old typewriters, plain base characters (white spaces, punctuations, symbols, digits, or letters) can be followed by one or more non-spacing symbols (usually diacritics, like accent marks modifying letters) to form a single printable character; but Unicode also provides a limited set of precomposed characters, i.e. characters that already include one or more combining characters. A sequence of a base character + combining characters should be matched with the identical single precomposed character (only some of these combining sequences can be precomposed into a single Unicode character, but infinitely many other combining sequences are possible in Unicode, and needed for various languages, using one or more combining characters after an initial base character; these combining sequences may include a base character or combining characters partially precomposed, but not necessarily in canonical order and not necessarily using the canonical precompositions). The process of standardizing sequences of a base character + combining characters by decomposing these canonically equivalent sequences, before

reordering them into canonical order (and optionally recomposing some combining characters into the leading base character) is called normalization. New control codes. Unicode introduced amongst others, byte order marks and text direction markers. These codes might have to be dealt with in a special way. Introduction of character classes for Unicode blocks, scripts, and numerous other character properties. Block properties are much less useful than script properties, because a block can have code points from several different scripts, and a script can have code points from several different blocks.[47] In Perl and the java.util.regex library, properties of the form \p{InX} or \p{Block=X} match characters in block X and \P{InX} or \P{Block=X} matches code points not in that block. Similarly, \p{Armenian}, \p{IsArmenian}, or \p{Script=Armenian} matches any character in the Armenian script. In general, \p{X} matches any character with either the binary property X or the general category X. For example, \p{Lu}, \p{Uppercase_Letter}, or \p{GC=Lu} matches any uppercase letter. Binary properties that are not general categories include \p{White_Space}, \p{Alphabetic}, \p{Math}, and \p{Dash}. Examples of non–binary properties are \p{Bidi_Class=Right_to_Left}, \p{Word_Break=A_Letter}, and \p{Numeric_Value=10}.