

The string in PL/SQL is actually a sequence of characters with an optional size specification. The characters could be numeric, letters, blank, special characters or a combination of all. PL/SQL offers three kinds of strings:

- **Fixed-length strings:** In such strings, programmers specify the length while declaring the string. The string is right-padded with spaces to the length so specified.
- **Variable-length strings:** In such strings, a maximum length up to 32,767, for the string is specified and no padding takes place.
- **Character large objects CLOBs:** These are variable-length strings that can be up to 128 terabytes.

PL/SQL strings could be either variables or literals. A string literal is enclosed within quotation marks. For example,

```
'This is a string literal.' Or 'hello world'
```

To include a single quote inside a string literal, you need to type two single quotes next to one another, like:

```
'this isn't what it looks like'
```

## Declaring String Variables

Oracle database provides numerous string datatypes, like, CHAR, NCHAR, VARCHAR2, NVARCHAR2, CLOB, and NCLOB. The datatypes prefixed with an 'N' are 'national character set' datatypes, that store Unicode character data.

If you need to declare a variable-length string, you must provide the maximum length of that string. For example, the VARCHAR2 data type. The following example illustrates declaring and using some string variables:

```
DECLARE
  name varchar2(20);
  company varchar2(30);
  introduction clob;
  choice char(1);
BEGIN
  name := 'John Smith';
  company := 'Infotech';
  introduction := ' Hello! I'm John Smith from Infotech.';
  choice := 'y';
  IF choice = 'y' THEN
    dbms_output.put_line(name);
    dbms_output.put_line(company);
    dbms_output.put_line(introduction);
  END IF;
END;
/
```

When the above code is executed at SQL prompt, it produces the following result:

```
John Smith
Infotech Corporation
Hello! I'm John Smith from Infotech.

PL/SQL procedure successfully completed
```

To declare a fixed-length string, use the CHAR datatype. Here you do not have to specify a

maximum length for a fixed-length variable. If you leave off the length constraint, Oracle Database automatically uses a maximum length required. So following two declarations below are identical:

```
red_flag CHAR(1) := 'Y';  
red_flag CHAR      := 'Y';
```

## PL/SQL String Functions and Operators

PL/SQL offers the concatenation operator || for joining two strings. The following table provides the string functions provided by PL/SQL:

S.N.	Function & Purpose
1	<b>ASCII<sub>x</sub></b> ; Returns the ASCII value of the character x.
2	<b>CHR<sub>x</sub></b> ; Returns the character with the ASCII value of x.
3	<b>CONCAT<sub>x,y</sub></b> ; Concatenates the strings x and y and return the appended string.
4	<b>INITCAP<sub>x</sub></b> ; Converts the initial letter of each word in x to uppercase and returns that string.
5	<b>INSTR<sub>x, find_string[, start][, occurrence]</sub></b> ; Searches for find_string in x and returns the position at which it occurs.
6	<b>INSTRB<sub>x</sub></b> ; Returns the location of a string within another string, but returns the value in bytes.
7	<b>LENGTH<sub>x</sub></b> ; Returns the number of characters in x.
8	<b>LENGTHB<sub>x</sub></b> ; Returns the length of a character string in bytes for single byte character set.
9	<b>LOWER<sub>x</sub></b> ; Converts the letters in x to lowercase and returns that string.
10	<b>LPAD<sub>x, width[, pad_string]</sub></b> ; Pads x with spaces to left, to bring the total length of the string up to width characters.
11	<b>LTRIM<sub>x[, trim_string]</sub></b> ; Trims characters from the left of x.

- 12     **NANVL<sub>x</sub>, value;**  
Returns value if x matches the NaN special value *notanumber*, otherwise x is returned.
- 13     **NLS\_INITCAP<sub>x</sub>;**  
Same as the INITCAP function except that it can use a different sort method as specified by NLSSORT.
- 14     **NLS\_LOWER<sub>x</sub> ;**  
Same as the LOWER function except that it can use a different sort method as specified by NLSSORT.
- 15     **NLS\_UPPER<sub>x</sub>;**  
Same as the UPPER function except that it can use a different sort method as specified by NLSSORT.
- 16     **NLSSORT<sub>x</sub>;**  
Changes the method of sorting the characters. Must be specified before any NLS function; otherwise, the default sort will be used.
- 17     **NVL<sub>x</sub>, value;**  
Returns value if x is null; otherwise, x is returned.
- 18     **NVL2<sub>x</sub>, value1, value2;**  
Returns value1 if x is not null; if x is null, value2 is returned.
- 19     **REPLACE<sub>x</sub>, search\_string, replace\_string;**  
Searches x for search\_string and replaces it with replace\_string.
- 20     **RPAD<sub>x</sub>, width[, pad\_string];**  
Pads x to the right.
- 21     **RTRIM<sub>x</sub>[, trim\_string];**  
Trims x from the right.
- 22     **SOUNDEX<sub>x</sub> ;**  
Returns a string containing the phonetic representation of x.
- 23     **SUBSTR<sub>x</sub>, start[, length];**  
Returns a substring of x that begins at the position specified by start. An optional length for the substring may be supplied.
- 24     **SUBSTRB<sub>x</sub>;**  
Same as SUBSTR except the parameters are expressed in bytes instead of characters for

the single-byte character systems

25     **TRIM***[trim<sub>c</sub>harFROM x]*;

Trims characters from the left and right of x.

26     **UPPER***x*;

Converts the letters in x to uppercase and returns that string.

## Example 1

```
DECLARE
  greetings varchar2(11) := 'hello world';
BEGIN
  dbms_output.put_line(UPPER(greetings));

  dbms_output.put_line(LOWER(greetings));

  dbms_output.put_line(INITCAP(greetings));

  /* retrieve the first character in the string */
  dbms_output.put_line ( SUBSTR (greetings, 1, 1));

  /* retrieve the last character in the string */
  dbms_output.put_line ( SUBSTR (greetings, -1, 1));

  /* retrieve five characters,
     starting from the seventh position. */
  dbms_output.put_line ( SUBSTR (greetings, 7, 5));

  /* retrieve the remainder of the string,
     starting from the second position. */
  dbms_output.put_line ( SUBSTR (greetings, 2));

  /* find the location of the first "e" */
  dbms_output.put_line ( INSTR (greetings, 'e'));
END;
/
```

When the above code is executed at SQL prompt, it produces the following result:

```
HELLO WORLD
hello world
Hello World
h
d
World
ello World
2
```

PL/SQL procedure successfully completed.

## Example 2

```
DECLARE
  greetings varchar2(30) := '.....Hello World.....';
BEGIN
  dbms_output.put_line(RTRIM(greetings, '.'));
  dbms_output.put_line(LTRIM(greetings, '.'));
  dbms_output.put_line(TRIM( '.' from greetings));
END;
/
```

When the above code is executed at SQL prompt, it produces the following result:

```
.....Hello World  
Hello World.....  
Hello World
```

```
PL/SQL procedure successfully completed.
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js