International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 11 Issue 7, July -2021,
ISSN(O): 2249-3905, ISSN(P) : 2349-6525 | Impact Factor: 7.196| Thomson Reuters ID: L-5236-2015

Python - String Concatenation

AUTHOR-MS R.JOTHI

CO-AUTHOR-S. GOWRI

DR N.ELAVARASAN

INTRODUCTION-

"string" to mean any items arranged in a line, series or succession dates back centuries.[5][6] In 19th-Century typesetting, compositors used the term "string" to denote a length of type printed on paper; the string would be measured to determine the compositor's pay.[7][4][8]

Use of the word "string" to mean "a sequence of symbols or linguistic elements in a definite order" emerged from mathematics, symbolic logic, and linguistic theory to speak about the formal behavior of symbolic systems, setting aside the symbols' meaning.[4]

For example, logician C. I. Lewis wrote in 1918:[9]

A mathematical system is any set of strings of recognisable marks in which some of the strings are taken initially and the remainder derived from these by operations performed according to rules which are independent of any meaning assigned to the marks. That a system should consist of 'marks' instead of sounds or odours is immaterial.

According to Jean E. Sammet, "the first realistic string handling and pattern matching language" for computers was COMIT in the 1950s, followed by the SNOBOL language of the early 1960s.[10]

A **string datatype** is a datatype modeled on the idea of a formal string. Strings are such an important and useful datatype that they are implemented in nearly every programming language. In some languages they are available as primitive types and in others as composite types. The syntax of most high-level programming languages allows for a string, usually quoted in some way, to represent an instance of a string datatype; such a meta-string is called a *literal* or *string literal*.

String length[edit]

Although formal strings can have an arbitrary finite length, the length of strings in real languages is often constrained to an artificial maximum. In general, there are two types of string datatypes: *fixed-length strings*, which have a fixed maximum length to be determined at compile time and which use the same amount of memory whether this maximum is needed or not, and *variable-length strings*, whose length is not arbitrarily fixed and which can use varying amounts of memory depending on the actual requirements at run time (see Memory management). Most strings in modern programming languages are variable-length strings. Of course, even variable-length strings are limited in length – by the size of available computer memory. The string length can be stored as a separate integer (which may put another artificial limit on the length) or implicitly through a termination character, usually a character value with all bits zero such as in C programming language. See also "Null-terminated" below.

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 11 Issue 7, July -2021,
ISSN(O): 2249-3905, ISSN(P) : 2349-6525 | Impact Factor: 7.196| Thomson Reuters ID: L-5236-2015

Character encoding

String datatypes have historically allocated one byte per character, and, although the exact character set varied by region, character encodings were similar enough that programmers could often get away with ignoring this, since characters a program treated specially (such as period and space and comma) were in the same place in all the encodings a program would encounter. These character sets were typically based on ASCII or EBCDIC. If text in one encoding was displayed on a system using a different encoding, text was often mangled, though often somewhat readable and some computer users learned to read the mangled text.

Logographic languages such as Chinese, Japanese, and Korean (known collectively as CJK) need far more than 256 characters (the limit of a one 8-bit byte per-character encoding) for reasonable representation. The normal solutions involved keeping single-byte representations for ASCII and using two-byte representations for CJK ideographs. Use of these with existing code led to problems with matching and cutting of strings, the severity of which depended on how the character encoding was designed. Some encodings such as the EUC family guarantee that a byte value in the ASCII range will represent only that ASCII character, making the encoding safe for systems that use those characters as field separators. Other encodings such as ISO-2022 and Shift-JIS do not make such guarantees, making matching on byte codes unsafe. These encodings also were not "self-synchronizing", so that locating character boundaries required backing up to the start of a string, and pasting two strings together could result in corruption of the second string.

Unicode has simplified the picture somewhat. Most programming languages now have a datatype for Unicode strings. Unicode's preferred byte stream format UTF-8 is designed not to have the problems described above for older multibyte encodings. UTF-8, UTF-16 and UTF-32 require the programmer to know that the fixed-size code units are different from the "characters", the main difficulty currently is incorrectly designed APIs that attempt to hide this difference (UTF-32 does make *code points* fixed-sized, but these are not "characters" due to composing codes).

Implementations

Some languages, such as C++, Perl and Ruby, normally allow the contents of a string to be changed after it has been created; these are termed *mutable* strings. In other languages, such as Java, JavaScript, Lua, Python, and Go, the value is fixed and a new string must be created if any alteration is to be made; these are termed *immutable* strings. Some of these languages with immutable strings also provide another type that is mutable, such as Java and .NET's `StringBuilder`, the thread-safe Java `StringBuffer`, and the Cocoa `NSMutableString`. There are both advantages and disadvantages to immutability: although immutable strings may require inefficiently creating many copies, they are simpler and completely thread-safe.

Strings are typically implemented as arrays of bytes, characters, or code units, in order to allow fast access to individual units or substrings—including characters when they have a fixed length. A few languages such as Haskell implement them as linked lists instead.

Some languages, such as Prolog and Erlang, avoid implementing a dedicated string datatype at all, instead adopting the convention of representing strings as lists of character codes.

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 11 Issue 7, July -2021,
ISSN(O): 2249-3905, ISSN(P) : 2349-6525 | Impact Factor: 7.196| Thomson Reuters ID: L-5236-2015

Representations

Representations of strings depend heavily on the choice of character repertoire and the method of character encoding. Older string implementations were designed to work with repertoire and encoding defined by ASCII, or more recent extensions like the ISO 8859 series. Modern implementations often use the extensive repertoire defined by Unicode along with a variety of complex encodings such as UTF-8 and UTF-16.

The term *byte string* usually indicates a general-purpose string of bytes, rather than strings of only (readable) characters, strings of bits, or such. Byte strings often imply that bytes can take any value and any data can be stored as-is, meaning that there should be no value interpreted as a termination value.

Most string implementations are very similar to variable-length arrays with the entries storing the character codes of corresponding characters. The principal difference is that, with certain encodings, a single logical character may take up more than one entry in the array. This happens for example with UTF-8, where single codes (UCS code points) can take anywhere from one to four bytes, and single characters can take an arbitrary number of codes. In these cases, the logical length of the string (number of characters) differs from the physical length of the array (number of bytes in use). UTF-32 avoids the first part of the problem.

String processing algorithms

"Stringology" redirects here. For the physical theory, see String theory.

There are many algorithms for processing strings, each with various trade-offs. Competing algorithms can be analyzed with respect to run time, storage requirements, and so forth. The name **stringology** was coined in 1984 by computer scientist ZviGalil for the theory of algorithms and data structures used for string processing.[18][19][20]

Some categories of algorithms include:

- String searching algorithms for finding a given substring or pattern
- String manipulation algorithms
- Sorting algorithms
- Regular expression algorithms
- Parsing a string
- Sequence mining

Advanced string algorithms often employ complex mechanisms and data structures, among them suffix trees and finite-state machines.

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 11 Issue 7, July -2021,
ISSN(O): 2249-3905, ISSN(P) : 2349-6525 | Impact Factor: 7.196| Thomson Reuters ID: L-5236-2015

Character string-oriented languages and utilities

Character strings are such a useful datatype that several languages have been designed in order to make string processing applications easy to write. Examples include the following languages:

- awk
- Icon
- MUMPS
- Perl
- Rexx
- Ruby
- sed
- SNOBOL
- Tcl
- TTM

Many Unix utilities perform simple string manipulations and can be used to easily program some powerful string processing algorithms. Files and finite streams may be viewed as strings.

Some APIs like Multimedia Control Interface, embedded SQL or printf use strings to hold commands that will be interpreted.

Many scripting programming languages, including Perl, Python, Ruby, and Tcl employ regular expressions to facilitate text operations. Perl is particularly noted for its regular expression use,[21] and many other languages and applications implement Perl compatible regular expressions.

Some languages such as Perl and Ruby support string interpolation, which permits arbitrary expressions to be evaluated and included in string literals.

Character string functions

See also: Comparison of programming languages (string functions)

String functions are used to create strings or change the contents of a mutable string. They also are used to query information about a string. The set of functions and their names varies depending on the computer programming language.

The most basic example of a string function is the string length function – the function that returns the length of a string (not counting any terminator characters or any of the string's internal structural information) and does not modify the string. This function is often named `length` or `len`. For example, `length("hello world")` would return 11. Another common function is concatenation, where a new string is created by appending two strings, often this is the + addition operator.

Some microprocessor's instruction set architectures contain direct support for string operations, such as block copy

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 11 Issue 7, July -2021,
ISSN(O): 2249-3905, ISSN(P) : 2349-6525 | Impact Factor: 7.196| Thomson Reuters ID: L-5236-2015

The "+" operator is well-known as an addition operator, returning the sum of two numbers. However, the "+" symbol acts as string **concatenation operator** in Python. It works with two string operands, and results in the concatenation of the two.

The characters of the string on the right of plus symbol are appended to the string on its left. Result of concatenation is a new string.

```
str1="Hello"
str2="World"
print ("String 1:",str1)
print ("String 2:",str2)
str3=str1+str2
print("String 3:",str3)
```

It will produce the following **output** −

String 1: Hello
String 2: World
String 3: HelloWorld

To insert a whitespace between the two, use a third empty string.

```
str1="Hello"
str2="World"
blank=" "
print ("String 1:",str1)
print ("String 2:",str2)
str3=str1+blank+str2
print("String 3:",str3)
```

It will produce the following **output** −

String 1: Hello
String 2: World
String 3: Hello World

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 11 Issue 7, July -2021,
ISSN(O): 2249-3905, ISSN(P) : 2349-6525 | Impact Factor: 7.196| Thomson Reuters ID: L-5236-2015

Another symbol *, which we normally use for multiplication of two numbers, can also be used with string operands. Here, * acts as a repetition operator in Python. One of the operands must be an integer, and the second a string. The operator concatenates multiple copies of the string. For example −

```
>>> "Hello"*3
'HelloHelloHello'
```

The integer operand is the number of copies of the string operand to be concatenated.

Both the string operators, (*) the repetition operator and (+) the concatenation operator, can be used in a single expression. The "*" operator has a higher precedence over the "+" operator.

```
str1="Hello"
str2="World"
print ("String 1:",str1)
print ("String 2:",str2)
str3=str1+str2*3
print("String 3:",str3)
str4=(str1+str2)*3
print ("String 4:", str4)
```

To form **str3** string, Python concatenates 3 copies of World first, and then appends the result to Hello

```
String 3: HelloWorldWorldWorld
```

In the second case, the strings str1 and str2 are inside parentheses, hence their concatenation takes place first.

REFERENCES-

1. *ntroduction To Java – MFC 158 G". Archived from the original on 2016-03-03. String literals (or constants) are called 'anonymous strings'*
2. ^ *de St. Germain, H. James. "Strings". University of Utah, Kahlert School of Computing.*
3. ^ *Francis, David M.; Merk, Heather L. (November 14, 2019). "DNA as a Biochemical Entity and Data String".*
4. ^ Jump up to:[a b c] *Burchfield, R.W. (1986). "string". A Supplement to the Oxford English Dictionary. Oxford at the Clarendon Press.*
5. ^ *"string". The Oxford English Dictionary. Vol. X. Oxford at the Clarendon Press. 1933.*

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org/journals.php
Vol. 11 Issue 7, July -2021,
ISSN(O): 2249-3905, ISSN(P) : 2349-6525 | Impact Factor: 7.196| Thomson Reuters ID: L-5236-2015

6. ^ *"string (n.)". Online Etymology Dictionary.*
7. ^ *Whitney, William Dwight; Smith, Benjamin E. "string". The Century Dictionary. New York: The Century Company. p. 5994.*
8. ^ *"Old Union's Demise". Milwaukee Sentinel. January 11, 1898. p. 3.*
9. ^ *Lewis, C.I. (1918). A survey of symbolic logic. Berkeley: University of California Press. p. 355.*
10. ^ *Sammet, Jean E. (July 1972). "Programming Languages: History and Future" (PDF). Communications of the ACM. **15** (7). doi:10.1145/361454.361485. S2CID 2003242.*
11. ^ *Bryant, Randal E.; David, O'Hallaron (2003), Computer Systems: A Programmer's Perspective (2003 ed.), Upper Saddle River, NJ: Pearson Education, p. 40, ISBN 0-13-034074-X, archived from the original on 2007-08-06*
12. ^ *Wearmouth, Geoff. "An Assembly Listing of the ROM of the Sinclair ZX80". Archived from the original on August 15, 2015.*
13. ^ *Allison, Dennis. "Design Notes for Tiny BASIC". Archived from the original on 2017-04-10.*
14. ^ Charles Crowley. "Data Structures for Text Sequences" Archived 2016-03-04 at the Wayback Machine. Section "Introduction" Archived 2016-04-04 at the Wayback Machine.
15. ^ "strlcpy and strlcat - consistent, safe, string copy and concatenation." Archived 2016-03-13 at the Wayback Machine
16. ^ "A rant about strcpy, strncpy and strlcpy." Archived 2016-02-29 at the Wayback Machine
17. ^ Keith Thompson. "No, strncpy() is not a "safer" strcpy()". 2012.
18. ^ *"The Prague Stringology Club". stringology.org. Archived from the original on 1 June 2015. Retrieved 23 May 2015.*
19. ^ *Evarts, Holly (18 March 2021). "Former Dean ZviGalil Named a Top 10 Most Influential Computer Scientist in the Past Decade". Columbia Engineering. He invented the terms 'stringology,' which is a subfield of string algorithms,*
20. ^ *Crochemore, Maxime (2002). Jewels of stringology. Singapore. p. v. ISBN 981-02-4782-6. The term stringology is a popular nickname for string algorithms as well as for text algorithms.*
21. ^ *"Essential Perl". Archived from the original on 2012-04-21. Perl's most famous strength is in string manipulation with regular expressions.*
22. ^ *"x86 string instructions". Archived from the original on 2015-03-27.*
23. ^ Jump up to: **a b** *Barbara H. Partee; Alice terMeulen; Robert E. Wall (1990). Mathematical Methods in Linguistics. Kluwer.*
24. ^ *John E. Hopcroft, Jeffrey D. Ullman (1979). Introduction to Automata Theory, Languages, and Computation. Addison-Wesley. ISBN 0-201-02988-X.* Here: sect.1.1, p.1